Oracle® Data Provider for .NET

Developer's Guide

12*c* Release 2 (12.2) for Microsoft Windows

E83836-01

March 2017

Oracle Data Provider for .NET Developer's Guide, 12*c* Release 2 (12.2) for Microsoft Windows

E83836-01

Primary Author: Maitreyee Chaliha

Contributing Authors: Sumit Jeloka, Janis Greenberg, Alex Keh, Kiminari Akiyama, Sinclair Hsu, Shailendra Jain, Riaz Ahmed, Ashish Shah, Lakshminarayanan Suriamoorthy, Steven Caminez, Naveen Doraiswamy, Neeraj Gupta, Chithra Ramamurthy, Martha Woo, Arun Singh, Sujith Somanathan, Nishant Singh

Contents

List of Examples

List of Tables

Preface

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

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

This Preface contains these topics:

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

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Related Documents

For more information, see these Oracle resources:

- *Oracle Database Installation Guide for Microsoft Windows*
- *Oracle Database Release Notes for Microsoft Windows*
- *Oracle Database Platform Guide for Microsoft Windows*
- *Oracle Database Administrator's Guide*
- *Oracle Database Development Guide*
- *Oracle Database SecureFiles and Large Objects Developer's Guide*
- *Oracle Real Application Clusters Administration and Deployment Guide*
- *Oracle Database New Features Guide*
- *Oracle Database Concepts*
- *Oracle Database Reference*
- *Oracle Database Extensions for .NET Developer's Guide for Microsoft Windows*
- *Oracle Database Object-Relational Developer's Guide*
- *Oracle Database SQL Language Reference*
- *Oracle Database Net Services Administrator's Guide*
- *Oracle Database Net Services Reference*
- *Oracle Call Interface Programmer's Guide*
- *Oracle Services for Microsoft Transaction Server Developer's Guide for Microsoft Windows*
- *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 Developer's Guide*
- *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.

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

`http://www.oracle.com/technetwork/index.html`

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

`http://docs.oracle.com/database/122/index.htm`

For additional information, see:

`https://msdn.microsoft.com/en-us/default.aspx`

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:

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

Changes in This Release for Oracle Data Provider for .NET

This preface contains:

Changes in Oracle Data Provider for .NET Release 12.2.0.1

The following are the changes in Oracle Data Provider for .NET for Release 12.2.0.1.

New Features

The following features are new in this release:

- .NET Cloud Development and Deployment

  ODP.NET, Managed and Unmanaged Drivers can be deployed easily to Oracle Cloud, private clouds, and third-party cloud environments through Web Deploy. All ODP.NET specific settings no longer require any operating system level configuration. These settings can be made in the .NET configuration files. Managed and Unmanaged ODP.NET Drivers now share a unified configuration file format.

- Application Continuity

Application Continuity recovers incomplete requests from an ODP.NET, Unmanaged Driver perspective and masks many system failures, communication failures, hardware failures, and storage outages from the user.

See also "Application Continuity" for more information.

- Sharding and ODP.NET Routing

  Starting from Release 12.2.0.1, ODP.NET, Unmanaged Driver and Oracle Database support sharding. Oracle Sharding provides the ability to horizontally partition the data across multiple independent Oracle databases (shards). Based on a key specified in the connect string, ODP.NET can route the database requests to a particular shard.

  Oracle Sharding is a shared-nothing architecture that allows near-linear scaling of the database across low-cost commodity database servers located in one or more local or global data centers. Other key benefits include global data distribution (store particular data close to consumers) and fault containment (failure of one shard does not affect the availability of other shards). Global Data Services manages the location of data among the shards and allows ODP.NET client requests to be routed to the appropriate shard in this distributed database system.

  See also "Database Sharding" for more information.

- Longer Schema Identifiers

  Oracle Data Provider for .NET now supports schema object identifier names, such as tables, columns, views, stored procedures, and functions, up to 128 characters in length. This feature is available in both ODP.NET, Managed and Unmanaged Drivers.

- ODP.NET, Managed Driver – Data Integrity

  ODP.NET, Managed Driver supports cryptographic hash functions to better ensure data integrity between the database server and the client. The algorithms supported include `MD5`, `SHA-1`, and `SHA-2` (`SHA-256`, `SHA-384`, and `SHA-512`).

  See also "settings section" and "Network Data Encryption and Integrity" for more information.

- ODP.NET, Managed Driver -- Transport Layer Security (TLS)

  ODP.NET, Managed Driver has added support for TLS 1.1 and 1.2 in addition to existing support for TLS 1.0 and SSL 3.0.

- ODP.NET, Managed Driver -- Distinguished Name for SSL/TLS

  ODP.NET, Managed Driver connections using SSL/TLS can ensure that the distinguished name (DN) is correct for the database server that it is trying to connect to.

- ODP.NET, Managed Driver - Boolean Data Type

  ODP.NET, Managed Driver now supports the `OracleBoolean` data type when using the database's PL/SQL `Boolean` data type. The managed driver must be connected to Oracle Database 12*c* Release 2 (12.2) or higher. Booleans store `TRUE` or `FALSE` values.

  The ODP.NET `OracleBoolean` data type eases parameter binding and data type mapping setup with Boolean values.

  See also "OracleBoolean Structure" for more information.

Desupported Features

Some features previously described in this document are desupported in Oracle Database 12*c* Release 2 (12.2). See *Oracle Database Upgrade Guide* for a complete list of desupported features in this release.

The following features are no longer supported by Oracle:

- `OracleLogicalTransactionStatus` class
- `OracleConnection.GetLogicalTransactionStatus` method
- `OracleConnection.LogicalTransactionId` property
- `OracleConnection.OracleLogicalTransaction` property
- `OracleLogicalTransaction.DataSource` property
- `OracleLogicalTransaction.GetOutcome()` method
- `OracleLogicalTransaction.GetOutcome(sting, string, string)` method
- `OracleLogicalTransaction.UserId` property

Changes in Oracle Data Provider for .NET in ODAC 12*c* Release 4

The following are the changes in Oracle Data Provider for .NET for ODAC 12*c* Release 4.

New Features

The following features are new in this release:

- .NET Framework 4.6 and 4.6.1 Certification

  ODP.NET, Managed and Unmanaged Drivers are certified for .NET Framework 4.6 and 4.6.1.

  See also "System Requirements" for more information.
- ODP.NET, Managed Driver - Windows Installer

  ODP.NET, Managed Driver is now available as part of an ODAC Microsoft Windows Installer package.
- ODP.NET, Managed Driver - Network Data Encryption

  ODP.NET, Managed Driver supports database security network data encryption using Advanced Encryption Standard (AES), RC4, or Triple-DES to enable more secure database communication over intranet and cloud access.

  See also "settings section" and "Network Data Encryption and Integrity" for more information.
- ODP.NET, Managed Driver - Secure External Password Store

  ODP.NET, Managed Driver supports connection establishment by retrieving password credentials from a client-side Oracle wallet.

  See also "Using Secure External Password Store" for more information.

- ODP.NET, Managed Driver - Microsoft Local Security Authority (MSLSA)

  ODP.NET, Managed Driver now supports the Kerberos credential cache type, MSLSA. MSLSA is used to access the Microsoft Kerberos Logon Session credentials cache.

  See also "Using Kerberos" for more information.

- ODP.NET, Managed Driver - SSL/TLS Connections Use a Single Port

  An ODP.NET, Managed Driver SSL/TLS connection will now continue on the original connection to the database listener instead of the previous SSL/TLS client redirection to a database server created new listening endpoint on a dynamic (ephemeral) port. Hence, firewalls will only need to allow access to the TNS listener's port. For example, 1521.

  See also "Using Transport Layer Security and Secure Sockets Layer" for more information.

- Service Relocation Connection Timeout

  Whenever a database service becomes unavailable, an application can encounter numerous connectivity errors. To avoid connection attempts to an unavailable service, ODP.NET, Managed and Unmanaged Drivers block any connection attempts until the service is up or until the configured time limit expires from the time when the service DOWN event was received. This feature is useful for planned outages and service relocations. It works with Oracle RAC and Oracle Data Guard.

  See also "`ServiceRelocationConnectionTimeout`" for more information.

- ODP.NET, Unmanaged Driver - Transaction Guard

  Transaction Guard allows ODP.NET applications to use at-most-once execution in case of planned and unplanned outages and repeated submissions. This feature's architecture has been modified to simplify the application code needed for transaction recovery. Developers will find it easier to utilize Transaction Guard in their applications.

  See also "Using Transaction Guard to Prevent Logical Corruption" for more information.

- ODP.NET, Managed Driver - Transaction Guard

  ODP.NET, Managed Driver now supports Transaction Guard. Its API and architecture are the same as ODP.NET, Unmanaged Driver's in ODAC 12c Release 4 to provide improved developer productivity.

  See also "Using Transaction Guard to Prevent Logical Corruption" for more information.

- ODP.NET, Unmanaged Driver - Managed Code for Distributed Transactions

  In .NET Framework 4.5.2 or higher, ODP.NET, Unmanaged Driver includes managed code for distributed transaction enlistment and commitment services using Microsoft Distributed Transaction Coordinator. Previously, applications had to use Oracle Services for Microsoft Transaction Server for these services. This new feature simplifies setup and deployment of ODP.NET, Unmanaged Driver applications that use distributed transactions.

  See also "Distributed Transactions" for more information.

- ODP.NET, Unmanaged Driver - SQL Translation Framework

Introduced in Oracle Database 12c, SQL Translation Framework helps migrate .NET client applications that use SQL statements with vendor-proprietary syntax to semantically-equivalent Oracle syntax.

The framework automatically translates non-Oracle SQL to Oracle SQL, thereby enabling existing client-side application code to run largely unchanged against an Oracle Database. This reduces the cost of migration to Oracle Database significantly.

See also "Database Application Migration: SQL Translation Framework" and "SQL Translation Framework Configuration" for more information.

- Tracing Enhancements

  ODP.NET improves and unifies tracing features between managed and unmanaged ODP.NET. Key features include traces now output to a Windows temporary files directory and both providers use the same tracing parameters.

  See also "Debug Tracing" for more information.

Changes in Oracle Data Provider for .NET in ODAC 12*c* Release 3

The following are the changes in Oracle Data Provider for .NET for ODAC 12*c* Release 3.

New Features

The following features are new in this release:

- Entity Framework Code First and Code First Migrations

  In Entity Framework 6 and higher, managed and unmanaged ODP.NET support Code First and Code First Migrations.

  See also "ADO.NET Entity Framework and LINQ to Entities" for more information.

- Entity Framework 6

  ODP.NET, Managed and Unmanaged Drivers are certified and supported natively for Entity Framework version 6.

  See also "Entity Framework Requirements" for more information.

- NuGet

  ODP.NET, Managed Driver is available in a NuGet package. This feature simplifies distributing customized ODP.NET, Managed Driver to developers.

  The Entity Framework assembly for Code First and Entity Framework 6 is available as a separate NuGet package.

  NuGet is the package manager for Microsoft .NET. NuGet can install software by copying library files to a .NET solution and automatically updating the project accordingly by adding references and updating config files.

  See also "Installing Oracle Data Provider for .NET, Managed Driver" for more information.

- ODP.NET, Managed Driver - XML DB APIs

ODP.NET, Managed Driver now supports all ODP.NET XML classes supported by ODP.NET, Unmanaged Driver.

- Distributed Transactions without `Oracle.ManagedDataAccessDTC.dll`

  The `Oracle.ManagedDataAccessDTC.dll` assembly is no longer required for distributed transaction applications running in .NET Framework 4.5.2 or higher and ODP.NET, Managed Driver. Upon ODP.NET installation, `Oracle.ManagedDataAccessDTC.dll` is no longer placed into the Global Assembly Cache (GAC). For applications that use .NET Framework 4.5.1 or earlier, `Oracle.ManagedDataAccessDTC.dll` needs to either be placed in the application directory or in the GAC.

- ODP.NET, Managed Driver - Kerberos

  Kerberos is a network authentication service for security in distributed environments. ODP.NET, Managed Driver can now use Kerberos for single sign-on and centralized user authentication.

  See also "Using Kerberos" for more information.

- ODP.NET, Managed Driver - Implicit Ref Cursor

  ODP.NET, Managed Driver introduces support for the new Oracle Database 12c Implicit Ref Cursor. Configuration occurs using the <implicitrefcursor> .NET configuration section. When using database implicit ref cursors, the bindInfo element should be specified with a mode of "Implicit":

  ```
  <bindinfo mode="Implicit" />
  ```

  See also "implicitRefCursor section" for more information.

- Configuration Files: Unified Managed and Unmanaged ODP.NET Format

  ODP.NET, Unmanaged Driver now has the option of using the same configuration file format as ODP.NET, Managed Driver. The format simplifies configuration by using a single unified scheme. To utilize this format, the existing unmanaged ODP.NET configuration section should be renamed from <oracle.dataaccess.client> to <oracle.unmanageddataaccess.client>. The existing unmanaged ODP.NET elements and values are supported within the new section using the same format as with ODP.NET, Managed Driver.

  The traditional ODP.NET, Unmanaged Driver configuration file format will continue to be supported.

  See Also "Configuration File Support" for more information.

Changes in Oracle Data Provider for .NET Release 12.1.0.2

The following are the changes in Oracle Data Provider for .NET for Release 12.1.0.2.

New Features

The following features are new in this release:

- .NET Framework 4.5.2 Certification

  ODP.NET, Managed and Unmanaged Drivers are certified for .NET Framework 4.5.2.

  See also "System Requirements" for more information.

- Character Data Types Extended to 32 KB

  ODP.NET, Managed Driver supports the `VARCHAR2`, `NVARCHAR2`, and `RAW` data types up to 32 KB in size. No code changes are required to use the larger data types.

  By being able to store more data, developers can use these data types more frequently, providing programming flexibility. In addition, SQL Server to Oracle Database application migration is easier with these new data type sizes.

- Return Number of Rows Affected from Each Input in Array Binding Operations

  When using array binding to execute multiple DML statements, ODP.NET, Managed Driver provides an array that lists the number of rows affected for each input value from the bound array, rather than just the total number of rows affected. This information provides more detailed feedback for the application developer. To retrieve the row count, ODP.NET can call the `OracleCommand.ArrayBindRowsAffected` property.

  With more detailed feedback on the array bound DML execution, the developer can better evaluate the query's efficiency and whether the data changes were correctly applied.

  See Also "ArrayBindRowsAffected" for more information.

Changes in Oracle Data Provider for .NET in ODAC 12*c* Release 2

The following are the changes in Oracle Data Provider for .NET for ODAC 12*c* Release 2.

New Features

The following features are new in this release:

- .NET Framework 4.5.1 Certification

  Oracle Data Provider for .NET is now certified for .NET Framework 4.5.1.

  See also "System Requirements" for more information.

- .NET Framework 4.6 Certification

  Oracle Data Provider for .NET is now certified for .NET Framework 4.6.

  See also "System Requirements" for more information.

- Improvements to ODP.NET, Managed Driver Versioning

  This feature allows unique identification of ODP.NET, Managed Driver assemblies which have the same assembly version number.

  See also "Oracle Data Provider for .NET Versioning Scheme" for more information.

Changes in Oracle Data Provider for .NET in ODAC 12*c* Release 1

The following are the changes in Oracle Data Provider for .NET for ODAC 12*c* Release 1.

New Features

The following feature is new in this release:

- LDAP Connections to Active Directory and Oracle Internet Directory

  ODP.NET, Managed Driver supports TNS alias resolution through a LDAP server/ service, specifically Microsoft Active Directory and Oracle Internet Directory.

  This feature allows ODP.NET, Managed Driver to connect to a database using a directory server/service.

  See also "Lightweight Directory Access Protocol".

Changes in Oracle Data Provider for .NET Release 12.1

The following are the changes in Oracle Data Provider for .NET for Release 12.1.

New Features

The following features are new in this release:

- ODP.NET, Managed Driver

  ODP.NET now includes a fully managed provider version, which is 100% native .NET code. ODP.NET, Managed Driver includes nearly all the features of ODP.NET, Unmanaged Driver and uses the same application programming interface. This makes migrating existing ODP.NET applications to ODP.NET, Managed Driver easier.

  With ODP.NET, Managed Driver, it is easier and faster to deploy ODP.NET. There are fewer assemblies, as few as one to deploy, which also makes patching straightforward, and the install size is smaller at less than 10 MB. Only one ODP.NET, Managed Driver assembly is necessary whether you are using 32-bit or 64-bit .NET Framework. Side-by-side deployment with other ODP.NET versions is simple since there are no unmanaged assemblies to account for. As a fully managed provider, ODP.NET can better integrate with Code Access Security and ClickOnce deployment.

  See also "Installing Oracle Data Provider for .NET, Managed Driver" .

- Support for Pluggable Database

  Pluggable Databases (PDBs) enable an Oracle database to contain a portable collection of schemas, schema objects, and nonschema objects that appears to ODP.NET as a separate database. ODP.NET can seamlessly use PDBs.

  PDBs allow fast database provisioning, fast database redeployment by unplugging and plugging in existing databases, and quick patching or upgrading many databases at the cost of doing it once or by unplugging a PDB and plugging it into a different container database. A machine can run more database instances in the form of PDBs than as individual, monolithic databases. It is also easier to separate application administrator duties from the Oracle system administrator duties.

See Also:

– "Pluggable Databases"

- Support for Auto Increment Identity Column

  Oracle Database 12c Release 1 (12.1) introduces an auto increment identity column. ODP.NET, Unmanaged Driver 12.1 and higher releases support interacting with this column data. Identity columns are generally used to uniquely identify rows in a table when there is no other natural primary key constraint.

  An identity column simplifies .NET development for applications with no natural primary key and eases application migration from databases that have an identity column.

See Also:

– "IdentityInsert"
– "IdentityUpdate"
– "OracleIdentityType Enumeration"

- Support for Character Data Types Extended to 32 KB

  Starting with Oracle Database 12c Release 1 (12.1), ODP.NET, Unmanaged Driver now supports the `VARCHAR2`, `NVARCHAR2`, and `RAW` data types up to 32 KB in size. No code changes are required to use the larger data types.

  By being able to store more data, developers can use these data types more frequently, providing programming flexibility. In addition, SQL Server to Oracle Database application migration is easier with these new data type sizes.

- Boolean Data Type

  Oracle Database 12c Release 1 (12.1) introduces a new PL/SQL Boolean data type, which ODP.NET, Unmanaged Driver can store as an `OracleBoolean` data type. Booleans store `TRUE` or `FALSE` values.

  The ODP.NET `OracleBoolean` data type eases parameter binding and data type mapping setup with Boolean values.

See Also:

– "OracleBoolean Structure"

- Enhanced Implicit REF Cursor Binding

In Oracle Database 12c Release 1 (12.1), ODP.NET 12c can retrieve the results of a SELECT statement run in PL/SQL without an explicit target nor REF CURSOR data type. ODP.NET retrieves result sets from stored procedures implicitly without declaring a return type. It is no longer necessary to declare REF CURSOR metadata in a .NET configuration file, except when using Entity Framework, REF Cursors that can be updated, or constraint metadata is required to be passed to the client side.

This capability simplifies using implicit Oracle result sets. In addition, it eases migration to the Oracle database from other vendor databases that use a similar feature.

> **See Also:**
>
> – "ImplicitRefCursors"
>
> – "Implicit REF CURSOR Binding"

- Return Number of Rows Affected from Each Input in Array Binding Operations

  When using array binding to execute multiple DML statements, Oracle Data Provider for .NET, Unmanaged Driver, now provides an array that lists the number of rows affected for each input value from the bound array, rather than just the total number of rows affected. This information provides more detailed feedback for the application developer. To retrieve the row count, ODP.NET can call the `OracleCommand.ArrayBindRowsAffected` property.

  With more detailed feedback on the array bound DML execution, the developer can better evaluate the query's efficiency and whether the data changes were correctly applied.

> **See Also:**
>
> – "ArrayBindRowsAffected"

- Support for `APPLY` Keyword

  Language Integrated Query (LINQ) is a .NET querying language. At runtime, LINQ is translated into native database SQL before it can query the database. In some circumstances, LINQ uses the non-standard APPLY keyword in its SQL translation for retrieving lateral views. Oracle Database and ODP.NET support the APPLY keyword in Oracle Database 12c Release 1 (12.1) to more fully support LINQ.

  This feature allows the occasional LINQ query that uses SQL APPLY to work seamlessly with ODP.NET and Oracle Database for lateral views.

> **See Also:**
>
> – ADO.NET Entity Framework and LINQ to Entities

- Transaction Guard Support

Transaction Guard in Oracle Database 12c Release 1 (12.1) preserves transaction commit outcomes for ODP.NET, Unmanaged Driver, 12c applications during planned and unplanned outages, preventing applications from repeatedly submitting the same transaction. Applications use a new logical transaction identifier to determine the last open transaction's outcome in a database session following an outage. With the known outcome, the application can confidently determine whether to resubmit the transaction or not. Without Transaction Guard, applications that retry operations following outages by committing duplicate transactions can cause logical corruptions.

Transaction Guard preserves the commit outcome for every transaction and makes it available to ODP.NET applications. It allows ODP.NET developers to maintain at-most-once transaction execution.

See Also:

– "Using Transaction Guard to Prevent Logical Corruption"

- Recoverable Error Detection and Recovery

  After an Oracle Database 12c Release 1 (12.1) failure, ODP.NET, Unmanaged Driver, 12c can determine if a failed transaction is recoverable or not. ODP.NET returns the OracleException `IsRecoverable` property indicating whether the transaction is recoverable. If true, the application can retry the transaction.

  This feature makes determining whether failed transactions are recoverable easier, allowing applications to proceed quickly to the next step in the recovery process.

See Also:

– "Using Transaction Guard to Prevent Logical Corruption"

- Support for Faster and Planned Database Outage

  In Oracle Database 12*c* Release 1 (12.1), a database being brought offline automatically alerts ODP.NET applications of the impending downtime. ODP.NET will then stop allocating new connections and close connections returned to the pool from that particular instance.

  This feature enables databases to be brought offline more quickly and minimizes potential end user disruptions by disallowing new ODP.NET connections to databases being brought offline.

See Also:

– "Using FCF Planned Outage to Minimize Service Disruption"

- Support for Oracle Notification Service

Oracle Notification Service (ONS) is a publish and subscribe service for communicating Fast Application Notification (FAN) events. ODP.NET receives fast connection failover and load balancing messages from the database server through ONS. Previously, ODP.NET used Oracle Advanced Queuing (AQ) as its FAN publish and subscribe service.

Because ONS is a memory-based service, it delivers messages faster than AQ. Using ONS, Oracle consolidates the publish and subscribe service that all Oracle data access drivers use.

See Also:

– "Fast Application Notification"

- Support for Global Data Services

  Global Data Services (GDS) is a capability of Oracle Database 12c that extends the concept of services, which previously only was available in Oracle RAC, to a globally distributed configuration that can include a combination of Oracle RAC, Oracle Data Guard, and Oracle GoldenGate. This allows services to be deployed anywhere within this globally distributed configuration, supporting load balancing, high availability, database affinity, and so on with ODP.NET.

  ODP.NET applications can now more efficiently use database resources on a global basis to improve performance and availability. Applications that utilize the Oracle RAC concept of services can now extend the same benefits of automatic workload management to their Oracle Data Guard and Oracle GoldenGate configurations. Similarly, Oracle Data Guard and Oracle GoldenGate customers can now fully utilize the benefits of services and automatic workload management for their replicated configurations.

  See Also:

  – "Runtime Connection Load Balancing"

  – "Fast Connection Failover (FCF)"

- Transaction and Connection Association

  Connections associate with `System.Transactions` transactions when they enlist either implicitly through `enlist=true` connection string attribute, or explicitly through `OracleConnection.EnlistTransaction()` method. A connection in ODP.NET now, by default, detaches from a transaction only when the connection object is closed or when the transaction object is disposed.

  In earlier ODP.NET releases, the connection would get detached from a transaction under the conditions mentioned earlier and when the transaction was complete (committed, aborted, or timed out). When the transaction timeout elapses before the transaction completes, the connection unbinds itself from the transaction and all subsequent operations on this connection execute in `AutoCommit` mode. Any operations prior to the timeout roll back, but operations performed after the timeout commit. The new transaction unbinding default behavior also alerts users with an exception if transactions time out and subsequent operations execute on this connection before the transaction is disposed. This new behavior provides a consistent transactional experience for the end user, even when a timeout occurs.

See also "LegacyTransactionBindingBehavior" for more information.

- Greater Granular Connection Pool Monitoring

  Performance counters can now monitor at the application domain, pool, or database instance level.

  It is now easier to distinguish which application domains, pools, and instances are healthy and which ones are having problems.

  See Also:

  "Connection Pool Performance Counters"

Changes in Oracle Data Provider for .NET Release 11.2.0.3.20

The following are changes in Oracle Data Provider for .NET for Release 11.2.0.3.20.

New Features

The following feature is new in this release:

- .NET Framework 4.5 and Entity Framework 5 Support

  Oracle Data Provider for .NET supports .NET Framework 4.5 and Entity Framework 5.

  See also "System Requirements" for more information.

Changes in Oracle Data Provider for .NET Release 11.2.0.3

The following are changes in Oracle Data Provider for .NET for Release 11.2.0.3.

New Features

The following features are new in this release:

- 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 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" for detailed information on implicit REF CURSOR parameter binding.

Changes in Oracle Data Provider for .NET Release 11.2.0.2

The following are changes in Oracle Data Provider for .NET for Release 11.2.0.2.

New Features

The following features are new in this release:

- 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, Unmanaged Driver"

- 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:
>
> `http://www.oracle.com/technetwork/database/database-technologies/timesten/overview/index.html`

Changes in Oracle Data Provider for .NET Release 11.2.0.1.2

The following are changes in Oracle Data Provider for .NET for Release 11.2.0.1.2.

New Features

The following features are new in this release:

- Support for Microsoft .NET Framework 4

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

Changes in Oracle Data Provider for .NET Release 11.2

The following are changes in Oracle Data Provider for .NET for Release 11.2.

New Features

The following features are new in this release:

- 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"
> – "ClientInfo"

- 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"

Changes in Oracle Data Provider for .NET Release 11.1.0.7.20

The following are changes in Oracle Data Provider for .NET for Release 11.1.0.7.20.

New Features

The following features are new in this release:

- 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"

- 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 Database Advanced Queuing Support"

- 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 11*g* release 1 (11.1.0.7) and higher.

  See Also:

  "System.Transactions and Promotable Transactions"

- 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:**
>
> – "Code Access Security"
> – "OraclePermission Class"
> – "OraclePermissionAttribute Class"

- 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:**
>
> – Oracle Data Provider for .NET HA Event Classes
> – "HAEvent"
> – "OracleConnection Properties"

- 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"
> – "Shutdown"
> – "Startup"

Changes in Oracle Data Provider for .NET Release 11.1.0.6.20

The following are changes in Oracle Data Provider for .NET for Release 11.1.0.6.20.

New Features

The following features are new in this release:

- 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"

- 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"
> – Oracle Data Provider for .NET UDT-Related Classes

- Bulk Copy Operations

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

> **See Also:**
>
> – "Bulk Copy"
> – Oracle Data Provider for .NET Bulk Copy Classes

- 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:**
>
> "Real Application Clusters and Global Data Services"

- 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 "

- 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"

- 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"

Changes in Oracle Data Provider for .NET Release 11.1

The following are changes in Oracle Data Provider for .NET for Release 11.1.

New Features

The following features are new in this release:

- 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 11*g* release 1 or later database versions.

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

1

Introducing Oracle Data Provider for .NET

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 Assemblies
- Differences between the ODP.NET Managed Driver and Unmanaged Driver
- Using ODP.NET Client Provider in a Simple Application

1.1 .NET Data Access in Oracle: Products and Documentation

This section discusses Oracle Data Provider for .NET and Oracle Database components that use Oracle Data Provider for .NET for data access. It briefly describes what each component does and where to find additional documentation.

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

1.1.1 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.

ODP.NET consists of two drivers: ODP.NET, Managed Driver and ODP.NET, Unmanaged Driver. ODP.NET, Managed Driver is a fully managed ADO.NET provider, consisting of fewer DLLs and smaller install size than ODP.NET, Unmanaged Driver. The managed driver has the same exact application programming interfaces (APIs) as ODP.NET, Unmanaged Driver. However, the managed driver's APIs are a subset of the Unmanaged Driver's APIs.

This guide describes Oracle Data Provider for .NET features, their use, installation, requirements, and classes. The guide distinguishes which classes and APIs are supported for the managed driver, unmanaged driver, .NET stored procedures, and .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 for Microsoft Windows*, 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.

1.1.2 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 Language Reference*
- *Oracle Database SQL Language Reference*
- *Oracle Database Extensions for .NET Developer's Guide for Microsoft Windows*
- *Oracle Database Error Messages Reference*
- Access to Oracle Data Provider for .NET Dynamic Help
- Access to Oracle Providers for ASP.NET Dynamic Help

1.1.3 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 Data Provider for .NET Developer's Guide for Microsoft Windows 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.*

1.1.4 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
- 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 for Microsoft Windows,* which is also provided as dynamic help.

1.1.5 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 promotable 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.

1.1.6 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*.

Note:

TimesTen does not support ODP.NET, Managed Driver.

1.2 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. ODP.NET consists of two drivers: ODP.NET, Managed Driver and ODP.NET, Unmanaged Driver. ODP.NET, Managed Driver is a fully managed ADO.NET provider, consisting of fewer DLLs and smaller install size than ODP.NET, Unmanaged Driver. The managed driver has the same exact application programming interfaces (APIs) as ODP.NET, Unmanaged Driver. However, the managed driver's APIs are a subset of the Unmanaged Driver's APIs.

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

- Client Applications: All ODP.NET classes are available for use in client applications.

  As ODP.NET, Managed Driver does not support all classes and members in the ODP.NET, Unmanaged Driver, the unsupported managed driver classes and members will be labeled *Not Supported in ODP.NET, Managed Driver*.
- .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.

1.3 Oracle Data Provider for .NET Assemblies

This section contains the following topics:

- Oracle Data Provider for .NET, Unmanaged Driver Assemblies
- Oracle Data Provider for .NET, Managed Driver Assemblies
- Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces
- Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces

1.3.1 Oracle Data Provider for .NET, Unmanaged Driver Assemblies

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).

To use Code First or Entity Framework 6 or higher with ODP.NET, Unmanaged Driver, add `Oracle.DataAccess.EntityFramework.dll` as a project assembly reference. It contains the namespace `Oracle.DataAccess.EntityFramework`.

1.3.2 Oracle Data Provider for .NET, Managed Driver Assemblies

The `Oracle.ManagedDataAccess.dll` assembly provides two namespaces:

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

ODP.NET, Managed Driver contains additional assemblies. These assemblies are optional to install if not using the specific functionality.

Applications do not need to explicitly add these assemblies to their project. ODP.NET, Managed Driver will access these assemblies by default if installed.

The one exception is `Oracle.ManagedDataAccess.EntityFramework.dll`. That DLL must be explicitly added to a project for its functionality to be used.

- `Oracle.ManagedDataAccessDTC.dll` - Only required when using distributed transactions. The assembly is fully managed, but has 32-bit and x64 versions depending on the .NET Framework's bitness in which it runs. The assembly makes calls to unmanaged assemblies.
- `Oracle.ManagedDataAccess.EntityFramework.dll` - Only required when using Code First or Entity Framework 6 or higher. It contains the `Oracle.ManagedDataAccess.EntityFramework` namespace.
- `Oracle.ManagedDataAccessIOP.dll` - Only required when using Kerberos. The assembly has 32-bit and x64 versions depending on the .NET Framework's bitness in which it runs. The assembly makes calls to unmanaged assemblies. Applications do not need to explicitly add this assembly to their project as ODP.NET is already configured to access this assembly by default.

1.3.3 Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces

The `Oracle.DataAccess.Client` and `Oracle.ManagedDataAccess.Client` namespaces 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` and `Oracle.ManagedDataAccess.Client` namespaces. The tables indicate which of them are not supported by ODP.NET, Managed Driver and/or by .NET stored procedures. All are supported by ODP.NET, Unmanaged Driver.

1.3.3.1 Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client

Table 1-1 lists the `Oracle.DataAccess.Client` and `Oracle.ManagedDataAccess.Client` classes and delegates.

Table 1-1 Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client

Class or Delegate	Supported in the ODP.NET, Managed Driver	Supported in .NET Stored Procedures	Description
OnChangeEventHandler Delegate	-	No	The `OnChangedEventHandler` event delegate represents the signature of the method that handles the notification.
OracleAQAgent Class	No	-	The `OracleAQAgent` class represents agents that may be senders or recipients of a message.
OracleAQDequeueOptions Class	No	-	An `OracleAQDequeueOptions` object represents the options available when dequeuing a message from an `OracleAQQueue` object.
OracleAQEnqueueOptions Class	No	-	The `OracleAQEnqueueOptions` class represents the options available when enqueuing a message to an `OracleAQQueue`.
OracleAQMessage Class	No	-	An `OracleAQMessage` object represents a message to be enqueued and dequeued.
OracleAQMessageAvailableEventArgs Class	No	-	The `OracleAQMessageAvailableEventArgs` class provides event data for the `OracleAQQueue.MessageAvailable` event.
OracleAQMessageAvailableEventHandler Delegate	No	-	The `OracleAQMessageAvailableEventHandler` delegate represents the signature of the method that handles the `OracleAQQueue.MessageAvailable` event.
OracleAQQueue Class	No	-	An `OracleAQQueue` object represents a queue.
OracleBulkCopy Class	No	-	An `OracleBulkCopy` object efficiently bulk loads or copies data into an Oracle table from another data source.

Table 1-1 (Cont.) Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client

Class or Delegate	Supported in the ODP.NET, Managed Driver	Supported in .NET Stored Procedures	Description
OracleBulkCopyColumnMapping Class	No	-	The `OracleBulkCopyColumnMapping` class defines the mapping between a column in the data source and a column in the destination database table.
OracleBulkCopyColumnMappingCollection Class	No	-	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`.
OracleDatabase Class	No	-	An `OracleDatabase` object represents an Oracle Database instance.

Table 1-1 (Cont.) Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client

Class or Delegate	Supported in the ODP.NET, Managed Driver	Supported in .NET Stored Procedures	Description
OracleDataReader Class	-	-	An `OracleDataReader` object represents a forward-only, read-only, in-memory result set.
OracleDataSourceEnumerator Class	-	-	An `OracleDataSourceEnumerator` object allows applications to generically obtain a collection of data sources to connect to.
OracleDependency Class	-	No	An `OracleDependency` class represents a dependency between an application and an Oracle database.
OracleError Class	-	-	The `OracleError` object represents an error reported by an Oracle database.
OracleErrorCollection Class	-	-	An `OracleErrorCollection` object represents a collection of `OracleErrors`.
OracleException Class	-	-	The `OracleException` object represents an exception that is thrown when Oracle Data Provider for .NET encounters an error.
OracleFailoverEventArgs Class	No	No	The `OracleFailoverEventArgs` class provides event data for the `OracleConnection.Failover` event.
OracleFailoverEventHandler Delegate	No	No	The `OracleFailoverEventHandler` represents the signature of the method that handles the `OracleConnection.Failover` event.
OracleGlobalization Class	-	-	The `OracleGlobalization` class is used to obtain and set the Oracle globalization settings of the session, thread, and local computer (read-only).

Table 1-1 (Cont.) Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client

Class or Delegate	Supported in the ODP.NET, Managed Driver	Supported in .NET Stored Procedures	Description
OracleHAEventArgs Class	-	-	The `OracleHAEventArgs` class provides event data for the `OracleConnection.HAEvent` event.
OracleHAEventHandler Delegate	-	-	The `OracleHAEventHandler` delegate represents the signature of the method that handles the `OracleConnection.HAEvent` event.
OracleInfoMessageEventArgs Class	-	-	The `OracleInfoMessageEventArgs` object provides event data for the `OracleConnection.InfoMessage` event.
OracleInfoMessageEventHandler Delegate	-	-	The `OracleInfoMessageEventHandler` delegate represents the signature of the method that handles the `OracleConnection.InfoMessage` event.
OracleNotificationEventArgs Class	-	-	The `OracleNotificationEventArgs` class provides event data for a notification.
OracleNotificationRequest Class	-	No	An `OracleNotificationRequest` class represents a notification request to be subscribed in the database.
OracleParameter Class	-	-	An `OracleParameter` object represents a parameter for an `OracleCommand`.
OracleParameterCollection Class	-	-	An `OracleParameterCollection` object represents a collection of `OracleParameters`.
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.

Table 1-1 (Cont.) Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client

Class or Delegate	Supported in the ODP.NET, Managed Driver	Supported in .NET Stored Procedures	Description
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.
OracleRowsCopiedEventHandler Delegate	No	-	The `OracleRowsCopiedEventHandler` delegate represents the method that handles the `OracleRowsCopied` event of an `OracleBulkCopy` object.
OracleRowsCopiedEventArgs Class	No	-	The `OracleRowsCopiedEventArgs` class represents the set of arguments passed as part of event data for the `OracleRowsCopied` event.
OracleRowUpdatedEventArgs Class	-	-	The `OracleRowUpdatedEventArgs` object provides event data for the `OracleDataAdapter.RowUpdated` event.
OracleRowUpdatedEventHandler Delegate	-	-	The `OracleRowUpdatedEventHandler` delegate represents the signature of the method that handles the `OracleDataAdapter.RowUpdated` event.
OracleRowUpdatingEventArgs Class	-	-	The `OracleRowUpdatingEventArgs` object provides event data for the `OracleDataAdapter.RowUpdating` event.
OracleRowUpdatingEventHandler Delegate	-	-	The `OracleRowUpdatingEventHandler` delegate represents the signature of the method that handles the `OracleDataAdapter.RowUpdating` event.

Table 1-1 (Cont.) Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client

Class or Delegate	Supported in the ODP.NET, Managed Driver	Supported in .NET Stored Procedures	Description
OracleShardingKey Class	No	No	An `OracleShardingKey` object can represent either a sharding key or a super sharding key.
OracleTransaction Class	-	No	An `OracleTransaction` object represents a local transaction.
OracleXmlQueryProperties Class	-	-	An `OracleXmlQueryProperties` object represents the XML properties used by the `OracleCommand` class when the `XmlCommandType` property is `Query`.
OracleXmlSaveProperties Class	-	-	An `OracleXmlSaveProperties` object represents the XML properties used by the `OracleCommand` class when the `XmlCommandType` property is `Insert`, `Update`, or `Delete`.

1.3.3.2 Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Enumerations

Table 1-2 lists the client enumerations.

Table 1-2 Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Enumerations

Enumeration	Supported in the ODP.NET, Managed Driver	Supported in .NET Stored Procedures	Description
FailoverEvent Enumeration	No	No	`FailoverEvent` enumerated values are used to specify the state of the failover.
FailoverReturnCode Enumeration	No	No	`FailoverReturnCode` enumerated values are passed back by the application to the ODP.NET provider to request a retry in case of a failover error, or to continue in case of a successful failover.
FailoverType Enumeration	No	No	`FailoverType` enumerated values are used to indicate the type of failover event that was raised.

Table 1-2 (Cont.) Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Enumerations

Enumeration	Supported in the ODP.NET, Managed Driver	Supported in .NET Stored Procedures	Description
OracleAQDequeueMode Enumeration	No	-	The `OracleAQDequeueMode` enumeration type specifies the dequeue mode.
OracleAQMessageDeliveryMode Enumeration	No	-	The `OracleAQMessageDeliveryMode` enumeration type specifies the delivery mode of the message.
OracleAQMessageState Enumeration	No	-	The `OracleAQMessageState` enumeration type identifies the state of the message at the time of dequeue.
OracleAQMessageType Enumeration	No	-	The `OracleAQMessageType` enumeration type specifies the message payload type.
OracleAQNavigationMode Enumeration	No	-	The `OracleAQNavigationMode` enumeration type specifies the navigation mode.
OracleAQNotificationGroupingType Enumeration	No	-	The `OracleAQNotificationGroupingType` enumeration type specifies the notification grouping type.
OracleAQNotificationType Enumeration	No	-	The `OracleAQNotificationType` enumeration type specifies the notification type of the received notification.
OracleAQVisibilityMode Enumeration	No	-	The `OracleAQVisibilityMode` enumeration type specifies whether the enqueue or dequeue operation is part of the current transaction.
OracleBulkCopyOptions Enumeration	No	-	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.
OracleCollectionType Enumeration	-	No	`OracleCollectionType` enumerated values specify whether or not the `OracleParameter` object represents a collection, and if so, specifies the collection type.

Table 1-2 (Cont.) Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Enumerations

Enumeration	Supported in the ODP.NET, Managed Driver	Supported in .NET Stored Procedures	Description
OracleConnectionType Enumeration	No	-	`OracleConnectionType` enumerated values specify whether a particular connection object is associated with an Oracle database connection, a TimesTen database connection, or no physical connection at all.
OracleDBShutdownMode Enumeration	No	-	`OracleDBShutdownMode` enumerated values specify the database shutdown options.
OracleDBStartupMode Enumeration	No	-	`OracleDBStartupMode` enumerated values specify the database startup options.
OracleDbType Enumeration	-	-	`OracleDbType` enumerated values are used to explicitly specify the `OracleDbType` of an `OracleParameter`.
OracleHAEventSource Enumeration	-	-	The `OracleHAEventSource` enumeration indicates the source of the HA event.
OracleHAEventStatus Enumeration	-	-	The `OracleHAEventStatus` enumeration indicates the status of the HA event source.
OracleIdentityType Enumeration	-	-	The `OracleIdentityType` enumeration specifies how Oracle identity column values are generated.
OracleNotificationInfo Enumeration	-	No	`OracleNotificationInfo` enumerated values specify the database event that causes the notification.
OracleNotificationSource Enumeration	-	No	`OracleNotificationSource` enumerated values specify the different sources that cause notification.
OracleNotificationType Enumeration	-	No	`OracleNotificationType` enumerated values specify the different types that cause the notification.
OracleParameterStatus Enumeration	-	-	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 1-2 (Cont.) Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Enumerations

Enumeration	Supported in the ODP.NET, Managed Driver	Supported in .NET Stored Procedures	Description
OracleRowidInfo Enumeration	-	-	The `OracleRowidInfo` enumeration values specify whether `ROWID` information is included as part of the `ChangeNotificationEventArgs` or not
OracleXmlCommandType Enumeration	-	-	The `OracleXmlCommandType` enumeration specifies the values that are allowed for the `OracleXmlCommandType` property of the `OracleCommand` class.

1.3.4 Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces

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

1.3.4.1 Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Structures

Table 1-3 lists the type structures.

Table 1-3 Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Structures

Structure	Description
OracleBinary Structure	The `OracleBinary` structure represents a variable-length stream of binary data.
OracleBoolean Structure	The `OracleBoolean` structure represents a logical value that is either `TRUE` or `FALSE`.
OracleDate Structure	The `OracleDate` structure represents the Oracle `DATE` data type.
OracleDecimal Structure	The `OracleDecimal` structure represents an Oracle `NUMBER` in the database or any Oracle numeric value.
OracleIntervalDS Structure	The `OracleIntervalDS` structure represents the Oracle `INTERVAL DAY TO SECOND` data type.
OracleIntervalYM Structure	The `OracleIntervalYM` structure represents the Oracle `INTERVAL YEAR TO MONTH` data type.
OracleString Structure	The `OracleString` structure represents a variable-length stream of characters.

Table 1-3 (Cont.) Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Structures

Structure	Description
OracleTimeStamp Structure	The `OracleTimeStamp` structure represents the Oracle `TimeStamp` data type.
OracleTimeStampLTZ Structure	The `OracleTimeStampLTZ` structure represents the Oracle `TIMESTAMP WITH LOCAL TIME ZONE` data type.
OracleTimeStampTZ Structure	The `OracleTimeStampTZ` structure represents the Oracle `TIMESTAMP WITH TIME ZONE` data type.

1.3.4.2 Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Exceptions

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

Table 1-4 Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Exceptions

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

1.3.4.3 Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Classes

Table 1-5 lists the type classes.

Table 1-5 Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Classes

Class	Supported in the ODP.NET, Managed Driver	Description
OracleArrayMappingAttribute Class	No	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.

Table 1-5 (Cont.) Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Classes

Class	Supported in the ODP.NET, Managed Driver	Description
OracleBFile Class	-	An `OracleBFile` is an object that has a reference to `BFILE` data. It provides methods for performing operations on `BFILE` objects.
OracleBlob Class	-	An `OracleBlob` object is an object that has a reference to `BLOB` data. It provides methods for performing operations on `BLOB` objects.
OracleClob Class	-	An `OracleClob` is an object that has a reference to `CLOB` data. It provides methods for performing operations on `CLOB` objects.
OracleCustomTypeMappingAttribute Class	No	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.
OracleObjectMappingAttribute Class	No	The `OracleObjectMappingAttribute` class marks custom class fields or properties with information that ODP.NET uses when a custom type represents an Oracle Object type.
OracleRef Class	No	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`.
OracleRefCursor Class	-	An `OracleRefCursor` object represents an Oracle `REF CURSOR`.
OracleUdt Class	No	The `OracleUdt` class defines static methods that are used when converting between Custom Types and Oracle UDTs and vice-versa.
OracleXmlStream Class	-	An `OracleXmlStream` object represents a sequential read-only stream of XML data stored in an `OracleXmlType` object.
OracleXmlType Class	-	An `OracleXmlType` object represents an Oracle `XmlType` instance.

1.3.4.4 Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Interfaces

Table 1-6 lists the type interfaces.

Table 1-6 Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Interfaces

Interface	Supported in the ODP.NET, Managed Driver	Description
IOracleArrayTypeFactory Interface	No	The `IOracleArrayTypeFactory` interface is used by ODP.NET to create arrays that represent Oracle Collections.

Table 1-6 (Cont.) Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Interfaces

Interface	Supported in the ODP.NET, Managed Driver	Description
IOracleCustomType Interface	No	`IOracleCustomType` is an interface for converting between a Custom Type and an Oracle Object or Collection Type.
IOracleCustomTypeFactory Interface	No	The `IOracleCustomTypeFactory` interface is used by ODP.NET to create custom objects that represent Oracle Objects or Collections.
INullable Interface	-	The `INullable` interface is used to determine whether or not an ODP.NET type has a NULL value.

1.3.4.5 Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Enumerations

Table 1-7 lists the type enumerations.

Table 1-7 Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Enumerations

Enumeration	Supported in the ODP.NET, Managed Driver	Description
OracleUdtFetchOption Enumeration	No	`OracleUdtFetchOption` enumeration values specify how to retrieve a copy of the referenceable object.
OracleUdtStatus Enumeration	No	`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.

1.4 Differences between the ODP.NET Managed Driver and Unmanaged Driver

ODP.NET, Managed Driver and ODP.NET, Unmanaged Driver have a number of configuration setting differences.

Table 1-8 Application Programming Interfaces not supported in ODP.NET, Managed Driver

Namespace	Class/Enumeration/Interface	Unsupported Method/Property/Event
`Oracle.ManagedDataAccess.Client`	`FailoverEvent` enumeration	All
`Oracle.ManagedDataAccess.Client`	`FailoverReturnCode` enumeration	All
`Oracle.ManagedDataAccess.Client`	`FailoverType` enumeration	All
`Oracle.ManagedDataAccess.Client`	`OracleAQAgent` class	All
`Oracle.ManagedDataAccess.Client`	`OracleAQDequeueMode` enumeration	All
`Oracle.ManagedDataAccess.Client`	`OracleAQDequeueOptions` class	All

Table 1-8 (Cont.) Application Programming Interfaces not supported in ODP.NET, Managed Driver

Namespace	Class/Enumeration/Interface	Unsupported Method/Property/Event
`Oracle.ManagedDataAccess.Client`	`OracleAQEnqueueOptions` class	All
`Oracle.ManagedDataAccess.Client`	`OracleAQMessage` class	All
`Oracle.ManagedDataAccess.Client`	`OracleAQMessageAvailableEventArgs` class	All
`Oracle.ManagedDataAccess.Client`	`OracleAQMessageAvailableEventHandler` class	All
`Oracle.ManagedDataAccess.Client`	`OracleAQMessageDeliveryMode` enumeration	All
`Oracle.ManagedDataAccess.Client`	`OracleAQMessageState` enumeration	All
`Oracle.ManagedDataAccess.Client`	`OracleAQMessageType` enumeration	All
`Oracle.ManagedDataAccess.Client`	`OracleAQNavigationMode` enumeration	All
`Oracle.ManagedDataAccess.Client`	`OracleAQNotificationGroupingType` enumeration	All
`Oracle.ManagedDataAccess.Client`	`OracleAQNotificationType` enumeration	All
`Oracle.ManagedDataAccess.Client`	`OracleAQQueue` class	All
`Oracle.ManagedDataAccess.Client`	`OracleAQVisibilityMode` enumeration	All
`Oracle.ManagedDataAccess.Client`	`OracleBulkCopy` class	All
`Oracle.ManagedDataAccess.Client`	`OracleBulkCopyColumnMapping` class	All
`Oracle.ManagedDataAccess.Client`	`OracleBulkCopyColumnMappingCollection` class	All
`Oracle.ManagedDataAccess.Client`	`OracleBulkCopyOptions` class	All
`Oracle.ManagedDataAccess.Client`	`OracleCommand` class	`ArrayBindRowsAffected` property
`Oracle.ManagedDataAccess.Client`	`OracleCommand` class	`ImplicitRefCursors` property
`Oracle.ManagedDataAccess.Client`	`OracleConnection` class	`FlushCache()` method
`Oracle.ManagedDataAccess.Client`	`OracleConnection` class	`Failover` event
`Oracle.ManagedDataAccess.Client`	`OracleConnection` class	`ConnectionType` property
`Oracle.ManagedDataAccess.Client`	`OracleConnection` class	`SetShardingKey` method
`Oracle.ManagedDataAccess.Client`	`OracleConnectionType` enumeration	All
`Oracle.ManagedDataAccess.Client`	`OracleDBShutdownMode` enumeration	All
`Oracle.ManagedDataAccess.Client`	`OracleDBStartupMode` enumeration	All
`Oracle.ManagedDataAccess.Client`	`OracleDataReader` class	`GetOracleRef()` method
`Oracle.ManagedDataAccess.Client`	`OracleDataReader` class	`GetOracleBlobForUpdate()` method If the method is called, then a `NotSupportedException` is thrown.

Table 1-8 (Cont.) Application Programming Interfaces not supported in ODP.NET, Managed Driver

Namespace	Class/Enumeration/Interface	Unsupported Method/Property/Event
`Oracle.ManagedDataAccess.Client`	`OracleDataReader` class	`GetOracleClobForUpdate()` method If the method is called, then a `NotSupportedException` is thrown.
`Oracle.ManagedDataAccess.Client`	`OracleDataReader` class	`IsAutoIncrement` and `IdentityType` properties of the `GetSchemaTable`
`Oracle.ManagedDataAccess.Client`	`OracleDataAdapter` class	`IdentityInsert` property
`Oracle.ManagedDataAccess.Client`	`OracleDataAdapter` class	`IdentityUpdate` property
`Oracle.ManagedDataAccess.Client`	`OracleDataAdapter` class	`SafeMapping` property
`Oracle.ManagedDataAccess.Client`	`OracleDatabase` class	All
`Oracle.ManagedDataAccess.Client`	`OracleDbType` enumeration	`Array`
`Oracle.ManagedDataAccess.Client`	`OracleDbType` enumeration	`Object`
`Oracle.ManagedDataAccess.Client`	`OracleDbType` enumeration	`Ref`
`Oracle.ManagedDataAccess.Client`	`OracleException` class	`IsRecoverable` property
`Oracle.ManagedDataAccess.Client`	`OracleFailoverEventArgs` class	All
`Oracle.ManagedDataAccess.Client`	`OracleFailoverEventHandler` class	All
`Oracle.ManagedDataAccess.Client`	`OracleGlobalization` class	`ClientCharacterSet` property
`Oracle.ManagedDataAccess.Client`	`OracleGlobalization` class	`GetClientInfo()` method
`Oracle.ManagedDataAccess.Client`	`OracleGlobalization` class	`GetThreadInfo()` method
`Oracle.ManagedDataAccess.Client`	`OracleGlobalization` class	`SetThreadInfo()` method
`Oracle.ManagedDataAccess.Client`	`OracleIdentityType` enumeration	All
`Oracle.ManagedDataAccess.Client`	`OracleNotificationRequest` class	`GroupingInterval` property
`Oracle.ManagedDataAccess.Client`	`OracleNotificationRequest` class	`GroupingNotificationEnabled` property
`Oracle.ManagedDataAccess.Client`	`OracleNotificationRequest` class	`GroupingType` property
`Oracle.ManagedDataAccess.Client`	`OracleRowsCopiedEventArgs` class	All
`Oracle.ManagedDataAccess.Client`	`OracleRowsCopiedEventHandler` class	All
`Oracle.ManagedDataAccess.Types`	`IOracleArrayTypeFactory` interface	All
`Oracle.ManagedDataAccess.Types`	`IOracleCustomType` interface	All
`Oracle.ManagedDataAccess.Types`	`IOracleCustomTypeFactory` interface	All
`Oracle.ManagedDataAccess.Types`	`OracleArrayMappingAttribute` class	All
`Oracle.ManagedDataAccess.Types`	`OracleCustomTypeMappingAttribute` class	All
`Oracle.ManagedDataAccess.Types`	`OracleObjectMappingAttribute` class	All
`Oracle.ManagedDataAccess.Types`	`OracleRef` class	All
`Oracle.ManagedDataAccess.Types`	`OracleShardingKey` class	All

Table 1-8 (Cont.) Application Programming Interfaces not supported in ODP.NET, Managed Driver

Namespace	Class/Enumeration/Interface	Unsupported Method/Property/ Event
Oracle.ManagedDataAccess.Types	OracleTimestampTZ structure	OracleTimeStampTZ(DateTime dt, string timeZone) constructor. This constructor is supported but the timeZone must be an hour offset.
Oracle.ManagedDataAccess.Types	OracleUdt class	All
Oracle.ManagedDataAccess.Types	OracleUdtFetchOption enumeration	All
Oracle.ManagedDataAccess.Types	OracleUdtStatus enumeration	All

1.5 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 using ODP.NET, Unmanaged Driver:

```
// C#

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();
  }
}
```

If you are using OPD.NET, Managed Driver, then replace the contents of `Program.cs` with the following C# code. The namespace of ODP.NET, Managed Driver (`Oracle.ManagedDataAccess.*`) is different from the namespace of ODP.NET, Unmanaged Driver (`Oracle.DataAccess.*`)

```
// C#
using System;
using Oracle.ManagedDataAccess.Client;
using Oracle.ManagedDataAccess.Types;

namespace Connect
{
  class Program
  {
    static void Main(string[] args)
    {
      try
      {
        // Please replace the connection string attribute settings
        string constr = "user id=scott;password=tiger;data source=oracle";

        OracleConnection con = new OracleConnection(constr);
        con.Open();
        Console.WriteLine("Connected to Oracle Database {0}", con.ServerVersion);
        con.Dispose();

        Console.WriteLine("Press RETURN to exit.");
        Console.ReadLine();
      }
      catch (Exception ex)
      {
        Console.WriteLine("Error : {0}", ex);
      }
    }
  }
}
```

Note:

Additional samples are provided in the *ORACLE_BASE*\ORACLE_HOME\ODACsamples directory.

2

Installing and Configuring Oracle Data Provider for .NET

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

This section contains these topics:

2.1 System Requirements

Oracle Data Provider for .NET, Unmanaged Driver requires the following:

- Windows operating system
  - 64-bit: Windows 7 x64 (Professional, Enterprise, and Ultimate Editions), Windows 8 (Pro and Enterprise Editions), Windows 8.1 (Pro and Enterprise Editions), Windows Server 2012 x64 (Standard, Datacenter, Essentials, and Foundation Editions), Windows Server 2012 R2 x64 (Standard, Datacenter, Essentials, and Foundation Editions), or Windows 10 x64 (Pro, Enterprise, and Education Editions).

    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 3.5 SP 1 and later.
  - ODP.NET for .NET Framework 4 is only supported with Microsoft .NET Framework 4.5.2, 4.6, 4.6.1, and 4.6.2.
- Access to Oracle Database 10*g* Release 2 or later
- Oracle Client release 12.1

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

Oracle Data Provider for .NET, Managed Driver requires the following:

- Same Windows operating system support as ODP.NET, Unmanaged Driver.

  ODP.NET, Managed Driver is built with AnyCPU. It runs on either 32-bit or 64-bit (x64) Windows and on either 32-bit or 64-bit (x64) .NET Framework.
- Microsoft .NET Framework 4.5.2, 4.6, 4.6.1, or 4.6.2.
- Access to Oracle Database 10*g* Release 2 or later

Possible additional requirements for both ODP.NET, Managed and Unmanaged Drivers:

- Applications using promotable and distributed transactions require Oracle Services for Microsoft Transaction Server 12.1 in whole or in part. ODP.NET only supports the read committed isolation level for distributed transactions. Refer to the Distributed Transactions section for more information.

2.2 Entity Framework Requirements

This section contains the following topics:

- Entity Framework Database First and Model First Requirements
- Entity Framework Code First Requirements

2.2.1 Entity Framework Database First and Model First Requirements

Oracle's support for Entity Framework Database First and Model First has the following version requirements:

- ODP.NET 11.2.0.3 or higher
- Microsoft Entity Framework 4 or higher, up to and including the 6.x versions.

If using Visual Studio tools, then use Visual Studio 2010 or higher and install Oracle Developer Tools for Visual Studio.

2.2.2 Entity Framework Code First Requirements

Oracle's support for Entity Framework Code First has the following version requirements:

- ODP.NET 12.1.0.2 or higher
- Microsoft Entity Framework 6 or higher
- Microsoft .NET Framework 4.5 or higher

  Projects must set the target framework to .NET Framework 4.5 or higher. This can be done by modifying the project's properties in Visual Studio 2012 or higher.

2.3 Oracle Data Provider for .NET Versioning Scheme

Starting with 11.2.0.1.2, Oracle Data Provider for .NET, Unmanaged Driver ships with two sets of binaries: one set for .NET Framework 2.0 and another for .NET Framework 4. ODP.NET, Managed Driver ships with one set of binaries 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 11.2.0.2.0, 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.

ODP.NET, Managed Driver follows a similar version model for its binaries.

ODP.NET for .NET Framework 4:

- `Oracle.ManagedDataAccess.dll`
  - Built with .NET Framework 4
  - Assembly version number: 4.*x*.*x*.*x*
- `Oracle.ManagedDataAccessDTC.dll`
  - Used by ODP.NET for .NET Framework 4 for distributed transactions only.

ODP.NET, Managed Driver Versioning

Starting with ODAC 12c Release 2, the ODP.NET, Managed Driver uses assembly manifest attribute `AssemblyInformationalVersionAttribute` to uniquely identify assemblies with the same `AssemblyVersionAttribute` attribute value. This value can be accessed via .NET code, PowerShell, and other Windows applications to identify ODP.NET, Managed Driver versions uniquely.

`AssemblyInformationalVersionAttribute` is set to the same version as the actual .NET assembly version, except the fourth digit, which will no longer be 0. Instead, the version will be unique for each ODP.NET, Managed Driver release by incrementing the fourth digit for every subsequent release.

This value is accessible using .NET Framework System.Diagnostics.FileVersionInfo.ProductVersion property. The returned value can be used as a Version object or as a comparison String using comparison operators or methods. Essentially, among a collection of ODP.NET, Managed Driver assemblies that have the same assembly version, the newest ODP.NET, Managed Driver assembly will have the largest fourth digit `ProductVersion` value than an older assembly.

PowerShell Example: In this example, administrators uniquely distinguish the assemblies between ODP.NET, Managed Driver versions from an old version of ODP.NET, Managed Driver in `c:\old` and a more recent one in `c:\new`.

Script:

```
$VC1 = New-Object System.Version((Get-Command C:\old
\Oracle.ManagedDataAccess.dll).FileVersionInfo.ProductVersion)
$VC2 = New-Object System.Version((Get-Command C:\new
\Oracle.ManagedDataAccess.dll).FileVersionInfo.ProductVersion)
"Compare V1 to V2: " + $VC1.CompareTo($VC2)
"Compare V1 to V1: " + $VC1.CompareTo($VC1)
"Compare V2 to V1: " + $VC2.CompareTo($VC1)
```

Output:

```
Compare V1 to V2: -1
Compare V1 to V1: 0
Compare V2 to V1: 1
```

> **Note:**
>
> `ProductVersion` property comparisons will provide correct information on which version is more recent than the other *only* for ODP.NET, Managed Driver released from ODAC 12*c* Release 2 and later.

2.4 Installing Oracle Data Provider for .NET, Unmanaged Driver

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 the `OracleClientFactory` class. This enables the `DbProviderFactories` class to recognize ODP.NET.

ODP.NET, Unmanaged Driver Entity Framework 6 and Code First functionality are available through a NuGet package. OUI and Xcopy installations include this package as well, but require post-install configuration steps. The NuGet package for ODP.NET, Unmanaged Driver Entity Framework automates these post-install steps, except for the application-specific connection string settings.

2.4.1 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 can be accessed through *`ORACLE_BASE`*`\`*`ORACLE_HOME`*`\ODACDoc\DocumentationLibrary\doc\index.htm`.

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

2.4.2 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.

Note:

Both `Oracle.DataAccess.dll` for .NET Framework 2.0 and `Oracle.DataAccess.dll` for .NET Framework 4 use the same unmanaged DLL, `OraOps12.dll`.

2.4.2.1 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.

2.5 Installing Oracle Data Provider for .NET, Managed Driver

Getting started with ODP.NET, Managed Driver

You can get started with ODP.NET Managed Driver by either using the Oracle Universal Installer (OUI), XCopy, or NuGet.

If you are using OUI: Follow the Oracle Universal Installer (OUI) steps to install ODP.NET, Managed Driver

If you are using XCopy: Download ODP.NET, Managed Driver `.zip` file to a directory for staging the install. The `.zip` file contains a README file with XCopy installation instructions.

Run the `configure.bat` script in one of the following directories:

- For 32-bit .NET Framework: `OH\odp.net\managed\x86`
- For 64-bit .NET Framework: `OH\odp.net\managed\x64`

Each directory contains an `unconfigure.bat` if ODP.NET, Managed Driver needs to be unconfigured and removed from the machine.

If you are using NuGet: Download the ODP.NET NuGet package(s) and use NuGet Package Manager to install.

The following NuGet packages are available:

- ODP.NET, Managed Driver
- Entity Framework assembly for Code First and Entity Framework 6 or higher use with ODP.NET, Managed Driver

If you are using Windows Installer: Follow the Microsoft Windows Installer (MSI) steps to install ODP.NET, Managed Driver.

ODP.NET, Managed Driver Files

ODP.NET, Managed Driver consists of the following files:

Table 2-1 ODP.NET, Managed Driver Files with Descriptions

File	Description
`Oracle.ManagedDataAccess.dll`	Platform-independent (AnyCPU), fully-managed ADO.NET provider
`\x64\Oracle.ManagedDataAccessDTC.dll`	Platform-dependent (64-bit .NET Framework only), Managed Assembly for Distributed Transaction support.
`\x86\Oracle.ManagedDataAccessDTC.dll`	Platform-dependent (32-bit .NET Framework only), Managed Assembly for Distributed Transaction support.
`\Resources\<lang>\Oracle.ManagedDataAccess.resources.dll`	Platform-independent (AnyCPU), fully-managed ADO.NET provider resource DLLs.
`OraProvCfg.exe`	Platform-independent (AnyCPU) utility to configure/unconfigure ODP.NET, Managed and Unmanaged Drivers.
`configure.bat`	Batch file to place ODP.NET, Managed Driver into the GAC and add configuration entries into the machine.config.
`unconfigure.bat`	Batch file to remove ODP.NET, Managed Driver from the GAC and remove configuration entries from machine.config.
`tnsnames.ora`	A sample configuration file that defines data source aliases.
`sqlnet.ora`	A sample configuration file that configures network related settings.
`ConfigSchema.xsd`	An XML schema file that defines the configuration section for ODP.NET, Managed Driver.
`Oracle.ManagedDataAccess.EntityFramework.dll`	Platform-independent (AnyCPU), fully-managed assembly for Code First and Entity Framework 6 higher
`\x64\Oracle.ManagedDataAccessIOP.dll`	Platform-dependent (64-bit .NET Framework), Managed Assembly for Kerberos support
`\x86\Oracle.ManagedDataAccessIOP.dll`	Platform-dependent (32-bit .NET Framework), Managed Assembly for Kerberos support

- `Oracle.ManagedDataAccessDTC.dll` is only needed if the application uses distributed transactions and the .NET Framework version is 4.5.1 or earlier. Higher .NET Framework versions do not require this DLL.
- If distributed transactions are used by ODP.NET, Managed Driver running in .NET Framework 4.5.1 or earlier, then the appropriate `Oracle.ManagedDataAccessDTC.dll` (32-bit or 64-bit .NET Framework) must be loaded in the Global Assembly Cache (GAC) or in the same directory as the `.exe` for it to be loaded by `Oracle.ManagedDataAccess.dll`. The installer no longer GACs this DLL. It must now be performed manually.

- `Oracle.ManagedDataAccessDTC.dll` must not be referenced by the application. ODP.NET, Managed Driver will reference it implicitly.
- On a 64-bit OS, only the x64 version of `Oracle.ManagedDataAccessDTC.dll` is placed into the GAC upon the completion of an OUI install or an invocation of the XCopy `configure.bat`.

2.5.1 Platform-Dependent Assemblies and Their Search Order

ODP.NET, Managed Driver has two sets of platform-dependent DLLs: `Oracle.ManagedDataAccessDTC.dll` and `Oracle.ManagedDataAccessIOP.dll`. For each DLL, there is a 32-bit .NET version and a 64-bit .NET version. While they consist of 100% managed code, they call APIs outside of .NET, which is why they are platform dependent.

`Oracle.ManagedDataAccessDTC.dll` supports coordinating distributed transactions. This assembly is only needed in your application if you use distributed transactions with .NET Framework 4.5.1 or lower. It is optional to use with .NET Framework 4.5.2 or higher.

`Oracle.ManagedDataAccessIOP.dll` supports Kerberos. This assembly is only needed in your application if you are using Kerberos security.

These two assemblies are not intended to be directly referenced by an application. Rather, they will be referenced implicitly. ODP.NET, Managed Driver will reference these assemblies by using the following search order:

1. Global Assembly Cache
2. The web application's bin directory or Windows application's `EXE` directory
3. The `x86` or `x64` subdirectory based on whether the application runs in 32-bit or 64-bit .NET Framework. If the application is built using AnyCPU, then ODP.NET will use the correct DLL bitness as long as the assembly is available. Oracle recommends using this method of finding dependent assemblies if your application is AnyCPU.

For example, use the following steps for your application to use the 64-bit version of `Oracle.ManagedDataAccessIOP.dll`:

1. Right click **Visual Studio project**, select **Add**, and then select **New Folder**.
2. Name the folder x64.
3. Right-click the newly created **x64** folder, select **Add**, and then select **Existing Item**.
4. Browse to the folder where the DLL is located, which usually is *`ORACLE_HOME`*`\odp.net\managed\x64`, and then select **Oracle.ManagedDataAccessIOP.dll**.
5. Click **Add**.
6. Click the newly added **Oracle.ManagedDataAccessIOP.dll** in the x64 folder.
7. In the properties window, set Copy To Output Director to **Copy Always**.

For x86 targeted applications, name the folder x86 and add the assembly from the x86 directory.

Use the same steps for adding `Oracle.ManagedDataAccessDTC.dll`.

To make your application platform independent even if it depends on `Oracle.ManagedDataAccessDTC.dll`, `Oracle.ManagedDataAccessIOP.dll` or both, create both x64 and x86 folders with the necessary assemblies added to them.

2.5.2 File Locations After Installation

In an Oracle Universal Installer based install, the `Oracle.ManagedDataAccess.dll` assembly is installed to the following location:

.NET Framework 4:

ORACLE_BASE\ORACLE_HOME\odp.net\managed\common directory

Documentation and the `readme.txt` file can be accessed through *ORACLE_BASE*\ORACLE_HOME\ODACDoc\DocumentationLibrary\doc\index.htm.

Samples are provided in the *ORACLE_BASE*\ORACLE_HOME\ODACsamples directory.

2.6 Entity Framework Code First Assemblies and File Location

ODP.NET now ships with a separate assembly to support Code First and Entity Framework 6. This functionality resides in a dedicated assembly, while the ADO.NET and earlier Entity Framework version functionality resides in the main ODP.NET assembly. This model physically separates Entity Framework 6 functionality from ADO.NET functionality.

This ODP.NET assembly is:

- `Oracle.DataAccess.EntityFramework.dll` for ODP.NET, Unmanaged Driver.
- `Oracle.ManagedDataAccess.EntityFramework.dll` for ODP.NET, Managed Driver.

Whether it is installed using the Oracle Universal Installer or the XCopy package, the Oracle Entity Framework assemblies may be found in the following location after install (where `%ORACLE_HOME%` represents the operating system path to the installation directory):

For Unmanaged Driver:

```
%ORACLE_HOME%\odp.net\bin\4\EF6\Oracle.DataAccess.EntityFramework.dll
```

For Managed Driver:

```
%ORACLE_HOME%\odp.net\managed\common
\EF6\Oracle.ManagedDataAccess.EntityFramework.dll
```

Both assemblies are compiled as Any CPU and therefore there is no need for separate 32-bit and 64-bit versions of the assemblies. Each assembly is designed to be `bin deployable` meaning that the assembly should be copied into the application's `bin` directory. As such the assemblies are not registered in the Global Assembly Cache (GAC) during installation.

Note:

If desired the Oracle Entity Framework 6 assemblies may be registered in the GAC manually but Oracle recommends not doing so.

2.7 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.

  Windows registry based configuration is not supported for ODP.NET, Managed Driver.
- 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.

  Note:

  There is one exception to `app/web/config` settings overriding `machine.config`. For `oracle.manageddataaccess.client` and `oracle.unmanageddataaccess.client` sections, a `machine.config` with a specific ODP.NET version subsection, that is, `<version number="4.121.2.0">`, will override an `app/web.config` subsection that references all versions generically, that is, `<version number="*">`. To override the `machine.config` subsection, create a subsection for that version in the `app/web/config` file, that is, `<version number="4.121.2.0">`.
- 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 granularities.

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.

All boolean attributes in ODP.NET .NET configuration settings accept `true`, `false`, `1`, and `0` as valid values. `1` is equivalent to `true` and `0` is equivalent to `false`.

2.7.1 Oracle Client Configuration File Automated Setup During Installation

When installing Oracle Data Access Components (ODAC) in a new Oracle Home, Oracle Universal Installer (OUI) automatically copies the Oracle local naming (tnsnames.ora), profile (sqlnet.ora), and directory (ldap.ora) parameter files and settings from an existing Oracle home into the newly installed ODAC home, as long as they share the same bitness. That is, they are both 32-bit installations or they are both 64-bit installations.

Alternatively, existing `*.ora` files can be copied over from another existing Oracle home, besides the last active one, to the new ODAC Oracle home. OUI provides location information for these files from up to three other existing Oracle homes if they exist. The `*.ora` files can be customized if the new Oracle home uses a different configuration from the previous Oracle home from which the files were copied over.

If you install into an existing ODAC or RDBMS Oracle home, then no new `*.ora` files is copied or created.

If you install onto a computer without any previous Oracle homes present, then OUI prompts the user for the database connection alias information. OUI then automatically creates the `tnsnames.ora` file. If no alias information is provided, then no `tnsnames.ora` file is created. Even if the user does not have all the database connection information readily available, Oracle recommends inserting placeholder values during the install process, then modifying the `tnsnames.ora` file later with actual values to replace the placeholders.

2.7.2 Oracle Client Configuration File Settings

ODP.NET `tnsnames.ora`, `sqlnet.ora`, and `ldap.ora` parameter values can be set in a .NET configuration file or within the `*.ora` file itself. The `*.ora` file location can be a location different from the standard *`ORACLE_HOME`*`/network/admin` directory. The `*.ora` settings order of precedence is similar to ODP.NET's settings order of precedence. The main difference is that the `*.ora` files themselves are included in the search order. The `tnsnames.ora` and `sqlnet.ora` precedence order is as follows:

1. `app.config` or `web.config`
2. `machine.config`
3. File location specified by `TNS_ADMIN` setting
4. The current `.EXE` or web application root directory
5. `%ORACLE_HOME%\network\admin` if using ODP.NET, Unmanaged Driver

The `ldap.ora` precedence order is as follows:

1. `app.config` or `web.config`
2. `machine.config`

3. File location specified by `TNS_ADMIN` setting in .NET config file
4. File location specified by `LDAP_ADMIN` setting in .NET config file
5. The current `.EXE` or web application root directory
6. `%ORACLE_HOME%\network\admin` if using ODP.NET, Unmanaged Driver
7. `%ORACLE_HOME%\ldap\admin` if using ODP.NET, Unmanaged Driver

Oracle recommends using an `app.config` or `web.config` file to store all these Oracle Client configuration parameter settings.

Once the first `tnsnames.ora`, `sqlnet.ora`, and `ldap.ora` are found and read, no additional `*.ora` file lower in the precedence order is read. That means all Oracle Client configuration settings must be made in the `app.config`, `web.config`, `machine.config`, or the first set of `*.ora` files found. Additional parameter values set in `*.ora` files lower in the precedence order will not be read.

2.7.3 Machine-Wide Configuration Option

ODAC OUI and xcopy installs ODP.NET with either machine-wide or non-machine-wide configuration for managed and unmanaged ODP.NET. Machine-wide configuration makes global changes to the machine's .NET setup, including placing the provider assembly into the Global Assembly Cache (GAC) and updating the `machine.config` with configuration section handler and `DbProviderFactory` information.

Machine-wide configuration also creates a `TNS_ADMIN` `machine.config` setting. If `TNS_ADMIN` already exists as a Windows environment variable in an OUI ODAC installation, then the `TNS_ADMIN` `machine.config` setting is set to that directory location. If `TNS_ADMIN` does not already exist for an OUI ODAC installation, then the `machine.config` `TNS_ADMIN` value is set to *`ORACLE_HOME`*`\network\admin`. Xcopy installations always create a `machine.config` `TNS_ADMIN` value set to ORACLE_HOME`\network\admin`.

For ODAC OUI machine-wide configuration installations only, the `LDAP_ADMIN` setting may also be created in `machine.config` if an `ldap.ora` file can be found through the existing `LDAP_ADMIN` or `TNS_ADMIN` Windows environment variables. ODAC OUI installations may also create a `NAMES.DIRECTORY_PATH` setting in `machine.config` for machine-wide configuration.

If non-machine-wide configuration is selected, then none of these changes are made. Starting with release 12.2, ODAC installs default to non-machine-wide configuration for a new Oracle home installation. For existing Oracle homes, ODAC re-installs the default to the same configuration setting chosen for that Oracle home from the previous installation.

If you plan to install ODAC and the ODP.NET NuGet install on the same machine, then ODP.NET should be configured for non-machine-wide, especially if both share the same ODP.NET version number that .NET Framework uses to distinguish assembly versions, for example, 4.121.2.0.

Users can reconfigure ODP.NET from machine-wide configuration to non-machine-wide configuration by re-installing ODP.NET to the same Oracle home where ODP.NET of the same version is already installed. For example, if you have already configured ODP.NET machine-wide, then you can re-configure it by re-installing ODP.NET onto the same Oracle home *and* selecting the non-machine-wide configuration option.

For applications that depend on an ODP.NET version that was not configured machine-wide, it is important to note the following:

- ODP.NET assembly or assemblies that the application depends on will need to be copied over to the application directory.
- Proper .NET configuration settings will be required to use Provider Factory, or Provider-specific configuration, or both.

2.8 Oracle Data Provider for .NET, Unmanaged Driver Configuration

The following sections explain how to configure ODP.NET, Unmanaged Driver.

ODP.NET can be configured using an XML file named `web.config`, `app.config`, or `machine.config`. These config files contain sections specific to ODP.NET configuration.

For unmanaged ODP.NET, developers use either the traditional `<oracle.dataaccess.client>` section or the newer `<oracle.unmanageddataaccess.client>` section. Oracle recommends applications use `<oracle.unmanageddataaccess.client>` when possible. For managed ODP.NET, developers use `<oracle.manageddataaccess.client>`.

`<oracle.unmanageddataaccess.client>` is a superset of `<oracle.manageddataaccess.client>` as unmanaged ODP.NET supports some features not available in the managed driver. For features both providers have in common, they share the same structure, properties, and nearly all values. Programmers will find using either provider interchangeably or migrating between unmanaged and managed ODP.NET is easier with the shared format.

This documentation section covers unmanaged ODP.NET configuration settings in the Windows registry, `<oracle.dataaccess.client>`, or unique `<oracle.unmanageddataaccess.client>` settings. For shared settings with `<oracle.manageddataaccess.client>`.

2.8.1 Supported Configuration Settings

ODP.NET, Unmanaged Driver 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 2-2 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 2-2 Configuration Attributes

Attribute/Setting Name	Description
`CheckConStatus`	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. Configuration Support: Windows Registry and XML file Valid Values: `1`: Check the status of the connection. `0`: Do not check the status of the connection. Default: `1`
`DbNotificationPort`	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 `OracleDependency.Port` static field. Configuration Support: XML file, and ODP.NET class Valid Values: `-1`: Open a random unused port to listen to. *n* > = `0`: Listen on port *n*. Default: `-1`
`DemandOraclePermission`	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

Table 2-2 (Cont.) Configuration Attributes

Attribute/Setting Name	Description
`DllPath`	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: *`ORACLE_BASE`*`\\`*`ORACLE_HOME`*`\bin`
`DynamicEnlistment`	Due to a behavior change with the ODAC 12c Release 3 version of ODP.NET connection string attribute `enlist=dynamic`, `DynamicEnlistment` has no operation now.
`FetchSize`	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` <= *n* <= `int.MaxValue`: *n* is the size of the cache in bytes. Default: `131072`

`LegacyEntireLobFetch`	Returns either `OracleBlob` and `OracleClob` types or `OracleBinary` and `OracleString` types from Oracle Database `BLOB` and `CLOB` columns. This setting only applies when `InitialLobFetchSize` is set to `-1`.
	Valid Values:
	`0`: Returns `OracleBlob` and `OracleClob`
	`1`: Returns `OracleBinary` and `OracleString`
	Default: `0`

Table 2-2 (Cont.) Configuration Attributes

Attribute/Setting Name	Description
`LegacyTransactionBindingBehavior`	Specifies when a database connection detaches from a `System.Transactions` transaction. By default, connections detach from a transaction only when explicitly unbound as is the case when the connection closes or implicitly unbound when the transaction is disposed. Alternatively, this attribute can be set so that the connection detaches whenever the transaction ends (commits, aborts, or times out), the connection closes, or the transaction is disposed. In ODP.NET 11.2.0.3.20 and earlier releases, the latter was the default behavior. Oracle recommends using the current default behavior. In the earlier default behavior, when the timeout elapses before the transaction completes, the connection unbinds itself from the transaction and all subsequent executions on this connection execute in `AutoCommit` mode. Any operations prior to the timeout roll back, but operations performed after the timeout commit. In the current default setting, users receive an exception when the transaction times out and additional operations execute on the connection. Configuration Support: Windows Registry and XML file Valid Values: `0`: Connections detach from transaction when the connection closes or the transaction is disposed. `1`: Connections detach from transaction when the connection closes, the transaction is disposed, or the transaction completes (commits, rolls back, times out). Default: `0`
`MaxStatementCacheSize`	Specifies the maximum number of statements that can be cached when self-tuning is enabled. Configuration Support: Windows Registry and XML file Valid Values: 0 to `System.Int32.MaxValue`. Default: `100`

Table 2-2 (Cont.) Configuration Attributes

Attribute/Setting Name	Description
`MetaDataXml`	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. 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`

Table 2-2 (Cont.) Configuration Attributes

Attribute/Setting Name	Description
`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` *This property has been deprecated in 12.2.0.1. It will be desupported in a future release.*
`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`

Table 2-2 (Cont.) Configuration Attributes

Attribute/Setting Name	Description
`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.

Table 2-2 (Cont.) Configuration Attributes

Attribute/Setting Name	Description
`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.

Table 2-2 (Cont.) Configuration Attributes

Attribute/Setting Name	Description
`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`
`UDT Mapping`	Specifies a mapping between a custom type and an Oracle UDT in the database. The mappings can be specified in configuration files and custom type factories. However, if the mapping is specified in both places, mappings specified in the configuration files takes precedence over mappings specified using custom type factories. Configuration Support: XML file and Custom Type Factory Classes Valid Values: Any valid mapping. Default: *none*

2.8.2 Windows Registry

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 3.5, and one key for .NET Framework 4 and later.

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\`.

2.8.3 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 later:

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <oracle.dataaccess.client>
    <settings>
      <add name="DllPath"              value="C:\oracle\bin"/>
      <add name="FetchSize"            value="131072"/>
      <add name="StatementCacheSize"   value="10"/>
      <add name="TraceFileName"        value="D:\odpnet2.trc"/>
      <add name="TraceLevel"           value="63"/>
      <add name="TraceOption"          value="1"/>
    </settings>
  </oracle.dataaccess.client>
</configuration>
```

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

```
<?xml version="1.0" encoding="utf-8" ?>
<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="StatementCacheSize" value="50"/>
     <add name="ThreadPoolMaxSize" value="30"/>
     <add name="TraceFileName" value="D:\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>
```

ODP.NET, Unmanaged Driver now has the option of using the same configuration file format as ODP.NET, Managed Driver. The format simplifies configuration by using a single unified scheme. To utilize this format, the existing unmanaged ODP.NET configuration section should be renamed from `<oracle.dataaccess.client>` to `<oracle.unmanageddataaccess.client>`. The existing unmanaged ODP.NET elements and values are supported within the new section using the same format as with ODP.NET, Managed Driver. To see how to set the elements and values, see "Oracle Data Provider for .NET, Managed Driver Configuration" for more information.

For example, converting the `FetchSize` element and value from the traditional to the new format would be done as follows:

```
<oracle.dataaccess.client>
   <settings>
     <add name="FetchSize" value="131072" />
   </settings>
</oracle.dataaccess.client>

<oracle.unmanageddataaccess.client>
   <version number="*">
     <settings>
       <setting name="FetchSize" value="131072" />
     </settings>
   </version>
</oracle.unmanageddataaccess.client>
```

The traditional ODP.NET, Unmanaged Driver configuration file format will continue to be supported.

2.8.3.1 SQL Translation Framework Configuration

Configuring the SQL Translation Profile

The default SQL Translation Profile can be set in the .NET config file, either for all connections across the application, or it is also possible to limit the scope of a profile based on optional `dataSource` and `userId` XML attributes. Please note that these `dataSource` and `userId` XML attributes directly correspond to the `Data Source` and `User Id` attributes in the connection string used to open a database connection.

Note:

SQL Translation Profile settings are only supported in the `<oracle.unmanageddataaccess.client>` section. It is not supported in the `<oracle.dataaccess.client>` section nor the `<oracle.manageddataaccess.client>` section.

This would be used for all connections to the `Data Sources` and `User Ids`.

This would be used for all connections to the specified `Data Source`.

This would be used for all connections to the specified `User Id`.

This would be used for all connections to the specified `Data Source` and `User Id`.

It is possible to configure multiple default profile entries which allow configuring default profiles for different `dataSource` and `userId` attributes, but while selecting a profile, the profile with maximum matching attributes will be selected.

In case there are 2 matching entries, one with `dataSource` only and the other with `userId` only then the entry with matching the `userId` would be given priority over the entry with matching `dataSource`.

With the above configuration, if we try to connect with a connection string which has `stf_ds` for `Data Source` and `stf_user` for `User Id` attributes, then both the entries given above will match and in such cases, we will give priority to the entry with a matching `User Id` attribute which means `profile_user` will be selected as the default profile.

Configuring the Error Mapping

Applications can configure the connection related error mapping in their application configuration file. The error mapping can also be scoped based on `Data Source` name, `User Id` and the profile name itself.

Here is an example of providing error mapping with all three attributes.

```
<configuration>
  <oracle.unmanageddataaccess.client>
   <version number="*">
   <sqlTranslation>
     <defaultProfiles>
      <defaultProfile dataSource="stf_ds" userId="stf_user" profile=" Profile4"/>
     </defaultProfiles>
     <ErrorMappings>
        <ErrorMapping dataSource="stf_ds" userId="stf_user" profile="Profile4">
          <add oracleErrorNumber="1017" translatedErrorCode="222" />
          <add oracleErrorNumber="1005" translatedErrorCode="888" />
        </ErrorMapping>
     </ErrorMappings>
    </sqlTranslation>
   </version>
  </oracle.unmanageddataaccess.client>
</configuration>
```

Please note that `dataSource` and `userId` attributes are optional but can be used to scope the mapping.

It is also possible to provide an error mapping which could be used for all profiles. Here is an example:

```
<ErrorMappings>
  <ErrorMapping profile="*">
    <add oracleErrorNumber="1017" translatedErrorCode="222" />
    <add oracleErrorNumber="1018" translatedErrorCode="888" />
  </ErrorMapping>
</ErrorMappings>
```

Configuring the Default Error Mapping Profile

The default error mapping profile can be configured through the `defaultErrorMappingProfile` setting. This is to be used to specify the default error mapping profile, especially in scenarios when the default profile is not specified through the .NET configuration file, but specified on the server side. In this case, if connectivity related errors occur, then ODP.NET will be able to properly use error mappings specified in the .NET configuration file for the profile specified by the `defaultErrorMappingProfile` setting.

Here is an example to configure the default error mapping profile:

```
<sqlTranslation>
  <settings>
    <add name="defaultErrorMappingProfile" value="error_mapping_profile" />
  <settings>
</sqlTranslation>
```

Configuring the SQL Translation Framework Statement Cache Size

Client can configure the number of translated statements that ODP.NET can cache internally to avoid translations, which can be an expensive operation.

Here is an example to configure default error mapping profile:

```
<sqlTranslation>
    <settings>
      <add name="translatedStatementCacheSize" value="50" />
      <settings>
</sqlTranslation>
```

Sample SQL Translation Framework configuration file

Here is a sample configuration file with all possible elements that can be used:

```
<sqlTranslation>
  <settings>
    <add name="translatedStatementCacheSize" value="50" />
    <add name="defaultErrorMappingProfile" value="def_Profile" />
  <settings>
  <defaultProfiles>
    <defaultProfile profile="STF.NO_DS_NO_USERID"/>
    <defaultProfile userId="stf" profile="STF_NO_DS"/>
    <defaultProfile dataSource="stf_inst" profile="STF_NO_USERID"/>
    <defaultProfile dataSource="stf_inst" userId="stf" profile="STF.STF_X"/>
  </defaultProfiles>
  <ErrorMappings>
    <ErrorMapping profile="def_profile">
      <add oracleErrorNumber="1017" translatedErrorCode="444" />
    </ErrorMapping>
    <ErrorMapping dataSource="stf_inst" userId="stf" profile=" STF.STF_X ">
      <add oracleErrorNumber="1018" translatedErrorCode="88888" />
    </ErrorMapping>
  </ErrorMappings>
</sqlTranslation>
```

Example 2-1 Setting the profile which could be used for all connections

```
<configuration>
  <oracle.unmanageddataaccess.client>
```

```
   <version number="*">
    <sqlTranslation>
     <defaultProfiles>
       <defaultProfile profile="Profile1"/>
      </defaultProfiles>
     </sqlTranslation>
   </version>
  </oracle.unmanageddataaccess.client>
</configuration>
```

Example 2-2   Setting the Profile for a Specific Data Source

```
<defaultProfiles>
  <defaultProfile dataSource="stf_ds" profile="Profile2"/>
</defaultProfiles>
```

Example 2-3   Setting the Profile for a Specific User Id

```
<defaultProfiles>
  <defaultProfile userId="stf_user" profile="Profile3"/>
</defaultProfiles>
```

Example 2-4   Setting the Profile for a Specific Data Source and User Id'

```
<defaultProfiles>
  <defaultProfile dataSource="stf_ds" userId="stf_user" profile="Profile4"/>
</defaultProfiles>
```

Example 2-5   Configuring Multiple Default Profile Entries

```
<defaultProfiles>
  <defaultProfile dataSource="stf_ds" profile="profile_ds"/>
  <defaultProfile userId="stf_user" profile="profile_user"/>
</defaultProfiles>
```

2.8.3.2 Specifying UDT Mappings with Unified Configuration for Unmanaged ODP.NET

As UDT mapping is not currently supported by ODP.NET, Managed Driver, a new section within the `<version>` section is used to support custom UDT mappings for unmanaged ODP.NET in the unified configuration format. This new section is identified as `<udtmappings>` and each mapping is identified using a `<udtmapping>` element. The following attributes may be specified for each `udtMapping` element:

- `typeName` (required)
- `factoryName` (required)
- `dataSource` (optional)
- `schemaName` (optional)

These elements retain the same name and meaning as when used with the traditional configuration format.

Example of converting traditional format to unified format:

```
<configuration>
   <oracle.dataaccess.client>
     <settings>
       <add name="Person" value="udtMapping factoryName='PersonFactory, Sample,
Version=0.0.0.0, Culture=neutral, PublicKeyToken=null' typeName='PERSON'
```

```
schemaName='SCOTT' dataSource='oracle'" />
     </settings>
   </oracle.dataaccess.client>
</configuration>

<configuration>
   <oracle.unmanageddataaccess.client>
     <udtmappings>
       <udtmapping typename="PERSON" factoryname="PersonFactory, Sample,
Version=0.0.0.0, Culture=neutral, PublicKeyToken=null" schemaname="SCOTT"
datasource="oracle" />
     </udtmappings>
   </oracle.unmanageddataaccess.client>
</configuration>
```

2.9 Oracle Data Provider for .NET, Managed Driver Configuration

ODP.NET, Managed Driver supports .NET configuration file-based settings in `machine.config`, `app.config`, and `web.config`. It does not support Windows registry based configuration. ODP.NET, Managed Driver settings in .NET configuration files are similar to ODP.NET, Unmanaged Driver settings to make porting easier.

The ODP.NET, Managed Driver configuration file section name is `<oracle.manageddataaccess.client>`. The `<oracle.manageddataaccess.client>` settings and values are also supported in unmanaged ODP.NET configuration file: `<oracle.unmanageddataaccess.client>`. While this documentation section discusses managed ODP.NET configuration, it is also applicable to `<oracle.unmanageddataaccess.client>`. The `<oracle.unmanageddataaccess.client>` settings are actually a superset of `<oracle.manageddataaccess.client>`. The `<oracle.unmanageddataaccess.client>` settings not available in managed ODP.NET are documented in "Oracle Data Provider for .NET, Unmanaged Driver Configuration". A typical .NET config that uses ODP.NET, Managed Driver has some or all of the following subsections nested within a `<version>` subsection under `<oracle.manageddataaccess.client>` section. Note the tag names are case sensitive, while the attribute names are case insensitive.

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <oracle.manageddataaccess.client>
    <version number="*">
      <dataSources>
        ...
        ...
      </dataSources>
      <settings>
        ...
        ...
      </settings>
      <LDAPsettings>
        ...
        ...
      </LDAPsettings>
      <implicitRefCursor>
        ...
        ...
      </implicitRefCursor>
      <edmMappings>
        ...
        ...
      <edmMappings>
    </version>
    <version number="4.121.2.0">
      <dataSources>
        ...
        ...
      </dataSources>
      <settings>
        ...
        ...
      </settings>
      <LDAPsettings>
        ...
        ...
      </LDAPsettings>
      <implicitRefCursor>
        ...
        ...
      </implicitRefCursor>
      <edmMappings>
        ...
        ...
      <edmMappings>
    </version>
  </oracle.manageddataaccess.client>
</configuration>
```

The ODP.NET, Managed Driver configuration and settings are described in the following sections. Many of the attributes are the same as ODP.NET, Unmanaged Driver. See Table 2-2 for detailed attribute descriptions.

2.9.1 version Section

All the information required by an application should be grouped under the `version` subsections. Each `<version number="X">` section contains parameters applicable for version `X` of the ODP.NET, Managed Driver. For example, `<version number="4.121.2.0">` section parameters will be applicable only for those applications using ODP.NET, Managed Driver assembly 4.121.2.0.

Apart from version specific sections, there can also be a generic section `<version number="*">`. This section's parameters are applicable for all ODP.NET, Managed Driver versions. Parameters in the version specific section take precedence over the parameters of the generic section. The following is an example of a `version` section:

```
<oracle.manageddataaccess.client>
  <version number="*">
     <settings>
          <setting name="TraceOption" value="1"/>
          <setting name="PerformanceCounters" value="0" />
     </settings>
  </version>
  <version number="4.121.2.0">
     <settings>
          <setting name="PerformanceCounters" value="4095" />
     </settings>
  </version>
</oracle.manageddataaccess.client>
```

An application referencing ODP.NET, Managed Driver 4.121.2.0 has the following values set:

- `TraceOption = 1`
- `PerformanceCounters= 4095`

2.9.2 dataSources Section

This section can appear only under a `<version>` section. The mapping between the different data source aliases and corresponding data descriptors should appear in this section. The following is an example.

```
<dataSources>
  <dataSource alias="inst1" descriptor="(DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)
(HOST=sales-server)......)))"/>
  <dataSource alias="inst2" descriptor="(DESCRIPTION= ......)))"/>
</dataSources>
```

Note:

The `data source` connection string attribute can alternatively be set to a full descriptor or Easy Connect syntax rather than a data source alias.

Requirements for connecting to a local database *without* specifying "data source" connection string attribute:

- The listener must be up and running.
- `ORACLE_SID` environment variable must be set appropriately.

Note:

When "data source" connection string attribute is not specified, protocol defaults to 'tcp' and port defaults to '1521'.

The ODP.NET managed driver reads and caches all the alias entries from the `app.config`, `web.config`, `machine.config`, and from a `tnsnames.ora` file that is found at application start-up time. However, aliases that are defined in LDAP servers are resolved and cached on demand. This means for each unique alias that is used by the application, an alias resolution query is executed against an LDAP server and the full descriptor associated with the alias will be cached once it is fetched.

For developers that need to change or add alias settings while developing applications, one may consider using `OracleDataSourceEnumerator.GetDataSources()` rather than restarting the application. Invoking this method will first wipe out existing cache entries that were read from the `tnsnames.ora` file and all aliases obtained from the LDAP Server. Then, the `tnsnames.ora` is re-parsed and all its entries will be cached again. Please note that the `app.config`, `web.config`, and `machine.config` entries are read only once at application start-up time and thus their contents are maintained and not re-parsed even if `OracleDataSourceEnumerator.GetDataSources()` is invoked.

The `OracleDataSourceEnumerator.GetDataSources()` method invocation has an impact on the connection pool. This is because a connection pool, which is created for each unique connection string, will cache the resolved full descriptor information after the first connection is created for a given connection pool. After that, the connection pool uses the cached full descriptor information for all subsequent connection creations. Thus, for applications that have their `tnsnames.ora` or LDAP entries modified during the execution of an application where an alias points to a different database than before, one should call the `OracleDataSourceEnumerator.GetDataSources()` method to remove old cached entries. This should be followed by the invocation of the `ClearPool(OracleConnection)` instance method or the `ClearAllPools()` static method to remove existing connections and also have it obtain a new full descriptor value that was read by the invocation of `OracleDataSourceEnumerator.GetDataSources()`. Following this scheme will assure that *all* the connections in the connection pool uses the new full descriptor that is now associated with the alias and all connections in a connection pool is established to the same database.

The following keywords are supported within the descriptor setting:

- `ADDRESS`
- `ADDRESS_LIST` (Note: only failover supported)

  Oracle recommends using SCAN listener and Runtime Load Balancing to balance the load when connecting to an Oracle RAC database.
- `DESCRIPTION`
- `DESCRIPTION_LIST` (Note: Failover supported; `Address_list` load balancing not supported)
- `HOST` (Note: <hostname>, <IPv6 literal>, and <IPv4 literal> are supported)
- `IP` (Note: "loopback" is supported)
- `PROTOCOL` (Note: tcp and tcps are supported)
- `SDU` (Note: `256` to `65536` are supported)
- `SECURITY: SSL_VERSION` (Note: overrides sqlnet.ora:ssl_version)
- `TRANSPORT_CONNECT_TIMEOUT` (Note: overrides tcp.connect_timeout)

Note:

- SSL is now supported via method MCS and FILE.
- Both Kerberos5 and NTS authentication are supported. RADIUS is not supported.
- Only NTS authentication is supported. No RADIUS nor Kerberos5 authentication.
- Only Net Services, Easy Connect naming, and LDAP (namely, Active Directory and Oracle Internet Directory) are supported.
- No bequeath (`beq`) support. Default address is instead TCP loopback with port 1521 and Oracle service name from environment (`ORACLE_SID`)

Though managed ODP.NET does not support TNS descriptor based load balancing, it does support failover through both an ADDRESS_LIST and DESCRIPTION_LIST.

Note that you need not specify either the LOAD_BALANCE or the FAILOVER directive, because only failover is supported. The directives are ignored.

The following examples demonstrate TNS descriptors utilizing failover:

```
(DESCRIPTION=
   (ADDRESS_LIST=
     (ADDRESS=(PROTOCOL=tcp)(HOST=host1)(PORT=1630))
     (ADDRESS=(PROTOCOL=tcp)(HOST=host2)(PORT=1630))
     (ADDRESS=(PROTOCOL=tcp)(HOST=host3)(PORT=1521)))
   (CONNECT_DATA=(SERVICE_NAME=Sales.us.example.com)))

(DESCRIPTION_LIST=
 (DESCRIPTION=
  (ADDRESS_LIST=
   (ADDRESS=(PROTOCOL=tcp)(HOST=sales1a-svr)(PORT=1521))
   (ADDRESS=(PROTOCOL=tcp)(HOST=sales1b-svr)(PORT=1521)))
  (CONNECT_DATA=(SERVICE_NAME=sales1.example.com)))
 (DESCRIPTION=
  (ADDRESS_LIST=
   (ADDRESS=(PROTOCOL=tcp)(HOST=sales2a-svr)(PORT=1521))
   (ADDRESS=(PROTOCOL=tcp)(HOST=sales2b-svr)(PORT=1521)))
  (CONNECT_DATA=(SERVICE_NAME=sales2.us.example.com))))
```

2.9.3 settings section

This section can appear only under a <version> section. Any ODP.NET, Managed Driver specific settings should appear in this section. The following is an example of a settings section:

```
<settings>
  <setting name="TraceLevel" value="7" />
  <setting name="TraceOption" value="1"/>
  <setting name="TNS_ADMIN" value="C:\oracle\work"/>
</settings>
```

A new default behavior has been introduced for ODP.NET Release 12.1.0.2 and higher when InitialLobFetchSize is set to -1. The new default value is LegacyEntireLOBFetch = 0. To use the old behavior, set LegacyEntireLobFetch = 1 in the ODP.NET configuration as explained in Setting InitialLONGFetchSize to -1.

ODP.NET, Managed Driver configuration settings that are supported:

- BindByName
- DbNotificationPort
- DemandOraclePermission
- Disable_Oob: Interrupts database query execution via either TCP/IP urgent data or normal TCP/IP data, called out of band data (default) or in band data, respectively. (Default=off).

  All Oracle database clients support interrupting database query execution, such as through an ODP.NET command timeout. Windows-based database servers only support in band breaks, whereas all other (predominantly UNIX-based) database servers can support out of band (OOB) or in band breaks. ODP.NET, Managed

Driver uses OOB breaks by default with database servers that support it. For certain network topologies, the routers or firewalls involved in the route to the database may have been configured to drop urgent data or in band the data. If the routers or firewalls can not be changed to handle urgent data appropriately, then the ODP.NET, Managed Driver can be configured to utilize in band breaks by setting the .NET configuration parameter `Disable_Oob` to `on`.

- `FetchSize`
- `LDAP_ADMIN`: Specifies the `ldap.ora` location. The `LDAP_ADMIN` setting works in conjunction with the `TNS_ADMIN` setting to set `ldap.ora` search order.
- `LegacyEntireLOBFetch`
- `MaxStatementCacheSize`
- `MetaDataXml`
- `NAMES.DIRECTORY_PATH`: The default search order is `TNSNAMES` and `EZCONNECT`. `TNSNAMES`, `LDAP`, and `EZCONNECT` are the only name resolution methods supported, but their order of precedence can be modified.
- `NAMES.LDAP_AUTHENTICATE_BIND`
- `NAMES.LDAP_CONN_TIMEOUT`
- `NODELAY`
- `ORA_DEBUG_JDWP`: Allows Oracle PL/SQL Debugger and database to connect automatically without application code changes. Value is set as `host=<`*`IP_address or host_name`*`>;port=<`*`debugging port number`*`>`. Ex. `host=localhost;port=1234`
- `ORACLE_SID`
- `PerformanceCounters`
- `RECEIVE_BUF_SIZE`: Sets TCP `SO_RECVBUF`, the total buffer space associated with the local side of a TCP socket
- `SelfTuning`
- `SEND_BUF_SIZE`: Sets TCP `SO_SENDBUF`, the total buffer space associated with the local side of a TCP socket
- `ServiceRelocationConnectionTimeout`

  In seconds. (`Default = 90`).

  Whenever a database service becomes unavailable, such as due to a service being relocated, an application can encounter numerous connectivity errors during this time. To avoid unnecessary connection attempts to an unavailable service which will result in an error, ODP.NET, Managed and Unmanaged Drivers block any connection attempts until the service is up or until this property's specified time limit expires from the time when the service DOWN event was received, whichever comes first. Once the specified time elapses, all the connection attempts to the specific service which is known to be down will no longer be blocked. Those requests will be sent to the server. `ServiceRelocationConnectionTimeout` is only operational in conjunction with Oracle Fast Connection Failover (`HA Events = true`). Once Fast Connection Failover is enabled for the .NET application, Service Relocation Connection Timeout is automatically enabled. It will use its default value if no `ServiceRelocationConnectionTimeout` value has been explicitly set. It works with planned and unplanned outages.

- SQLNET.AUTHENTICATION_SERVICES: Supported values are Kerberos5, NTS, TCPS, or NONE.

  Managed ODP.NET supports NTS, Kerberos5, and TCPS external authentication methods. This setting should be set based on the desired database authentication method. If internal database authentication is desired, then the setting should be set to NONE. Example settings made in sqlnet.ora are:

  ```
  SQLNET.AUTHENTICATION_SERVICES = (TCPS)
  SQLNET.AUTHENTICATION_SERVICES = (NTS)
  SQLNET.AUTHENTICATION_SERVICES = (Kerberos5, NTS)
  SQLNET.AUTHENTICATION_SERVICES = (NONE)
  ```

  Note:

  The NTS external authentication methodology is only supported on a Windows-based client and server.

- SQLNET.CRYPTO_CHECKSUM_CLIENT: Specifies the desired data integrity behavior when this client connects to a server. Supported values are accepted, rejected, requested, or required. Default = accepted.
- SQLNET.CRYPTO_CHECKSUM_TYPES_CLIENT: Specifies the data integrity algorithms that this client uses. Supported values are SHA512, SHA384, SHA256, SHA1, and MD5.
- StatementCacheSize
- SSL_SERVER_DN_MATCH: To enforce the distinguished name (DN) for the database server matches its service name. (Default=no).

  If you enforce the match verification, then SSL/TLS ensures that the certificate is from the server. If you select to not enforce the match verification, then SSL/TLS performs the check but allows the connection, regardless if there is a match. Not enforcing the match allows the server to potentially fake its identify.

  Supported values: yes | on | true to enforce a match.

  Supported values: no | off | false to not enforce a match.

  SSL_SERVER_DN_MATCH is often used together with SSL_SERVER_CERT_DN. SSL_SERVER_CERT_DN specifies the distinguished name (DN) of the database server. It can be set in the connect descriptor.

  ```
  net_service_name=
    (DESCRIPTION=
      (ADDRESS=(PROTOCOL=tcp)(HOST=sales1-svr)(PORT=1521))
      (ADDRESS=(PROTOCOL=tcp)(HOST=sales2-svr)(PORT=1521))
      (CONNECT_DATA=
        (SERVICE_NAME=sales.us.acme.com))
        (SECURITY=
         (SSL_SERVER_CERT_DN="cn=sales,cn=OracleContext,dc=us,dc=acme,dc=com")))
  ```

  The client uses this information to obtain the list of DNs it expects for each of the servers, enforcing the database server DN to match its service name. Use this parameter with SSL_SERVER_DN_MATCH to enable server DN matching.

- SSL_VERSION: Sets the version of the SSL/TLS connection. By default, all supported versions are enabled, in the order 3.0, 1.0, 1.1, and 1.2.

The client and server negotiate to the highest version among the common conversions specified in their configurations. The versions from lowest to highest are: 3.0 (lowest), 1.0, 1.1, and 1.2 (highest).

- `TNS_ADMIN`: Location where either one or more of `tnsnames.ora`, `ldap.ora`, and `sqlnet.ora` are located. Locations can consist of either absolute or relative directory paths.
- `TraceFileLocation`: Trace file destination directory, for example, `D:\traces`. The default `TraceFileLocation` is `<`*`Windows user temporary folder`*`>\ODP.NET\managed\trace`.
- `TraceLevel`: `1` = public APIs; `2` = private APIs; `4` = network APIs/data. These values can be `OR`ed. To enable everything, set `TraceLevel` to `7`. Errors will always be traced.
- `TraceOption`
- `TCP.CONNECT_TIMEOUT`
- `WALLET_LOCATION`: Microsoft Certificate Store (MCS) and file system wallets are supported.
- `SQLNET.ENCRYPTION_CLIENT` = Negotiates whether to turn on encryption. Supported values are accepted, rejected, requested, or required.
- `SQLNET.ENCRYPTION_TYPES_CLIENT` = Encryption algorithm(s) to use.

The following table lists the valid encryption algorithms for ODP.NET, Managed Driver.

Table 2-3 Encryption Algorithms for ODP.NET, Managed Driver

Algorithm Name	Legal Value
AES 128-bit key	AES128
AES 192-bit key	AES192
AES 256-bit key	AES256
RC4 128-bit key	RC4_128
RC4 256-bit key	RC4_256
2-key 3DES	3DES112
3-key 3DES	3DES168

2.9.4 LDAPsettings section

This section can appear only under a `<version>` section. Any ODP.NET, Managed Driver specific LDAP settings should appear in this section. The following is an example of a `<LDAPsetting>` subsection under the `<LDAPsettings>` section:

```
<LDAPsettings>
  <LDAPsetting name="DIRECTORY_TYPE" value="AD" />
  <LDAPsetting name="DEFAULT_ADMIN_CONTEXT" value="dc=Oracle,dc=com"/>
</LDAPsettings>
```

2.9.5 Lightweight Directory Access Protocol

ODP.NET, Managed Driver supports TNS alias resolution through a LDAP server/ service, specifically Microsoft Active Directory and Oracle Internet Directory (OID). TNS alias resolution occurs when using the `LDAPsettings` section or `ldap.ora` file settings. The `LDAPsettings` section settings take precedence over `ldap.ora` settings.

For Active Directory, only the `DIRECTORY_TYPE` and `DEFAULT_ADMIN_CONTEXT` parameters are required in `ldap.ora`. When the `DIRECTORY_SERVERS` parameter is missing or has no value, the default LDAP server for the current domain will be used.

For OID, all `ldap.ora` parameters must be set with valid values to complete configuration.

ODP.NET, Managed Driver and ODP.NET, Unmanaged Driver support the same level of security when using LDAP for name resolution.

Table 2-4 Microsoft Active Directory: Encryption Types and Authentication Credentials For Connecting and Binding

No Encryption	SSL Encryption
Anonymous authentication	Anonymous authentication
Domain User authentication	Domain User authentication

Table 2-5 Oracle Internet Directory: Encryption Types and Authentication Credentials For Connecting and Binding

No Encryption	SSL Encryption
Anonymous authentication	Anonymous authentication
-	Wallet based authentication *Note: Wallet based authentication for Oracle Internet Directory is not supported for this release*

2.9.6 implicitRefCursor section

This section can appear only under a `<version>` section. Any information about `REF CURSOR` parameters that need to be bound implicitly should appear in this section. The following is an example of an `<implicitRefCursor>` section:

```
<implicitRefCursor>
  <storedProcedure schema="USERREFCUR" name="TestProc1">
    <refCursor name="Param3">
     <bindInfo mode="Output"/>
     <metadata columnOrdinal="0" columnName="DEPTNO" baseColumnName="DEPTNO"
baseSchemaName="USERREFCUR" baseTableName="DEPT" nativeDataType="number"
providerType="Int32" dataType="System.Int16" columnSize="2" allowDBNull="true" />
     <metadata columnOrdinal="1" columnName="DNAME" baseColumnName="DNAME"
baseSchemaName="USERREFCUR" baseTableName="DEPT" nativeDataType="varchar2"
providerDBType="String" columnSize="30" />
    </refCursor>
    <refCursor name="param2">
```

```
      <bindInfo mode="Output"/>
      <metadata columnOrdinal="0" columnName="EMPNO" baseColumnName="EMPNO"
baseSchemaName="USERREFCUR" baseTableName="EMP" nativeDataType="number"
providerType="Int32" dataType="System.Int16" columnSize="4" allowDBNull="false" />
    </refCursor>
   </storedProcedure>

   <!--Next stored procedure information-->
   <storedProcedure name="TestProc2">
      ...
      ...
   </storedProcedure>
</implicitRefCursor>
```

2.9.7 distributedTransaction section

This section can appear only under a `<version>` section. Any information about distributed transactions should appear in this section. The following is an example of a `distributedTransaction` section:

```
<distributedTransaction>
  <setting name="OMTSRECO_IP_ADDRESS" value="my-pc" />
  <setting name="OMTSRECO_PORT" value="2040" />
  <setting name="ORAMTS_SESS_TXNTIMETOLIVE" value="240" />
</distributedTransaction>
```

- `OMTSRECO_IP_ADDRESS`: Specifies the machine name (or IP address) that the OraMTS Recovery service will be running on to resolve database in-doubt transactions. The default is the local machine name.
- `OMTSRECO_PORT`: Specifies the port that the OraMTS Recovery service will be listening on to resolve database in-doubt transactions. The default is 2030.
- `ORAMTS_SESS_TXNTIMETOLIVE` : Specifies the time in seconds that the transaction can remain inactive after it has been detached or delisted from the database. Once this time expires, the transaction is automatically terminated by the provider. The default is 120 seconds.
- `UseManagedDTC`: When set to `false` and using .NET Framework 4.5.2 or higher, ODP.NET uses .NET Framework for distributed transaction support. In all other instances, ODP.NET uses Oracle Services for Microsoft Transaction Server to support distributed transactions. Boolean (`Default = false`) for ODP.NET, Managed Driver only.
- `UseOraMTSManaged`: When set to `true` and using .NET Framework 4.5.2 or higher, ODP.NET uses managed code for distributed transactions. If set to `true`, but .NET 4.5.1 or lower is used, an exception will be thrown. If set to `false`, ODP.NET uses Oracle Services for Microsoft Transaction Server to support distributed transactions. Boolean (`Default = false`) for ODP.NET, Unmanaged Driver only.

2.9.8 edmMappings section

This section can appear only under a `<version>` section. Any information related to EDM mappings should appear in this section. Refer to Oracle Number Default Data Type Mapping and Customization for more examples on `edmMappings` section.

2.9.9 onsConfig section

Oracle Notification Service (ONS) can be configured using either local or remote configuration. Remote configuration is the preferred configuration for standalone client applications. For releases earlier than Oracle Database 12c, this section is mandatory for ODP.NET to receive ONS notifications. With Oracle Database 12c and later, this section is optional and the information about the ONS daemons is received from the server itself. However, ODP.NET will also listen for events from any `<host:port>` pairs that is provided by the user in this section in addition to the `<host:port>` pairs received from the server.

For local configuration, please ensure that ONS is configured and available on the node where ODP.NET is running, so that ODP.NET can receive events directly from the local ONS daemon. The following is a sample format for the local configuration:

```
<onsConfig configFile="C:\temp\test.config" mode="local">
</onsConfig>
```

Note:

The `configFile` specified in .NET config should contain the same `localport` and `remoteport` values as specified in the `ons.config` used by the local ONS daemon. This will enable the application to receive events from the local ONS daemon.

Remote configuration is used in scenarios where the application directly receives ONS events from the ONS daemons running on remote machines. One of the advantages of this configuration is that no ONS daemon is needed on the client end and, therefore, there is no need to manage this process.

The following is a sample format for remote configuration:

```
  <onsConfig mode="remote">
        <ons database="db1">
          <add name="nodeList" value="racnode1:4100, racnode2:4200" />
        </ons>
        <ons database="db2">
          <add name="nodeList" value=" racnode3:4100, racnode4:4200" />
        </ons>
      </onsConfig>
```

In case of remote configuration, the application has to specify the `<host>:<port>` values for every potential database that it can connect to. The `<host>:<port>` value pairs represent the ports on the the different Oracle RAC nodes where the ONS daemons are talking to their remote clients.

2.9.10 Client Side ONS Daemon Configuration

ONS configuration is controlled by the ONS configuration file, *`ORACLE_HOME`*`/opmn/conf/ons.config`. This file tells the ONS daemon how it should behave. The `SRVCTL` utility can be used to start and stop the ONS daemon. It is installed on each node by default during server install.

Configuration information within ons.config is defined in simple name and value pairs. An example of `ONS.config` is given below

```
# This is an example ons.config file
#
# The first three values are required
localport=4100
remoteport=4200
nodes=racnode1.example.com:4200,racnode2.example.com:4200
```

Some parameters in the ons.config file are required and some are optional. Table Table 2-6 lists the required ONS configuration parameters and Table 2-7 lists the optional ONS configuration parameters.

Table 2-6 Required ONS Configuration Parameters

Parameter	Explanation
`localport`	The port that ONS binds to on the local host interface to talk to local clients. For example, `localport=4100`
`remoteport`	The port that ONS binds to on all interfaces for talking to other ONS daemons. For example, `remoteport=4200`
`nodes`	A list of other ONS daemons to talk to. Node values are given as a comma-delimited list of either host names or IP addresses plus ports. The port value that is given is the remote port that each ONS instance is listening on. In order to maintain an identical file on all nodes, the `host:port` of the current ONS node can also be listed in the nodes list. It will be ignored when reading the list. For example, `nodes=myhost.example.com:4200,123.123.123.123:4200` The nodes listed in the nodes line correspond to the individual nodes in the Oracle RAC instance. Listing the nodes ensures that the middle-tier node can communicate with the Oracle RAC nodes. At least one middle-tier node and one node in the Oracle RAC instance must be configured to see one another. As long as one node on each side is aware of the other, all nodes are visible. You need not list every single cluster and middle-tier node in the ONS configuration file of each Oracle RAC node. In particular, if one ONS configuration file cluster node is aware of the middle tier, then all nodes in the cluster are aware of it.

Table 2-7 Optional ONS Configuration Parameters

Parameter	Description
`loglevel`	The level of messages that should be logged by ONS. This value is an integer that ranges from 1, which indicates least messages logged, to 9, which indicates most messages logged. The default value is 3. For example, `loglevel=3`
`logfile`	A log file that ONS should use for logging messages. The default value for log file is `$ORACLE_HOME/opmn/logs/ons.log`. For example, `logfile=C:\app\user\product\12.1.0\opmn\logs\myons.log`

Table 2-7 (Cont.) Optional ONS Configuration Parameters

Parameter	Description
walletfile	The wallet file used by the Oracle Secure Sockets Layer (SSL) to store SSL certificates. If a wallet file is specified to ONS, then it uses SSL when communicating with other ONS instances and require SSL certificate authentication from all ONS instances that try to connect to it. This means that if you want to turn on SSL for one ONS instance, then you must turn it on for all instances that are connected. This value should point to the directory where your ewallet.p12 file is located. For example, walletfile=C:\app\user\product\12.1.0\opmn \conf\ssl.wlt\default
useocr	The value, reserved for use on the server-side, to indicate ONS whether it should store all Oracle RAC nodes and port numbers in Oracle Cluster Registry (OCR) instead of the ONS configuration file or not. A value of useocr=on is used to store all Oracle RAC nodes and port numbers in Oracle Cluster Registry (OCR). Do not use this option on the client-side.

The ons.config file allows blank lines and comments on lines that begin with the number sign (#).

2.9.11 Relative Windows Path and Windows Environment Variable Configuration Settings

The following managed ODP.NET configuration settings support relative Windows path and environment variables:

- TraceFileLocation
- WALLET_LOCATION

File locations for the above config parameters can now be set using relative Windows paths. The "." notation informs ODP.NET to use the current working directory. Sub-directories can be added by appending them. For example, .\mydir refers to the sub-directory mydir in the current working directory. To navigate to a parent directory, use the ".." notation.

For web applications, the current working directory is the application directory. For Windows applications, the .EXE location is the current working directory.

Windows paths can also be set using Windows environment variable names within "%" characters.

For example, %tns_admin%, c:\%dir%\my_app_location, c:\%top_level_dir%\ %bottom_level_dir% etc.

Note:

- If the environment variable that is used by the configuration parameter is not set to anything, then an exception will be thrown.
- A directory name cannot partially be using an environment variable. For example, `c:\my_app_%id%`
- Multiple variables can used in given directory location. For example, `c:\%top_level_dir%\%bottom_level_dir%`.

2.10 Distributed Transactions

ODP.NET, Managed and Unmanaged Drivers provide its resource manager, which manage Oracle database transactions, and work in cooperation with Microsoft Distributed Transaction Coordinator (MSDTC) to guarantee atomicity and isolation to an application across networks. MSDTC coordinates with all the resource managers that are enlisted to the same `System.Transactions`, to perform 2-phrase commit or rollback atomically. With that, Oracle distributed transactions can then be committed or rolled back across networks properly.

2.10.1 Oracle Services for Microsoft Transaction Server

Oracle Services for Microsoft Transaction (OraMTS) allow client components to leverage Oracle database participation in MSDTC transactions. It acts as a proxy for the Oracle database to MSDTC to ensure that Oracle distributed database transactions commit or rollback together with the rest of the distributed transaction.

If a failure occurs, such as a network failure or server hardware failure, then it can leave an in-process distributed transaction in-doubt. OraMTS has a recovery service to resolve these transactions on the machine that began this transaction. This recovery service runs as a Windows service.

It is required to install the OraMTS Recovery Service on all the client machines where ODP.NET is running and participating in MSDTC. As a machine may have multiple IP addresses, administrators for managed ODP.NET applications can specify the host machine name or IP address that has the running recovery service in the application's .NET configuration file. ODP.NET, Unmanaged Driver resolves the IP/machine name for the recovery service automatically.

With .NET Framework 4.5.2, Microsoft introduced new API support that allows Oracle to use only managed calls to coordinate ODP.NET transactions with the MSDTC. ODP.NET utilizes this managed code with the managed driver (starting with ODAC 12*c* Release 3) and with the unmanaged driver (starting with ODAC 12*c* Release 4).

While ODP.NET, Unmanaged Driver developers can opt out of using OraMTS when using the latest .NET Framework and ODP.NET versions, they still need to install and configure the OraMTS Windows recovery service to manage recovery scenarios.

Table 2-8 Supported ODP.NET Type and .NET Framework Version for Distributed Transaction

ODP.NET Type	.NET Framework Version	Distributed Transaction Support
Managed	4.5.2 and higher	Uses .NET Framework's native managed implementation (default) for distributed transactions. This is Oracle's recommended approach.
Managed	4.5.1 and lower	Uses the `Oracle.ManagedDataAccessDTC.dll`
Unmanaged	4.5.2 and higher	Uses OraMTS (default) or managed OraMTS implementation. Oracle recommends using managed OraMTS for unmanaged ODP.NET applications requiring high availability from Oracle RAC or Data Guard.
Unmanaged	4.5.1 and lower	Uses OraMTS

Note:

While .NET Framework 4.5.1 and lower within the .NET Framework 4 family are no longer supported, administrators can still use any of the distributed transaction configurations listed above in conjunction with .NET 4.5.2 and higher. For .NET 4.5.1 and lower, the table merely recommends specific setups based on user configuration. They are not requirements.

2.10.2 ODP.NET, Managed Driver Setup

This section explains the setup and configuration steps required for using distributed transactions with ODP.NET, Managed Driver.

Oracle recommends that applications use .NET's native managed distributed transaction implementation (default), which is available in .NET Framework 4.5.2 or higher. Applications can set whether .NET's native managed distributed transaction or `Oracle.ManagedDataAccessDTC.dll` is used by setting the `UseManagedDTC` parameter in the .NET configuration file. Follow these steps to configure distributed transactions in these .NET Framework versions:

1. Create and setup the OraMTS recovery service or make sure an existing recovery service is running.
2. Set the value of `OMTSRECO_PORT` in the .NET configuration to specify the port number that the OraMTS recovery service is running.

Alternatively, you can still use `Oracle.ManagedDataAccessDTC.dll` with .NET Framework 4.5.2 and managed ODP.NET. To do so, set `UseManagedDTC` to `true` and follow the instructions listed below for .NET Framework 4.5.1.

For .NET Framework 4.5.1 and lower applications, follow these steps to setup and configure managed ODP.NET for distributed instructions:

1. Create and setup the OraMTS recovery service or make sure an existing recovery service is running.
2. Deploy `Oracle.ManagedDataAccessDTC.dll` along with the application.
3. Set the value of `OMTSRECO_PORT` in the .NET configuration to specify the port number that the OraMTS recovery service is running.

`Oracle.ManagedDataAccessDTC.dll` is included with ODP.NET, Managed Driver. This DLL makes unmanaged MSDTC COM calls to MSDTC, which means there is a 32-bit version and 64-bit version of this DLL. These two DLLs share the same name. If you are using 32-bit .NET Framework, then deploy the 32-bit `Oracle.ManagedDataAccessDTC.dll`. If you are using 64-bit .NET Framework, then deploy the 64-bit `Oracle.ManagedDataAccessDTC.dll`. The DLLs are located in the following directories:

- For 32-bit .NET Framework: *`ORACLE_HOME`*`\odp.net\managed\x86`
- For 64-bit .NET Framework: ORACLE_HOME`\odp.net\managed\x64`

Upon ODP.NET installation, `Oracle.ManagedDataAccessDTC.dll` is no longer placed into the Global Assembly Cache (GAC). For applications that use this DLL, `Oracle.ManagedDataAccessDTC.dll` must either be placed in the application directory or in the GAC.

`Oracle.ManagedDataAccessDTC.dll` should not be directly referenced by a .NET application. It will be implicitly loaded by ODP.NET, Managed Driver when using distributed transactions.

For applications with platform target `x64` or `x86` specifically, `Oracle.ManagedDataAccess.dll` will load `Oracle.ManagedDataAccessDTC.dll` appropriately if it is placed into the GAC or if it resides in the application directory.

For applications that target AnyCPU, the corresponding `Oracle.ManagedDataAccessDTC.dll` needs to be placed into x64 and x86 subdirectories under wherever the `Oracle.ManagedDataAccess.dll` is loaded from by the application. ODP.NET, Managed Driver will load the appropriate `Oracle.ManagedDataAccessDTC.dll` assembly (32-bit or 64-bit), based on whether the application is 32-bit or 64-bit. If both 32-bit and 64-bit versions of `Oracle.ManagedDataAccessDTC.dll` are in the GAC, then the appropriate assemblies will be loaded automatically.

2.10.3 ODP.NET, Unmanaged Driver Setup

This section explains the setup and configuration steps required for using distributed transactions with ODP.NET, Unmanaged Driver.

For .NET Framework 4.5.2 and higher, ODP.NET, Unmanaged Driver has embedded a managed OraMTS implementation into its assembly. OraMTS remains the default implementation for the ODP.NET, Unmanaged Driver, but the managed OraMTS implementation is recommended when using any high availability FAN operations (HA Events = true) with Oracle Real Application Clusters or Oracle Data Guard. The managed OraMTS implementation supports this high availability functionality, while the traditional OraMTS does not.

Applications can set whether OraMTS (default) or managed OraMTS is used by setting the `UseOraMTSManaged` parameter in the .NET configuration file.

Install and configure OraMTS, including its recovery service to use OraMTS implementation for ODP.NET, Unmanaged Driver.

For .NET Framework 4.5.2 and higher applications, you can use the managed OraMTS implementation instead of the traditional OraMTS. To set this up, perform the following steps:

1. Set `UseOraMTSManaged` to `true` in the .NET configuration file.
2. Create and setup the OraMTS recovery service or make sure an existing recovery service is running.

2.11 Configuration differences between ODP.NET, Managed Driver and ODP.NET, Unmanaged Driver

Table 2-9 lists other configuration differences between ODP.NET, Managed Driver and ODP.NET, Unmanaged Driver.

Table 2-9 Configuration Differences between ODP.NET, Unmanaged Driver and ODP.NET, Managed Driver

Feature Category	Difference compared to ODP.NET, Unmanaged Driver
Configuration	The older, traditional ODP.NET, Unmanaged Driver configuration file format is different. The new format allows both providers to share the same format. See "Oracle Data Provider for .NET, Managed Driver Configuration."
Configuration	`ConfigSchema.xsd` file, shipped with ODP.NET, Managed Driver (when included as part of the schema (XML->Schemas) in Visual Studio) enables `app.config` intelli-sense.
Configuration	Windows Registry based configuration is not supported
Configuration	Oracle High Availability (HA) & Oracle RAC Load Balancing (RLB) notifications use Oracle Notification Service (ONS). Thus, to use HA or RLB, configure database and client to use ONS, rather than Oracle Database Advanced Queuing (AQ). Note that Continuous Query Notification will continue to use AQ.
Configuration Parameter	`Edition` is not supported.
Configuration Parameter	`CheckConStatus` is not supported.
Configuration Parameter	`DllPath` is not supported.
Configuration Parameter	`SatementCacheWithUdts` is not supported.
Configuration Parameter	`ThreadPoolMaxSize` is not supported.
Configuration Parameter	`TraceFileName` is not supported.
Configuration Parameter	`UdtCacheSize` is not supported.
Configuration Parameter	*`UDT Mapping`* is not supported.
Configuration Parameter	`UseManagedDTC` is supported by ODP.NET, Managed Driver only.
Configuration Parameter	`UseOraMTSManaged` is not supported.
Connection String	Context Connection is not supported.
Connection String	`LegacyTransactionBindingBehavior` setting will be ignored. It will always be set to the default value of 1.
Connection String	`Promotable Transaction` setting will be ignored. It will always be set to `promotable` and always support promotions.
Connection String	`Statement Cache Purge` is not supported.
Connectivity	Connection to Oracle Times Ten Database is not supported.

Table 2-9 (Cont.) Configuration Differences between ODP.NET, Unmanaged Driver and ODP.NET, Managed Driver

Feature Category	Difference compared to ODP.NET, Unmanaged Driver
Performance Monitor	`NumberOfStatisConnections` performance counter is not supported.
Performance Monitor	Performance monitor category name is "ODP.NET, Managed Driver"
Provider Types	Provider Types accept (via constructors) and generate (via `ToString()` methods) only culture-invariant strings
Tracing	Dynamic tracing is enabled by changing the `TraceLevel` setting in the `app/web/machine.config`. NOTE: For ASP.NET applications, doing so will recycle the application domain.

2.12 Configuring for Entity Framework Code First

Developers must configure applications to use the Oracle Entity Framework functionality. This consists of creating two entries in the `app.config` or `web.config` file and adding an assembly reference:

- Add entries in the .NET config file
  - Connection string

    A standard ADO.NET connection string is used rather than the Entity Framework connection string used by Database First or Model First paths. The connection string name should match the application context name. The connection string entry is an element of the `connectionStrings` section in the configuration file.
  - Provider registration

    Entity Framework uses the provider registration to determine the assembly to use for Oracle Entity Framework functionality. The provider registration is an element of the `providers` section within the `entityFramework` section in the application configuration file.
- Add Assembly reference

  Add Oracle Entity Framework assembly to the project references.

Note:

When using the official ODP.NET, NuGet installation, these preceding sections are created automatically, if they do not already exist. After the NuGet install, the ODP.NET connection string will need to be customized to the application's specific settings.

When using the Oracle Universal Installer or xcopy install, the preceding sections must all be configured manually.

Examples of connection strings are as follows:

- ODP.NET, Unmanaged Driver

```
<add name="TestContext" providerName="Oracle.DataAccess.Client"
connectionString="User Id=test;Password=testpassword;Data Source=eftest" />
```

- ODP.NET, Managed Driver

```
<add name="TestContext" providerName="Oracle.ManagedDataAccess.Client"
connectionString="User Id=test;Password=testpassword;Data Source=eftest" />
```

Examples of Oracle provider registration are as follows:

- ODP.NET, Unmanaged Driver

```
<provider invariantName="Oracle.DataAccess.Client"
type="Oracle.DataAccess.EntityFramework.EFOracleProviderServices,
Oracle.DataAccess.EntityFramework, Version=6.121.2.0, Culture=neutral,
PublicKeyToken=89b483f429c47342" />
```

- ODP.NET, Managed Driver

```
<provider invariantName="Oracle.ManagedDataAccess.Client"
type="Oracle.ManagedDataAccess.EntityFramework.EFOracleProviderServices,
Oracle.ManagedDataAccess.EntityFramework, Version=6.121.2.0, Culture=neutral,
PublicKeyToken=89b483f429c47342" />
```

2.12.1 Entity Framework 6 Code-Based Registration

Entity Framework 6 allows an application to register with an Entity Framework provider without using any configuration file. With ODP.NET, Managed Driver, the code will look as follows:

```
// C#
using Oracle.ManagedDataAccess.EntityFramework;
...
public class ModelConfiguration : DbConfiguration
{
   public ModelConfiguration()
   {
      SetProviderServices("Oracle.ManagedDataAccess.Client",
EFOracleProviderServices.Instance);
   }
}
```

For ODP.NET, Unmanaged Driver, replace occurrences of `ManagedDataAccess` with `DataAccess` in the preceding code.

If you are using code-based registration, then the configuration file should not include the registration. The configuration file based registration overrides the code-based registration.

2.13 Migrating from ODP.NET, Unmanaged Driver to ODP.NET, Managed Driver

To ease migration, the APIs of ODP.NET, Managed Driver are a complete subset of the APIs of ODP.NET, Unmanaged Driver. As long as the existing unmanaged ODP.NET applications use currently available managed ODP.NET APIs, migration is straightforward and simple.

In future versions, the managed driver will support more APIs of ODP.NET, Unmanaged Driver. Both drivers will continue to be enhanced to support the latest Oracle Database and .NET Framework features.

To migrate from unmanaged to managed ODP.NET, perform the following steps:

1. Add a Reference to `Oracle.ManagedDataAccess.dll` in the .NET project.
2. Change the existing ODP.NET, Unmanaged Driver namespace references to ODP.NET, Managed Driver references.

   ```
   // C#
   using Oracle.ManagedDataAccess.Client;
   using Oracle.ManagedDataAccess.Types;

   // VB
   Imports Oracle.ManagedDataAccess.Client
   Imports Oracle.ManagedDataAccess.Types
   ```

3. Some provider configuration settings may need to be migrated because ODP.NET, Managed Driver supports very few Windows Registry settings and a different .NET configuration setting format.

2.14 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
- Continuous Query 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.

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 in ODP.NET, Unmanaged Driver, for example, `1200`, an application or web configuration file can be used as follows:

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

To specify a particular port in ODP.NET, Managed Driver, an application or web configuration file can be used as follows:

```
<configuration>
  <oracle.manageddataaccess.client>
    <version number="*">
      <settings>
```

```
        <setting name="DbNotificationPort" value="1200"/>
      </settings>
    </version>
  </oracle.manageddataaccess.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.

2.15 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.

3 Features of Oracle Data Provider for .NET

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

This section contains the following topics:

3.1 Base Classes and Provider Factory Classes

With ADO.NET, 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 3-1 ODP.NET Classes that Inherit from ADO.NET 2.0 Base Classes

ODP.NET Classes	Inherited from ADO.NET 2.0 Base Class
`OracleClientFactory`	`DbProviderFactory`
`OracleCommand`	`DbCommand`
`OracleCommandBuilder`	`DbCommandBuilder`
`OracleConnection`	`DbConnection`
`OracleConnectionStringBuilder`	`DbConnectionStringBuilder`
`OracleDataAdapter`	`DbDataAdapter`
`OracleDataReader`	`DbDataReader`
`OracleDataSourceEnumerator`	`DbDataSourceEnumerator`
`OracleException`	`DbException`
`OracleParameter`	`DbParameter`
`OracleParameterCollection`	`DbParameterCollection`
`OracleTransaction`	`DbTransaction`

In general, applications still require Oracle-specific connection strings, SQL or stored procedure calls, and declare that a factory from ODP.NET is used.

3.2 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.

3.2.1 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 for unmanaged ODP.NET only, or an individual application configuration file for both unmanaged and managed ODP.NET.

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 for ODP.NET, Unmanaged Driver:

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

Similarly, you can use `DemandOraclePermission` to configure ODP.NET, Managed Driver under the settings section for managed provider configuration. See also "settings section" for more information.

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`.

3.2.2 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 `OracProvCfg` 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:

```
OraProvCfg.exe /action:config  /product:odp /component:oraclepermission
               /frameworkversion:v2.0.50727
               /providerpath:full_path_of_Oracle.DataAccess.dll
```

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:

```
OraProvCfg.exe /action:unconfig  /product:odp  /component:oraclepermission
               /frameworkversion:v2.0.50727
               /providerpath:full_path_of_Oracle.DataAccess.dll
```

3.2.3 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" />
```

3.3 Connecting to Oracle Database

Oracle Data Provider for .NET can connect to Oracle Database in a number of ways, such as using a user name and password, Windows Native Authentication, Kerberos, and Transport Layer Security/Secure Sockets Layer. This section describes `OracleConnection` provider-specific features, including:

- Connecting to Oracle Database Exadata Express Cloud Service
- Connection String Attributes
- Connection String Builder
- Specifying the Data Source Attribute
- Using Transport Layer Security and Secure Sockets Layer
- Using Secure External Password Store
- Using Kerberos
- Using Windows Native Authentication (NTS)
- Network Data Encryption and Integrity
- Schema Discovery
- Connection Pooling
- Connection Pool Management
- Connection Pool Performance Counters

- Pluggable Databases
- Edition-Based Redefinition
- 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

3.3.1 Connecting to Oracle Database Exadata Express Cloud Service

Managed and unmanaged ODP.NET supports connecting to Oracle Database Exadata Express Cloud Service.

Set-up Instructions

Oracle recommends using the latest ODAC version when connecting to Exadata Express. You can find instructions about how to download, install, and configure ODAC for Oracle Database Exadata Express Cloud Service at:

http://www.oracle.com/technetwork/topics/dotnet/tech-info/dotnetcloudexaexpress-3112654.html

Known Restrictions

Managed and unmanaged ODP.NET do not support the following features when connecting to Oracle Database Exadata Express Cloud Service:

- .NET Bulk Copy
- Advanced Queuing
- Any authentication besides username and password
- Application Continuity
- Client Result Cache
- Continuous Query Notification
- Data types
  - `BFILE`
  - User-Defined Types when using `IN` or `IN/OUT` parameter binding

    User-Defined Types include objects, collections (`VARRAY` and nested table), and references
  - `VARCHAR2` with increased size limit to 32 KB.

> **Note:**
>
> `VARCHAR2` of sizes up to 4 KB is supported.

  - XMLType when using `IN` or `IN/OUT` parameter binding
- Distributed transactions
- Fast Application Notification (FAN)
  - Features that rely on FAN, such as planned outage, run-time connection load balancing, and fast connection failover are not supported
  - In ODP.NET 12.1 or lower, ODP.NET applications will receive an error if FAN is turned on
- Sharding

3.3.2 Connection String Attributes

Table 3-2 lists the supported connection string attributes.

Table 3-2 Supported Connection String Attributes

Connection String Attribute	Description	Default Value
`Application Continuity`	Enables database requests to automatically replay transactional or non-transactional operations in a non-disruptive and rapid manner in the event of a severed database session, which results in a recoverable error. *Not available in ODP.NET, Managed Driver*	`true`
`Connection Lifetime`	Minimum life time (in seconds) of the connection.	`0`
`Connection Timeout`	Minimum time (in seconds) to wait for a free connection from the pool.	`15`
`Context Connection`	Returns an implicit database connection if set to `true`. *Supported in a .NET stored procedure only*	`false`
`Data Source`	Oracle Net Services Name, Connect Descriptor, or an easy connect naming that identifies the database to which to connect.	empty string
`DBA Privilege`	Administrative privileges: `SYSDBA` or `SYSOPER`.	empty string
`Decr Pool Size`	Number of connections that are closed when an excessive amount of established connections are unused.	`1`
`Enlist`	Controls the enlistment behavior and capabilities of a connection in context of COM+ transactions or `System.Transactions`.	`true`

Table 3-2 (Cont.) Supported Connection String Attributes

Connection String Attribute	Description	Default Value
`HA Events`	Enables ODP.NET connection pool to proactively remove connections from the pool when an Oracle database service, service member, instance, or node goes down. Works with Oracle Global Data Services, including Oracle RAC, Data Guard, GoldenGate, and some single instance deployments.	`true`
`Load Balancing`	Enables ODP.NET connection pool to balance work requests across Oracle database instances based on the load balancing advisory and service goal. Works with Oracle Global Data Services, including Oracle RAC, Active Data Guard, and GoldenGate.	`true`
`Incr Pool Size`	Number of new connections to be created when all connections in the pool are in use.	`5`
`Max Pool Size`	Maximum number of connections in a pool.	`100`
`Metadata Pooling`	Caches metadata information.	`True`
`Min Pool Size`	Minimum number of connections in a pool.	`1`
`Password`	Password for the user specified by `User Id`.	empty string
`Persist Security Info`	Retrieval of the password in the connection string.	`false`
`Pooling`	Connection pooling.	`true`
`Promotable Transaction`	Indicates whether or not a transaction is local or distributed throughout its lifetime.	promotable
`Proxy User Id`	User name of the proxy user.	empty string
`Proxy Password`	Password of the proxy user.	empty string
`Self Tuning`	Enables or disables self-tuning for a connection.	`true`
`Statement Cache Purge`	Statement cache purged when the connection goes back to the pool.	`false`
`Statement Cache Size`	Statement cache enabled and cache size, that is, the maximum number of statements that can be cached.	`0`
`User Id`	Oracle user name.	empty string
`Validate Connection`	Validation of connections coming from the pool.	`false`

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

```
// 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");
  }
}
```

3.3.3 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:

```
<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.

3.3.4 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)))
```

The connection pool will maintain the full descriptor of an alias so that subsequent connection requests with the same connection string will not need to resolve the alias again. This applies to `tnsnames.ora`, .NET config data sources, and LDAP aliases. To flush out the cached full descriptor maintained by the connection pool, invoke `OracleDataSourceEnumerator.GetDataSources()` followed by `OracleConnection.ClearPool()` or `OracleConnection.ClearAllPools()`.

If connection pooling is not used, the alias will need to be resolved to the full descriptor for each request. In the case of LDAP, the LDAP server is contacted for each connection request.

3.3.4.1 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";
```

3.3.4.2 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)))"
```

3.3.4.3 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 `NAMES.DIRECTORY_PATH` parameter in the `sqlnet.ora` file as follows:

```
NAMES.DIRECTORY_PATH= (TNSNAMES, EZCONNECT)
```

With this enabled, ODP.NET allows applications to specify the `"Data Source"` attribute in the form of:

```
//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.

3.3.4.4 Using LDAP

ODP.NET can connect with connect identifiers mapped to connect descriptors in an LDAP-compliant directory server, such as Oracle Internet Directory and Microsoft Active Directory.

To configure LDAP for ODP.NET, Unmanaged Driver, follow these Oracle documentation instructions in Configuring the Directory Naming Method in *Oracle Database Net Services Administrator's Guide*.

To configure LDAP for ODP.NET, Managed Driver, follow the instructions in "settings section" and "LDAPsettings section."

3.3.4.5 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.

3.3.5 Using Transport Layer Security and Secure Sockets Layer

Transport Layer Security (TLS) and its predecessor, Secure Sockets Layer (SSL), are industry standard protocols for securing network connections. Both managed and unmanaged ODP.NET support SSL for database and transport authentication.

3.3.5.1 Secure Sockets Layer and Transport Layer Security Differences

Although SSL was primarily developed by Netscape Communications Corporation, the Internet Engineering Task Force (IETF) took over development of it, and renamed it Transport Layer Security (TLS).

Essentially, TLS is an incremental improvement to SSL version 3.0.

ODP.NET, Managed Driver supports SSL 3.0 and TLS 1.0, 1.1, and 1.2. ODP.NET, Unmanaged Driver supports the same SSL and TLS versions as the Oracle Database Client version it is certified with.

The SSL/TLS client can ensure that the distinguished name (DN) is correct for the database server it is trying to connect to. Parameters for DN Matching are `SSL_SERVER_DN_MATCH` (`sqlnet.ora`) and `SSL_SERVER_CERT_DN` (`tnsnames.ora`), which can be defined in the .NET config file as well.

To turn DN Match on, set `SSL_SERVER_DN_MATCH` to `True` (or `On` or `Yes`). `SSL_SERVER_CERT_DN` is optional. It allows the administrator to specify exactly the DN they want to match. If the `SSL_SERVER_CERT_DN` is not set, then the match is done by comparing the `SERVICE_NAME` value to the Common Name (CN) portion of the server certificate's DN.

See Also:

- The TLS Protocol Version 1.0 [RFC 2246] at the IETF Web site, which can be found at:

  `http://www.ietf.org`
- `SSL_VERSION` in the "settings section."

Note:

To simplify the discussion, this section uses the term SSL where either SSL or TLS may be appropriate because SSL is the most widely recognized term. However, where distinctions occur between how you use or configure these protocols, this section specifies what is appropriate for either SSL or TLS.

3.3.5.2 ODP.NET Secure Sockets Layer Configuration

When you configure Secure Sockets Layer on the client, you must confirm that the wallet is created and use TCP/IP with SSL on the client. Optionally, you can perform additional steps to enhance the configuration.

SSL Configuration Topics:

- Step 1: Confirm Client Wallet Creation
- Step 2: Use TCP/IP with SSL on the Client
- Step 3: Specify Required Client SSL Configuration (Wallet Location)
- Step 4: Set the Required SSL Version on the Client (Optional)
- Step 5: Set SSL as an Authentication Service on the Client (Optional)

Step 1: Confirm Client Wallet Creation

Before proceeding to the next step, you must confirm that a wallet has been created on the client and that the client has a valid certificate.

ODP.NET, Managed Driver supports file and Microsoft Certificate Store (MCS) based wallets.

- For file-based wallets, use Oracle Wallet Manager to check that the wallet has been created. See Step 1A: Confirm Wallet Creation on the Server in *Oracle Database Security Guide* for information about checking a wallet.
- For MCS, the Windows domain credentials will be used for the client credentials. Thus, a valid domain logon must be used while running the ODP.NET application. ODP.NET, Managed Driver will retrieve the credentials from the MY or Personal certificate store. Note that the server must also be configured to use MCS wallets. See Microsoft Certificate Services in *Oracle Database Platform Guide for Microsoft Windows* for information about setting up the server for MCS.

Step 2: Use TCP/IP with SSL on the Client

The ODP.NET Data Source must be modified to use SSL. Specifically, the transport protocol must be changed to use TCP/IP with SSL or what Oracle calls "tcps". An example ODP.NET Data Source for use with SSL is:

```
finance = (DESCRIPTION=
  (ADDRESS = (PROTOCOL=tcps) (HOST=finance_server) (PORT=1575) )
  (CONNECT_DATA = (SERVICE_NAME=Finance.us.example.com) ) )
```

Step 3: Specify Required Client SSL Configuration (Wallet Location)

Edit the `sqlnet.ora` or .NET application configuration to specify the wallet location.

- An example of setting the SSL wallet location for file based wallets, where `<wallet_location>` is the specified location where the client wallet is stored:

  ```
  wallet_location = (SOURCE=(METHOD= File)
                            (METHOD_DATA=(DIRECTORY=<wallet_location>)))
  ```

- An example of setting the SSL wallet location for MCS based wallets is:

  ```
  wallet_location = (SOURCE=(METHOD= MCS))
  ```

Step 4: Set the Required SSL Version on the Client (Optional)

The `SSL_VERSION` parameter can be set through the `sqlnet.ora` or the .NET `application.config`, `web.congig`, or `machine.config` file. Normally, it is not necessary to set this parameter. The default setting for this parameter is `any`, which allows the database server to apply any necessary restrictions to the SSL version accepted. An example setting in the `sqlnet.ora` is:

```
SSL_VERSION=3.0
```

Step 5: Set SSL as an Authentication Service on the Client (Optional)

Set the `SQLNET.AUTHENTICATION_SERVICE` parameter in the `sqlnet.ora` or `application.config`, `web.congig`, or `machine.config` file to allow SSL to be used as a database external authentication methodology.

Note that SSL can be used as just a transport encryption vehicle. Hence, the "optional" designation for this setting.

If SSL is to be used as a database external Authentication Service, then a database externally authenticated user matching the client certificate must be created.

An example setting allowing SSL external authentication in the `sqlnet.ora` is:

```
SQLNET.AUTHENTICATION_SERVICES = (TCPS)
```

Note:

Prior to ODAC 12*c* Release 4, ODP.NET, Managed Driver SSL connections would be redirected to dynamic (ephemeral) port on the database server machine. With ODAC 12*c* Release 4 and later, managed ODP.NET SSL connections will now continue to the original socket connection to the Oracle Listener. Hence, firewalls will now only need to allow access to the Oracle Listener's port (e.g., 1521).

3.3.5.3 Troubleshooting TLS/SSL Setup

This section discusses commonly encountered issues and their typical resolution steps.

Common TLS/SSL Wallet Errors

Microsoft Windows now restricts wallets from using the MD5 algorithm. Oracle wallets may have been generated with this algorithm as that was the default option in Oracle Public Key Infrastructure (`orapki`) utility 12.1 and earlier.

Orapki refers to `orapki.exe`. This utility is part of full Oracle client (administrator) installations. It is not included with Oracle Instant Client. The utility is only needed to setup up the wallet; it is not necessary to deploy it with the wallet.

When you setup TLS/SSL and encounter an "ORA-0052: Failure during SSL handshake" error combined with a 0x80004005 error code and first inner exception "A SSPI-call failed" and second inner exception "A token sent to the function is invalid", then it is very likely that Microsoft Security Support Provider Interface (SSPI) rejected your Oracle Wallet, such as when MD5 is used. This is a failure on the handshake. You can resolve this error by using the SHA-2 algorithm instead.

If the second inner exception instead indicates "The credentials supplied to the package were not recognized", it is possible the user certificate was generated without a certificate authority (CA). You can resolve this error by using `orapki` to generate a CA/root certificate and then regenerating your user wallet/certificate to point to this new CA/root certificate.

The steps below will regenerate your Oracle Wallet using `orapki` and SHA-2. Any `orapki` version can be used to generate the wallet with these instructions.

1. Create root wallet, for example, a CA wallet.

   ```
   orapki wallet create -wallet ./root -pwd <password>
   ```

2. Add a self-signed certificate (CA certificate) to the root wallet.

   ```
   orapki wallet add -wallet ./root -dn 'CN=<my root>' -keysize 1024 -self_signed -
   validity 3650 -pwd <password> -sign_alg sha512
   ```

3. Export the self-signed certificate from the wallet.

   ```
   orapki wallet export -wallet ./root -dn 'CN=<my root>' -cert ./root/
   b64certificate.txt -pwd <password>
   ```

4. Create a user wallet, for example, a customer wallet.

   ```
   orapki wallet create -wallet ./user -pwd <password> -auto_login
   ```

5. Add a certificate request.

   ```
   orapki wallet add -wallet ./user -dn 'CN=<client's hostname>' -keysize 1024 -pwd
   <password> -sign_alg sha512
   ```

6. Export the certificate request.

   ```
   orapki wallet export -wallet ./user -dn 'CN=<client's hostname>' -request ./user/
   creq.txt -pwd <password>
   ```

7. Create a certificate issued by a CA.

   ```
   orapki cert create -wallet ./root -request ./user/creq.txt -cert ./user/cert.txt
   -validity 3650 -pwd <password> -sign_alg sha512
   ```

8. Add a trusted certificate (CA certificate) to the wallet. This example assumes the same CA for both the client and server wallets.

   ```
   orapki wallet add -wallet ./user -trusted_cert -cert ./root/b64certificate.txt -
   pwd <password>
   ```

9. Add a user certificate.

   ```
   orapki wallet add -wallet ./user -user_cert -cert ./user/cert.txt -pwd
   <password> -sign_alg sha512
   ```

10. Display contents of user wallet.

    ```
    orapki wallet display -wallet ./user -pwd <password>
    ```

11. Create a server wallet.

    ```
    orapki wallet create -wallet ./server -pwd <password> -auto_login
    ```

12. Add a server certificate request.

    ```
    orapki wallet add -wallet ./server -dn 'CN=<server's hostname>' -keysize 1024 -
    pwd <password> -sign_alg sha512
    ```

13. Export the certificate request.

    ```
    orapki wallet export -wallet ./server -dn 'CN=<server's hostname>' -request ./
    server/creq.txt -pwd <password>
    ```

14. Create a server certificate issued by a CA.

    ```
    orapki cert create -wallet ./root -request ./server/creq.txt -cert ./server/
    cert.txt -validity 3650 -pwd <password> -sign_alg sha512
    ```

15. Add a trusted certificate (CA certificate) to the server wallet. This example assumes the same CA for both the client and server wallets.

    ```
    orapki wallet add -wallet ./server -trusted_cert -cert ./root/b64certificate.txt
    -pwd <password>
    ```

16. Add an `user_cert` certificate for the server wallet.

    ```
    orapki wallet add -wallet ./server -user_cert -cert ./server/cert.txt -pwd
    <password> -sign_alg sha512
    ```

17. Display contents of server wallet.

    ```
    orapki wallet display -wallet ./server -pwd <password>
    ```

3.3.6 Using Secure External Password Store

The Secure External Password Store (SEPS) is the use of a client-side wallet for securely storing the password credentials. Both ODP.NET, Managed Driver and Unmanaged Driver can be configured to use the external password store.

An Oracle wallet is a container that securely stores authentication and signing credentials. Wallets can simplify large-scale deployments that rely on password credentials for database connections. Applications no longer need embedded user names and passwords, which reduces security risk.

3.3.6.1 Configuring Secure External Password Store (SEPS)

Steps for configuring SEPS:

- Step 1. Create the wallet file
- Step 2. Point the configuration to the client wallet
- Step 3. Turn on SEPS

Step 1. Create the wallet file

Use the `mkstore` utility to create the wallet file and insert the credentials.

Step 1a. Create a wallet on the client by using the following syntax at the command line:

```
mkstore -wrl wallet_location -create
```

For example:

```
mkstore -wrl c:\oracle\product\12.1.0\db_1\wallets -create
Enter password: password
```

Step 1b. Create database connection credentials in the wallet by using the following syntax at the command line:

```
mkstore -wrl wallet_location -createCredential db_connect_string username
Enter password: password
```

For example:

```
mkstore -wrl c:\oracle\product\12.1.0\db_1\wallets -createCredential orcl system
Enter password: password
```

Step 2. Point the configuration to the client wallet

In the client `sqlnet.ora` file, enter the `WALLET_LOCATION` parameter and set it to the directory location of the wallet you created in Step 1.

For example, if you created the wallet in `$ORACLE_HOME/network/admin` and your Oracle home is set to `C:\app\client\<user>\product\<version>\client_1\`, then you need to enter the following into your client `sqlnet.ora` file:

```
WALLET_LOCATION =
  (SOURCE =(METHOD = FILE)
           (METHOD_DATA =
              (DIRECTORY = C:\app\client\<user>\product\<version>\client_1\Network
\Admin) ) )
```

Step 3. Turn on SEPS

Step 3. Turn on SEPS

```
SQLNET.WALLET_OVERRIDE = TRUE
```

This setting causes all `CONNECT /@db_connect_string` statements to use the information in the wallet at the specified location to authenticate to databases.

When external authentication is in use, an authenticated user with such a wallet can use the `CONNECT /@db_connect_string` syntax to access the previously specified databases without providing a user name and password. Note however, that the wallet file needs to be kept up to date with the database credentials. If the database credentials change, but the wallet file is not changed appropriately, then the connections will fail.

3.3.7 Using Kerberos

Kerberos is a network authentication service for security in distributed environments. ODP.NET applications can use Kerberos for single sign-on and centralized user authentication. ODP.NET, Unmanaged Driver and Managed Driver both support Kerberos for external authentication to the database server.

3.3.7.1 File Based Credential Cache and MSLSA

ODP.NET supports both a file-based Kerberos client credential cache (CC) and the ability to use Windows logon credentials as Kerberos client credentials. The latter is called MSLSA-based Kerberos authentication.

In order to utilize a file based Kerberos client credential cache (CC), the following executables associated with the full Oracle Call Interface (OCI) install are needed:

- `okinit.exe`
- `oklist.exe`
- `okdstry.exe`

The executables are required in order to acquire the Kerberos5 credentials and store them in the file based credential cache (CC). However, after credential cache creation, as long as the credentials remain valid, the above executables are then unneeded by the ODP.NET application at run-time.

3.3.7.2 ODP.NET, Managed Driver Dependency on MIT Kerberos

To use Kerberos5 database authentication in conjunction with ODP.NET, Managed Driver, download and install MIT Kerberos for Windows 4.0.1 on the same machine as ODP.NET, Managed Driver from the following location:

`http://web.mit.edu/kerberos/dist/`

3.3.7.3 Configuring Kerberos Authentication with ODP.NET

Please reference the following "key" when viewing the below Kerberos configuration examples:

- `oracleclient` = Kerberos/Windows Domain user ID used by the Oracle database client program to represent the Oracle Client user on the domain
- `oracleserver` = Kerberos/Windows Domain user ID used by the Oracle database server
- `DOMAIN.COMPANY.COM` = Kerberos/Windows domain
- `dbhost.company.com` = Oracle database server machine hostname
- `kerberos_service_name` = Kerberos service name
- `dc.company.com` = hostname for Kerberos Key Distribution Center (KDC) and Windows Domain Controller

Configuring Kerberos Authentication Topics:

Step 1. Update Windows services file to include a "kerberos5" entry

Change the Kerberos entry in the Windows service file (`C:\windows\system32\drivers\etc\services`) from:

```
kerberos   88/tcp           krb5 kerberos-sec      #Kerberos
```

to:

```
kerberos   88/tcp kerberos5 krb5 kerberos-sec      #Kerberos
```

Step 2. Create client and server Kerberos users (Windows domain users for MSLSA)

As noted in the above "key", we will use `oracleclient` and `oracleserver` as our client and server Kerberos user IDs, respectively.

ODP.NET supports MSLSA using Windows domain users which have the following attributes:

- "Kerberos DES" unchecked
- "Kerberos AES 128 bit" checked
- "Kerberos AES 256 bit" checked
- "Kerberos preauthentication not required" checked

Step 3. Associate the DB server's Kerberos principal name with the DB server's Kerberos Service (SPN mapping) and generate the server keytab file

Run the following commands on the Kerberos KDC (Windows Domain Controller for MSLSA) as an administrator:

```
> ktpass -princ kerberos_service_name/dbhost.company.com@DOMAIN.COMPANY.COM /crypto
all /mapuser oracleserver@DOMAIN.COMPANY.COM /pass <oracleserver password> /out
v5srvtab

> setspn -A kerberos_service_name/dbhost.company.com@DOMAIN.COMPANY.COM oracleserver
```

Step 4. Confirm the mapping of server user to service principal

Also on the Kerberos KDC, run the following command, noting the output:

```
> setspn -L oracleserver

Registered ServicePrincipalNames for
CN=oracleserver,CN=Users,DC=domain,DC=company,DC=com:
        kerberos_service_name/dbhost.company.com
kerberos_service_name/dbhost.company.com@DOMAIN.COMPANY.COM
```

Step 5. Setup server sqlnet.ora to point to the keytab file generated in step 2

Add the following line to the server `sqlnet.ora`:

```
sqlnet.kerberos5_keytab = c:\krb\v5srvtab
```

Step 6. Create a kerberos configuration file that points to the Kerberos KDC (Windows Domain Controller for MSLSA)

An example kerberos configuration file (`krb.conf`):

```
[libdefaults]
default_realm = DOMAIN.COMPANY.COM

[realms]
DOMAIN.COMPANY.COM = {
  kdc = dc.company.com
 }

[domain_realm]
.domain.company.com = DOMAIN.COMPANY.COM
domain.company.com = DOMAIN.COMPANY.COM
```

```
.DOMAIN.COMPANY.COM = DOMAIN.COMPANY.COM
DOMAIN.COMPANY.COM = DOMAIN.COMPANY.COM
```

Step 7. Configure the Oracle database client and server sqlnet.ora or .NET config to point to the above Kerberos configuration file

Edit the client or server `sqlnet.ora` to include:

```
sqlnet.kerberos5_conf = C:\krb\krb.conf
```

Or edit the client application config to include (in the settings section):

```
<setting name="sqlnet.kerberos5_conf" value="C:\krb\krb.conf" />
```

Step 8. Point the client sqlnet.ora or .NET config to a credential cache file or to MSLSA

Example pointing to Credential Cache file:

```
sqlnet.kerberos5_cc_name = c:\krb\krb.cc
```

Example pointing to MSLSA:

```
sqlnet.kerberos5_cc_name = MSLSA:
```

Step 9. Set the client and server authentication services in the sqlnet.ora or .NET config to Kerberos5

```
sqlnet.authentication_services=(Kerberos5)
```

Step 10. Setup an externally authenticated database user that matches the Kerberos client user setup in step 1 (note the case)

```
create user "ORACLECLIENT@DOMAIN.COMPANY.COM" identified externally;
grant connect, create session to "ORACLECLIENT@DOMAIN.COMPANY.COM";
```

Step 11. Login to the client machine via the Windows Domain client user (for MSLSA) or perform an okinit to authenticate the client Kerberos user (for file based CC):

```
okinit oracleclient
```

Step 12. Run the ODP.NET application

Note:

- After configuring the client and server, the last 2 steps are the only steps required on an ongoing basis to run the ODP.NET application.
- A Microsoft Visual C Run-Time Library (MSVCRT.DLL) bug can cause ODP.NET, Managed Driver's setting of the Kerberos5 configuration to be ignored by the Microsoft run-time. In such a case, you will encounter the error message:

  ```
  OracleInternal.Network.NetworkException (0x80004005): NA Kerberos5: 
  Authentication handshake failure at stage: krb5_sname_to_principal: 
  default realm not found. Please set SQLNET.Kerberos5_conf.
  ```

  To workaround this error, manually set `KRB5_CONFIG` in the ODP.NET application's run-time environment to point to the Kerberos5 configuration file pointed to by `SQLNET.Kerberos5_conf`. For example,

  ```
  set KRB5_CONFIG=c:\oracle\network\admin\krb5.ini
  ```

3.3.8 Using Windows Native Authentication (NTS)

With the Windows native authentication adapter, Oracle users can authenticate to the database using just their Windows user login credentials. It provides a way to enable single sign-on and to simplify user and role credential management. Windows native authentication is also known as Windows Native authentication (NTS).

Note:

Due to a limitation in the Microsoft .NET APIs, ODP.NET, Managed Driver only supports Windows Native authentication (NTS) via Microsoft NT LAN Manager (NTLM) instead of Kerberos-based credentials. Normally, this limitation would be invisible to the ODP.NET, Managed Driver application, since the Windows domain and the Oracle database server will transparently support both NTLM and Kerberos domain credentials by default.

3.3.8.1 Configuring Windows Native Authentication (NTS) for the ODP.NET Client

Steps in configuring the NTS for the ODP.NET Client:

- Step 1. Ensure OSAUTH_PREFIX_DOMAIN is set correctly
- Step 2. Setup the externally identified database user
- Step 3. Setup the client configuration to utilize NTS as the authentication methodology

Step 1. Ensure OSAUTH_PREFIX_DOMAIN is set correctly

Make sure `OSAUTH_PREFIX_DOMAIN` is set appropriately. If you desire the externally identified user ID to include the domain, set it to true, otherwise false. The parameter is a registry setting that can be found at `HKLM/software/oracle/HOME<ORACLE_SID>`. For example, if your `ORACLE_SID` is `r1`, it is located at `HKLM/software/oracle/HOMEr1`.

Step 2. Setup the externally identified database user

Assuming a Step 0 setting of true, use the following commands to setup the externally identified database user associated with the desired Windows domain user:

```
create user "MYDOMAIN\MYUSER" identified externally;
grant connect, create session to "MYDOMAIN\MYUSER";
```

Step 3. Setup the client configuration to utilize NTS as the authentication methodology

Edit the client sqlnet.ora or app config to add NTS to the `sqlnet.authentication_services`. For example.

```
sqlnet.authentication_services = (NTS)
```

Note:

After configuring the client and server, the last 2 steps are the only steps required on an ongoing basis to run the ODP.NET application.

3.3.9 Network Data Encryption and Integrity

ODP.NET enables data encryption and integrity over a network for both intranet and cloud deployments. This ensures that data is disguised to all, except authorized users, and guarantees the original message contents are not altered. In earlier releases, these features were known as Oracle Advanced Security Option (ASO) encryption. Starting with Oracle Database 12*c*, Oracle ASO is not required to use network data encryption and data integrity.

3.3.9.1 Using Data Encryption

Managed and unmanaged ODP.NET support the following encryption standards and algorithms:

- Advanced Encryption Standard (AES)
  - AES 128-bit
  - AES 192-bit
  - AES 256-bit
- RSA RC4
  - 128-bit
  - 256-bit

- Triple-DES (3DES)
  - 112-bit
  - 168-bit

ODP.NET, Managed Driver uses the following settings to configure network encryption:

- `SQLNET.ENCRYPTION_CLIENT`
- `SQLNET.ENCRYPTION_TYPES_CLIENT`

3.3.9.2 Using Data Integrity

Managed and unmanaged ODP.NET support the following data integrity algorithms:

- MD5
- SHA-1
- SHA-2
  - SHA-256
  - SHA-384
  - SHA-512

3.3.10 Schema Discovery

ADO.NET 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:

Oracle Schema Collections

3.3.10.1 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.

   ```
   <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 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.

3.3.11 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.

```
// 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=100000;Connection Timeout=60;" +
      "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.");
  }
}
```

3.3.11.1 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.

When a new connection is opened, if the connection string is not an exact match to an existing pool, then a new pool is created. Prior to ODP.NET 12.1.0.2, only connection string attribute values had to match. Now, connection strings themselves must be an exact match. Keywords supplied in a different order for the same connection will be pooled separately. 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, except when Fast Connection Failover removes invalid connections or Connection Lifetime is exceeded. In these two cases, the connection number could drop below the Min Pool Size. ODP.NET would then attempt to restore the minimum pool size level upon the next connection request.

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 destroys the connection; otherwise, the connection goes back to the connection pool. The connection pooling service enforces the `Connection Lifetime` only when `Close()` or `Dispose()` is invoked.

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. Upon a connection timeout, ODP.NET distinguishes whether the timeout occurred due to the database server failing to deliver a connection in the allotted time or no connection being available in the pool due to the maximum pool size having been reached. The exception text returned will either be "Connection request timed out" in the case of the former or "Pooled connection request timed out" in the case of the latter.

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.

3.3.12 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:

Note:

These APIs are not supported in a .NET stored procedure.

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

3.3.13 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.

Oracle Data Provider for .NET enables or disables publishing performance counters for connection pooling, using registry entries.

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

Table 3-3 Performance Counters for Connection Pooling

Performance Counter	Valid Values	Description
None	`0`	Not enabled (Default)
`HardConnectsPerSecond`	`1`	Number of sessions being established with the Oracle Database every second.
`HardDisconnectsPerSecond`	`2`	Number of sessions being severed from the Oracle Database every second.
`SoftConnectsPerSecond`	`4`	Number of active connections originating from connection pools every second.
`SoftDisconnectsPerSecond`	`8`	Number of active connections going back to the connection pool every second.
`NumberOfActiveConnectionPools`	`16`	Total number of active connection pools.
`NumberOfInactiveConnectionPools`	`32`	Number of inactive connection pools.
`NumberOfActiveConnections`	`64`	Total number of connections in use.
`NumberOfFreeConnections`	`128`	Total number of connections available for use in all the connection pools.

Table 3-3 (Cont.) Performance Counters for Connection Pooling

Performance Counter	Valid Values	Description
`NumberOfPooledConnections`	`256`	Number of pooled active connections.
`NumberOfNonPooledConnections`	`512`	Number of non-pooled active connections.
`NumberOfReclaimedConnections`	`1024`	Number of connections which were garbage-collected implicitly.
`NumberOfStasisConnections`	`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.

3.3.13.1 Publishing Performance Counters

Publication of individual performance counters is enabled or disabled using the registry value `PerformanceCounters` of type `REG_SZ` or a .NET configuration file. 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.

3.3.13.2 Setting Performance Counters Using .NET Configuration Entry

Performance counters can be set using an .NET configuration entry. Since .NET configuration entries take precedence over the registry value setting, they can be used for a specific application.

An .NET configuration entry uses name/value pairs as in the following example:

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

3.3.13.3 Instance Names of Performance Counters

Performance counters can now monitor at the application domain, pool, or database instance level. Database instance level monitoring only applies if load balancing or Fast Connection Failover features are enabled.

The instance name format is as follows:

`<Application Domain Name> [<Process Id>, <Application Domain Id>][<Connection String/Pool Name>][<Instance Name>]`. The entry is limited to 127 characters. There is a restriction length on every field in the instance name. The following table shows the maximum number of characters allocated for each field:

Table 3-4 Field Names of Performance Counters and Maximum Number of Characters

Field Name	Maximum Number of Characters
Application Domain	40
Pool Name/Connection String	70
Database Instance Name	16

When the length of a field value exceeds the length limit, the string is truncated and appended with "`...`" to fit within the length limit and indicate the continuation. For example, for a given application called `Program.exe` with a connection string `user id=scott;Password=tiger;data source=inst1;max pool size=125`, one may see the following similar to the following for a process that has two application domains:

- `Program.exe [123, 1]`
- `Program.exe [123, 1][ user id=scott;data source=inst1;max pool siz...]`
- `Program.exe [123, 1][ user id=scott;data source=inst1;max pool siz...] [instA]`
- `Domain 2[123, 2]`
- `Domain 2[123, 2][ user id=scott;data source=inst1;max pool siz...]`
- `Domain 2[123, 2][ user id=scott;data source=inst1;max pool siz...] [instB]`
- `Domain 2[123, 2][ user id=scott;data source=inst1;max pool siz...] [instC]`

Since connection pool attributes can be similar in their first 70 characters, applications can set a Pool Name to uniquely identify each one in the monitoring tool. For example, when using Pool Name, the process will show up as follows:

```
Domain 2[123, 2][Pool Name][instC]
```

The .NET config file can set the Pool Name attribute.

ODP.NET, Managed Driver

```
<oracle.manageddataaccess.client>
 <version number="*">
  <connectionPools>
           .
           .
     <connectionPool connectionString="[connection string without password]"
poolName="[Pool Name]">   </connectionPool>
           .
           .
  </connectionPools>
 </version>
</oracle.manageddataaccess.client>
```

ODP.NET, Unmanaged Driver can use the same Pool Name attribute and format as listed above by replacing the `<oracle.manageddataaccess.client>` tags with `<oracle.unmanageddataaccess.client>` tags.

ODP.NET, Unmanaged Driver

```
<configuration>
 <oracle.dataaccess.client>
  <settings>
            .
            .
      <add name="[connection string without password]" value="connectionPool
name='[Pool Name]'"/>
           .
           .
    </settings>
  </oracle.dataaccess.client>
</configuration>
```

The behavior of two of the performance counters has now changed in the 12*c* release:

- `NumberOfPooledConnections` -- Sum of the active connections and free connections. Previously, this value was equal to just the number of active connections.
- `NumberOfStasisConnections` -- No longer supported.

3.3.14 Pluggable Databases

Oracle Database 12*c* introduced a new feature, Pluggable Databases, which enable an Oracle database to contain a portable collection of schemas, schema objects, and nonschema objects that appears to ODP.NET as a separate database. This self-contained collection is called a pluggable database (PDB).

ODP.NET 12*c* and higher can connect to PDBs, which clients access through database services. Database services have an optional PDB property. When a PDB is created, a new default database service is created automatically. The service has the same name as the PDB and can be used to access the PDB using the easy connect syntax or the net service name. This service is intended primarily for performing administrative tasks. It is recommended that you create additional services for use in your applications.

All ODP.NET features can be used with PDBs with the following exceptions:

- Continuous Query Notification
- Switching from one PDB to another PDB using the `ALTER SESSION SET CONTAINER` statement

3.3.15 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.

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 .NET configuration file for ODP.NET, Unmanaged Driver:

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
 <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.
- ODP.NET, Managed Driver does not support Edition-Based Redefinition.

3.3.16 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.

Note:

Operating System Authentication is not supported in a .NET stored procedure.

All ODP.NET, Unmanaged Driver connections, including those using operating system authentication, can be pooled. ODP.NET, Managed Driver supports operating system authentication, except when the Windows domain is constrained to only support Kerberos-based domain authentication. 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:

```
/* 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
*/

// C#

using System;
using Oracle.DataAccess.Client;

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

    //Establish connection using OS Authentication
    con.ConnectionString = "User Id=/;Data Source=oracle;";
    con.Open();
    Console.WriteLine("Connected to Oracle" + con.ServerVersion);

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

3.3.17 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");
  }
}
```

3.3.18 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`:

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


// C#

using System;
using Oracle.DataAccess.Client;

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

    try
    {
      con.ConnectionString =
        "User Id=testexpire;Password=testexpire;Data Source=oracle";
      con.Open();
      Console.WriteLine("Connected to Oracle" + con.ServerVersion);
    }
    catch (OracleException ex)
    {
      Console.WriteLine(ex.Message);

      //check the error number
      //ORA-28001 : the password has expired
      if (ex.Number == 28001)
      {
        Console.WriteLine("\nChanging password to panther");
        con.OpenWithNewPassword("panther");
        Console.WriteLine("Connected with new password.");
      }
    }
    finally
    {
      // Close and Dispose OracleConnection object
      con.Close();
      con.Dispose();
      Console.WriteLine("Disconnected");
    }
  }
}
```

Note:

- 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.

3.3.19 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 User Id"` 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:

```
/*  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;
*/

// C#

using System;
using Oracle.DataAccess.Client;

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

    // Connecting using proxy authentication
    con.ConnectionString = "User Id=scott;Password=tiger;" +
      "Data Source=oracle;Proxy User Id=appserver;Proxy Password=eagle; ";
    con.Open();
    Console.WriteLine("Connected to Oracle" + con.ServerVersion);

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

3.3.20 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 `"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`" is equal to "`false`" or "`dynamic`", the connection cannot enlist in the transaction. ODP.NET, Unmanaged Driver in ODAC 12c Release 3 first introduced this new behavior for "`Enlist=dynamic`".

3.3.21 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`
- `Client info` - Specifies user session information
- `Module` - Specifies a functional block, such as Accounts Receivable or General Ledger, of an application

3.3.22 Transparent Application Failover (TAF) Callback Support

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

> **Note:**
>
> ODP.NET, Managed Driver does not support TAF nor TAF callbacks.

TAF enables an application connection to automatically reconnect to another database instance if the connection gets severed. Active transactions roll back, but the new database connection, made by way of a different node, is identical to the original. This is true regardless of how the connection fails.

With 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.

3.3.22.1 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.

3.3.22.2 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. No significant database operation should occur immediately after a `FailoverEvent.Begin` event. SQL and major database operations should wait until the `FailoverEvent.End` event. `FailoverEvent.Begin` is primarily used to reject failover or to trace it. `FailoverEvent.Begin` can also be used for non-database application operations, such as informing the end user a failover is in progress and to wait until it completes before proceeding. Transactions can be used in the `FailoverEvent.End` callback phase, such as to file fault tickets or audit. These transactions must be committed before the callback completes.

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.

3.3.22.3 Registering an Event Handler for Failover

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

```
// C#

using System;
using Oracle.DataAccess.Client;

class TAFCallBackSample
{
  public static FailoverReturnCode OnFailover(object sender,
                                              OracleFailoverEventArgs eventArgs)
  {
    switch (eventArgs.FailoverEvent)
    {
      case FailoverEvent.Begin :
        Console.WriteLine(
          " \nFailover Begin - Failing Over ... Please standby \n");
        Console.WriteLine(
          " Failover type was found to be " + eventArgs.FailoverType);
        break;

      case FailoverEvent.Abort :
        Console.WriteLine(" Failover aborted. Failover will not take place.\n");
        break;

      case FailoverEvent.End :
        Console.WriteLine(" Failover ended ...resuming services\n");
        break;

      case FailoverEvent.Reauth :
        Console.WriteLine(" Failed over user. Resuming services\n");
        break;

      case FailoverEvent.Error :
        Console.WriteLine(" Failover error gotten. Sleeping...\n");
        return FailoverReturnCode.Retry;

      default :
        Console.WriteLine("Bad Failover Event: %d.\n", eventArgs.FailoverEvent);
        break;
    }
    return FailoverReturnCode.Success;
  } /* OnFailover */

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

    con.ConnectionString = "User Id=scott;Password=tiger;Data Source=oracle;";
    con.Open();
    con.Failover += new OracleFailoverEventHandler(OnFailover);
    Console.WriteLine("Event Handler is successfully registered");

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

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.

3.4 Real Application Clusters and Global Data Services

This section discusses optimizations for the following products:

- Oracle Real Application Clusters (Oracle RAC) is a cluster database with a shared cache architecture that overcomes the limitations of traditional shared-nothing and shared-disk approaches to provide highly scalable and available database solutions for business applications.
- Oracle Data Guard provides one or more standby databases to protect Oracle data from failures, disasters, human error, and data corruptions for high availability in mission critical applications.
- Oracle GoldenGate replicates data among heterogeneous data environments. It enables high availability solutions, real-time data integration, transactional change data capture, data replication, transformations, and verification between operational and analytical enterprise systems.
- Global Data Services (GDS), new in Oracle Database 12*c*, provides database workload management features across replicated databases, such as Data Guard and GoldenGate.

ODP.NET supports Oracle Real Application Clusters (Oracle RAC), Data Guard, and GoldenGate transparently, meaning you do not need to change ODP.NET code to use these Oracle components. To further take advantage of these technologies, ODP.NET offers connection pooling optimization features for achieving better application high availability and performance. You can do this through configuring ODP.NET to receive, respond, and send database status messages to .NET applications.

These optimization configurations include the use of features such as Fast Application Notification (FAN), Runtime Connection Load Balancing, and Fast Connection Failover (FCF).

These connection pooling optimizations can improve high availability and performance for Oracle Real Application Clusters and Global Data Services products:

- Fast Application Notification
- Runtime Connection Load Balancing
- Fast Connection Failover (FCF)

3.4.1 Fast Application Notification

Fast Application Notification (FAN) is a high availability and load balancing notification mechanism that Oracle RAC, Data Guard, and GoldenGate use to notify ODP.NET applications about cluster configuration and service-level information, including status changes such as UP or DOWN events and server load. FAN UP and DOWN events can apply to instances, services, and nodes. Based on information received, ODP.NET can adjust its connection pool accordingly to improve application availability and performance.

With FAN, Oracle RAC, Data Guard, and GoldenGate use one of two Oracle messaging infrastructures to send notifications to ODP.NET applications:

- Oracle Notification Service (ONS)
- Oracle Streams Advanced Queueing (AQ).

Table 3-5 describes when each messaging system is used and the ODP.NET-related client configuration.

Table 3-5 Configurations for ODP.NET Driver Types

ODP.NET Driver Type	Database Server Version	FAN Infrastructure	Configuration	Manual ONS Configuration Locations
managed	12.1 and later	ONS	Automatic or Manual	Either of these two files: • .NET configuration file • ONS configuration file
managed	11.2 and earlier	ONS	Manual	Either of these two files: • .NET configuration file • ONS configuration file
unmanaged	12.1 and later	ONS	Automatic or Manual	`oraaccess.xml` file
unmanaged	11.2 and earlier	AQ	Automatic	N/A

For automatic ONS configuration, developers can add more nodes and ports for ODP.NET to listen to, in addition to the nodes and ports that ODP.NET obtains from the database automatically.

ODP.NET applications do not require code changes to migrate from the AQ to ONS FAN infrastructure. However, some ODP.NET client configuration changes may be necessary when migrating to ONS, a newer database server version, or from ODP.NET, Unmanaged Driver to the managed driver, as documented above.

On the database server side, FAN must be set up and configured.

Using FAN Messages from the database, ODP.NET can do the following:

- With Runtime Connection Load Balancing, ODP.NET load balances connections among Oracle RAC nodes, services, and service members and GDS resources. This feature improves ODP.NET response time and ensures better resource allocation of server resources.
- With the Fast Connection Failover (FCF) feature, Oracle RAC, Data Guard, and GoldenGate can inform the ODP.NET connection pool if database nodes, services, service members, or the databases have gone down. These DOWN messages indicate which connections in the pool are invalid and must be removed.

3.4.2 Runtime Connection Load Balancing

With Runtime Connection Load Balancing, Oracle Data Provider for .NET balances work requests across Oracle RAC instances based on the load balancing advisory and service goal. Because workloads can constantly change, load balancing occurs when the application requests a new connection. Thus, ODP.NET optimizes service levels by connecting users to the least loaded nodes in real-time.

In Oracle Database 12*c*, Runtime Connection Load Balancing has been extended to Oracle Data Guard and Oracle GoldenGate so that ODP.NET 12*c* connections can be load balanced with these two database services as part of Global Data Services. No ODP.NET applications require code changes to use Global Data Services if they are already using 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, ODP.NET is enabled to receive Runtime Connection Load Balancing FAN messages from the server. The feature has been enabled via the "`Load Balancing=true`" and "`pooling=true`" settings in the connection string, which are the default values. This feature can only be used if "`pooling=true`". In order to use Runtime Connection Load Balancing, specific Oracle server configurations must be set.

The following connection string example enables Runtime Connection Load Balancing:

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

3.4.3 Fast Connection Failover (FCF)

When an Oracle RAC service, service member, node, or a Data Guard database fails, the severed ODP.NET connection objects may continue to exist in the application. If users attempt to use these invalid connections, they will encounter errors. FCF enables ODP.NET to free these severed connections proactively and quickly. Users then will be able to use the application after a server side failure without manual intervention from an administrator.

In Oracle Database 12*c*, FCF has been extended to Oracle Data Guard and Oracle GoldenGate for ODP.NET 12*c* connections through Global Data Services. No

ODP.NET applications require code changes to use Global Data Services if they already use FCF.

ODP.NET applications can enable FCF through the High Availability Events, `"HA Events"`, connection string attribute. When HA Events are enabled:

- ODP.NET connection pool proactively removes connections from the pool when a Global Data Service or Oracle RAC service, service member, node, or database goes down.
- ODP.NET proactively forces threads waiting for responses from the downed database to exit out from the existing call to avoid any hangs. When such a connection is then returned to the pool, any resource associated with that connection is freed.
- ODP.NET establishes connections to existing Oracle instances if the removal of severed connections brings the total number of connections below the "`min pool size`", upon the next connection request.

By default, ODP.NET is enabled to receive FCF FAN messages from the server. This feature have been enabled via the `HA Events=true` and `pooling=true` settings in the connection string, which are the default values.

The following connection string example enables HA Events:

```
"user id=scott;password=tiger;data source=erp;HA events=true;"
```

3.4.4 Using FCF Planned Outage to Minimize Service Disruption

FCF not only provides high availability services for unplanned outages, such as node failures, but also for planned outages, such as server repairs, upgrades, and changes, to minimize service disruption to ODP.NET application users.

When a database service is set to be stopped or relocated, a FAN message is published with a planned reason code. A FCF-aware ODP.NET connection pool (`HA Events=true`) receives the notification and commences to close idle connections, no longer allowing new connections to that specific database service. Active connections to that specific database service remain until users complete their tasks and the connection is returned to the pool. Thus, no users must stop work mid-stream due to a planned outage.

Eventually, all users complete their tasks and no connections remain to that database service. The database administrator can then stop the service for the planned outage task. This feature allows the database service to be stopped as quickly as possible without end user disruption.

Oracle planned outage support works with Oracle Real Application Clusters (Oracle RAC), Oracle Data Guard, and some single instance scenarios.

Oracle RAC Planned Outage

A typical planned outage scenario for Oracle RAC follows below. Note that the database server commands apply to Oracle RAC 12*c* Release 2 or higher. Commands for earlier releases may be different.

1. There is a need to upgrade, patch, or repair a software or hardware issue on a database server. Stop the instance gracefully such that existing users experience no to few errors. You can wait until all users complete their work before doing so.

Business requirements will dictate whether you wait for all users to log out or begin the planned outage after a set time. An administrator could issue the following command line operation using Oracle Server Control Utility (`srvctl`):

```
srvctl relocate service -database <unique database name> -service <service name>
-drain_timeout 120 -stopoption IMMEDIATE -oldinst <existing instance>
```

This command relocates the database service from the existing instance to any instance it is configured to run on. Oracle Cluster Ready Services (CRS) will choose this instance, as the command line specifies no target. CRS will wait 120 seconds (`-drain_timeout 120`) for any active sessions to drain, after which any sessions remaining on the existing instance will be forcibly disconnected (`-stopoption IMMEDIATE`). If Application Continuity is used in conjunction with planned outage, an attempt is made to recover these killed sessions, masking the outage from end users.

The relocate operation starts the service in the new location prior to stopping the service in its existing location. Immediate relocation allows draining with no brownout. If the service cannot be started, it is not stopped at the original location to maintain availability.

2. Meanwhile in the connection pool, the FAN planned DOWN event clears idle sessions for the instance being shutdown from the ODP.NET connection pool immediately and marks that instance's active sessions to be released at the next check-in. These FAN actions drain the sessions from this instance without disrupting the users.

   Existing connections on other instances remain usable, and new connections can be opened to these other instances.

3. Not all sessions will check their connections into the pool immediately. The timeout period specified by `-drain_timeout` after which the instance is forcibly shut down, evicting any remaining client connections. Administrators can check whether any active sessions to the instance remain by querying the `v$session` table.

4. Once the upgrade, patch, or repair is complete, restart the instance and the service on the original node. The FAN UP event will inform the ODP.NET pool that it can now use the original machine again.

See Also:

Oracle Database High Availability Best Practices

Oracle Data Guard Planned Outage

Oracle Data Guard performs switchovers from primary databases to standby databases in planned failover scenarios. During the switchover, administrators will want to limit end user disruptions. In Oracle Database 12*c* Release 2 and higher, these administrators can use the Data Guard command-line interface (`DGMGRL`) command to switch roles between primary and standby databases:

```
SWITCHOVER TO <database name> [WAIT <timeout in seconds> ];
```

The `WAIT` option specifies to wait for sessions to drain before proceeding with the switchover.

Similar to the Oracle RAC scenario, FAN informs the ODP.NET to remove idle connections from the pool. Connections subsequently checked in are destroyed until no active connections remain to that primary database, which will allow the switchover to begin.

When switchover to the standby completes, a FAN UP event informs ODP.NET that it can start creating connections to the standby instance.

During the Data Guard service relocation process, new incoming connection requests will not be accepted until the service has fully relocated. Incoming connection requests arriving during the interim, such as in the middle of an Oracle Data Guard switchover, will receive connectivity errors.

To prevent these errors, ODP.NET can pause connection attempts until the new database service is available. ODP.NET, Managed and Unmanaged Drivers block any connection attempts until the service is up or until the configured time limit expires from the time when the service DOWN event was received. This feature is useful for planned outages and service relocations. It works with Oracle RAC and Oracle Data Guard.

This time limit is the `ServiceRelocationConnectionTimeout` setting, which can be set in the .NET configuration file.

3.4.5 Pool Behavior 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`*, then 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`

ODP.NET connects to the same Oracle RAC node when required by a distributed transaction that has already begun on a particular node, by an Oracle runtime connection load balancing advisory, or by Oracle RAC load balancing gravitation in which connections will gravitate to an under utilized node. If the connection pool has no idle connections to this particular node, then ODP.NET will create a new connection to this node. Node affinity is honored even when the connection pool runs out of idle connections to dispense.

3.5 Using Transaction Guard to Prevent Logical Corruption

Transaction Guard allows managed and unmanged ODP.NET applications to use at-most-once execution in case of planned and unplanned outages and repeated submissions. Without Transaction Guard, applications that attempt to retry operations following outages can cause logical corruption by committing duplicate transactions.

After an outage, one of the traditional problems for recovering applications had been the non-durable commit message sent back to the client. If there is a break between the client and the server, the client sees an error message indicating that the communication failed, also known as a recoverable error. This error does not inform the application if the submission executed any commit operations, or if a procedural call ran to completion while executing all expected commits. The error also does not indicate session state changes or intermittent failures. The client is left wondering if the transaction committed and if it fully completed.

These recoverable errors may require end users or applications to attempt replay by issuing duplicate transaction submissions or other forms of logical corruption. The transaction cannot be validly resubmitted if the non-transactional state is incorrect or if it is committed. Continuing to process a committed but not completed call can result in the application using a database session that is in the wrong state.

3.5.1 ODP.NET and Transaction Guard

Transaction Guard allows ODP.NET, Managed Driver and ODP.NET, Unmanaged Driver to eliminate duplicate transactions automatically and transparently, and in a manner that scales.

When a failure occurs, such as a node, network, or database failure, ODP.NET applications can deterministically conclude whether the transaction committed by querying its status, if the database service is up. Oracle retains the transaction status automatically, even after one of these failures.

In ODAC 12*c* Release 4, using Transaction Guard application development has been streamlined, reducing the application logic needed to determine the transaction outcome. Moreover, these benefits are available to both managed and unmanaged ODP.NET.

When a recoverable error is raised by a Transaction Guard enabled database service upon a database commit or upon a SQL or PL/SQL execution, which could have called a commit, then an ODP.NET `OracleException` is created with an `OracleLogicalTransaction` instance. The database maintains the outcome of the logical transaction for the retention period specified by the administrator. ODP.NET automatically queries the database on behalf of the application when a recoverable error occurs so that the `OracleLogicalTransaction` object instance on the `OracleException` object can indicate whether the transaction has committed or not and whether the user call has completed or not.

If the status is committed, then the transaction has completed successfully. No other action is likely needed by the administrator.

If not committed, then ODP.NET applications can learn the current transaction state, whether it is recoverable, and whether it can be retried using `OracleLogicalTransaction`. If the error is recoverable, then the transaction is safe to re-submit. If the error is not recoverable, the application will need to determine the transaction outcome using an alternative mechanism.

Note:

Transaction Guard supports only local transactions. It does not support distributed transactions.

The Transaction Guard feature is enabled or disabled through the Oracle service-level configuration through the `COMMIT_OUTCOME` setting. By default, it is not enabled. This setting can be changed without bringing down the database. Only new connections created against the service will use the new setting.

Here's an example of setting the `COMMIT_OUTCOME` using `SRVCTL`:

```
srvctl modify service -d orcl -s GOLD -commit_outcome TRUE
```

Note:

Grant the `EXECUTE` privilege on the `DBMS_APP_CONT` package to the database users that retrieve the transaction status:

```
GRANT EXECUTE ON DBMS_APP_CONT TO <user name> ;
```

The following is an example ODP.NET Transaction Guard application scenario:

An ODP.NET application receives a Fast Application Notification (FAN) down event or error. FAN automatically aborts the dead session and the application receives an `OracleException`. A Transaction Guard application built to handle errors transparently would do the following:

1. Check the value of the `OracleException.OracleLogicalTransaction` property. If the value is an `OracleLogicalTransaction` object, that is, non-null, then the error is recoverable. If the property's value is null, then the error is not recoverable and/or Transaction Guard has not been enabled.
2. For recoverable errors, check the `OracleLogicalTransaction.Committed` property. If `true`, the transaction has been committed. If `false`, the transaction was not submitted, but can now be safely re-submitted.
3. For recoverable errors, check the `OracleLogicalTransaction.UserCallCompleted` property if transaction state outside the commit operation is important. See the table below for the implications of what `Committed` and `UserCallCompleted` values mean.

Table 3-6 Implication of Committed and UserCallCompleted Values

Committed Value	UserCallCompleted Value	Outcome
`True`	`True`	The transaction was successful. The result can be returned to the application.
`False`	`False`	The transaction was not successful. The application can resubmit the transaction again.
`True`	`False`	The transaction committed, but there may be additional state, such as row counts or nested PL/SQL logic, that prevents the application from continuing as expected.

Sample Code

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

```
//alternatively can use using Oracle.ManagedDataAccess.Client;

class TransactionGuardSample
{
    static void Main()
    {
        bool bReadyToCommit = false;

        string constr = "user id=hr;password=hr;data source=oracle";
        OracleConnection con = new OracleConnection(constr);
        OracleTransaction txn = null;
        OracleCommand cmd = null;

        try
        {
            string sql = " update employees set salary=10000 where employee_id=103";
            con.Open();
            txn = con.BeginTransaction();
            cmd = new OracleCommand(con, sql);
            cmd.ExecuteNonQuery();
            bReadyToCommit = true;
        }
        catch (Exception ex)
        {
            // rollback here as the SQL execution is unsuccessful
            txn.Rollback();
            Console.WriteLine(ex.ToString());
        }

        try
        {
            if (bReadyToCommit)
                txn.Commit();
        }
        catch (Exception ex)
        {
            if (ex is OracleException)
            {
                //  It's safe to re-submit the work if the error is recoverable and
the transaction has not been committed
                if (ex.IsRecoverable && ex.OracleLogicalTransaction != null && !
ex.OracleLogicalTransaction.Committed)
                {
                    // safe to re-submit work
                }
                else
                {
                    // do not re-submit work
                }
            }
        }
        finally
        {
            // dispose all objects
            txn.Dispose();
            cmd.Dispose();
            con.Dispose(); // place the connection back to the connection pool
        }
    }
}
```

3.6 Application Continuity

Oracle Application Continuity enables database requests to automatically replay transactional or non-transactional operations in a non-disruptive and rapid manner in the event of a severed database session, which results in a recoverable error. Application Continuity improves end-user experience by masking planned and unplanned related errors. Applications can be developed without complex logic to handle exceptions, while automatically replaying database operations upon a recoverable error.

Without Application Continuity, it is almost impossible to mask outages in a safe and reliable manner. Common issues encountered include:

- The client state remains at present time, with entered data, returned data, and variables cached, while the database state changes are lost.
- If a transaction commit has occurred, the commit message is not durable. Moreover, checking a lost request does not guarantee that it will not commit after being checked.
- Non-transactional database session state is lost.
- If the request can continue, the database and the client session must be synchronized.

Application Continuity is available with Oracle Database Enterprise Edition with a Real Application Clusters or Active Data Guard option license.

3.6.1 ODP.NET and Application Continuity

ODP.NET, Unmanaged Driver first supported Application Continuity with version 12.2. While Application Continuity was first introduced in Oracle Database 12*c* Release 1 (12.1), ODP.NET requires a minimum of Oracle Database 12*c* Release 2 (12.2) server.

Note:

ODP.NET, Managed Driver does not support Application Continuity.

With Application Continuity enabled, ODP.NET ensures all the application's executed statements are logged appropriately so that they can be replayed upon a recoverable error. This applies for all application SQL and PL/SQL, as well as any internal ODP.NET operations.

On the client side, Application Continuity is enabled by setting the ODP.NET connection string attribute, `Application Continuity=true`.

If `Application Continuity` is set to `true`, but the database server does not enable `Application Continuity`, ODP.NET will still create new connections. However, these connections will not be `Application Continuity` enabled.

3.7 Database Sharding

Sharding is a data tier architecture, where data is horizontally partitioned across independent databases. Each database in such a configuration is called a shard. All shards together make up a single logical database, which is referred to as a sharded database. Sharding is a shared-nothing database architecture. The independent physical databases do not share CPU, memory, or storage devices. However, from the perspective of an application, the collection of physical databases looks like a single logical database.

Sharding uses Global Data Services (GDS), where GDS routes a client request to an appropriate database based on parameters such as availability, load, network latency, and replication lag. A GDS pool is a set of replicated databases that offers the same global service. The databases in a GDS pool can be located in multiple data centers across different regions. A sharded GDS pool contains all shards of a sharded database and their replicas, and appears as a single sharded database to database clients.

Applications can connect to multiple databases (shards) where data is partitioned based on one or more sharding strategies. The strategy can be hash based, range based, or list based. Each time a database operation is required, the application needs to determine which shard it must connect to.

A sharding key provides the partitioning key that determines in which shard a row of data is stored. A table can be partitioned using a sharding key.

A super sharding key is a collection of shard chunks, where only those chunks, which have a specific value of the super shard key identifier, are stored. A super sharding key is used for distributing data across database groups. Specifying super sharding keys are a way through which user-controlled data partitioning is possible.

3.7.1 ODP.NET Sharding

Starting from version 12.2, ODP.NET and Oracle Database both support sharding.

Note:

ODP.NET, Managed Driver does not support sharding.

ODP.NET applications must provide the sharding key and super sharding key information before opening the database connection for single shard queries. These sharding values cannot be set or changed after opening the connection. If any of the shard key values need to be modified, a new connection must be created with the new values and then opened.

If shard keys are set after the connection has been opened, the ODP.NET connection will not use these new shard key values until after the next `OracleConnection.Open()` call.

The `OracleShardingKey` object stores one or more key values. Multiple keys can be set to create a composite key. ODP.NET recognizes the sharding key(s) specified and connects to the correct shard and chunk.

Sharding is supported with or without connection pooling. The ODP.NET connection pool maintains connections to different shards and chunks of the sharded GDS database within the same shared pool.

The shard key (`SHARD_KEY`) and super sharding key (`GROUP_KEY`) can be specified in the TNS connect descriptor, rather than in the application code. The .NET developer then chooses the connect descriptor applicable to the shard that the application will use.

The data distribution across the shards and chunks in the database is transparent to the end user. ODP.NET minimizes the end user impact of chunk resharding within GDS.

To perform cross-shard queries, no ODP.NET shard APIs are used. Instead, applications connect to the GDS catalog service, allowing access to all the sharded databases. The SQL query is specifically constructed to iterate over all the necessary shards. For example, the non-shard database query `select count(*) from employees` is equivalent to the cross- shard query `select sum(c) from (Iterator(select count(*) c from employees(i))`.

ODP.NET Single Shard Query Example

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

class Sharding
{
  static void Main()
  {
    OracleConnection con = new OracleConnection("user id=hr;password=hr;Data 
Source=orcl;");
    //Setting a shard key
    OracleShardingKey shardingKey = new OracleShardingKey(OracleDbType.Int32, 123);
    //Setting a second shard key value for a composite key
    shardingKey.SetShardingKey(OracleDbType.Varchar2, "gold");
    //Creating and setting the super shard key
    OracleShardingKey superShardingKey = new OracleShardingKey();
    superShardingKey.SetShardingKey(OracleDbType.Int32, 1000);

    //Setting super sharding key and sharding key on the connection
    con.SetShardingKey(shardingKey, superShardingKey);
    con.Open();

    //perform SQL query
  }
}
```

3.8 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
- System.Transactions and Promotable Transactions
- Parameter Binding
- Batch Processing
- Statement Caching
- Self-Tuning

3.8.1 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.

3.8.2 System.Transactions and Promotable Transactions

ODP.NET supports `System.Transactions`. A local transaction is created for the first connection opened in the `System.Transactions` scope to Oracle Database 11*g* 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 11*g* 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.

Transaction promotion will throw an ORA-24797 error when the database transaction is already distributed due to the use of database links.

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. Starting with ODP.NET 12.1.0.2, connections with the `Promotable Transaction` setting set to `local` will begin as and remain a local transaction. If a second connection attempts to join the transaction, an exception will be thrown.

If applications use `System.Transactions`, it is required that the `enlist` connection string attribute is set to either `true` (default) or `dynamic`. However, `enlist=dynamic` cannot be used with `TransactionScope` because auto-enlistment requires `enlist=true`.

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

3.8.2.1 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 11*g* 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". 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.

```
// 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);
      }
    }
  }
```

3.8.2.2 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 commitable transaction.
      conn1.EnlistTransaction(cmtTx);

      // 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);
    }
  }
}
```

3.8.2.3 Local Transaction Support for Older Databases

If the first connection in a `TransactionScope` is opened to a pre-Oracle Database 11*g* release 1 (11.1) instance, then the connection creates a distributed transaction, by default. You can optionally have the first 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.

3.8.3 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 `DbType.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:

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

3.8.3.1 Command Timeouts

The OracleCommand CommandTimeout property limits how long a command is allowed to execute before terminating with an exception. This setting prevents long running commands from consuming excessive resources or from blocking other necessary operations from occurring.

The database server can be interrupted via either TCP/IP urgent data or normal TCP/IP data, called out of band (OOB) or in band data, respectively. Windows-based database servers only support in band breaks, whereas all other (predominantly UNIX-based) database servers can support OOB or in band breaks.

ODP.NET, Managed Driver uses OOB breaks by default with database servers that support it. For certain network topologies, the routers or firewalls involved in the route to the database may have been configured to drop urgent data or in band the data. If the routers or firewalls can not be changed to handle urgent data appropriately, then the ODP.NET, Managed Driver can be configured to utilize in band breaks by setting the .NET configuration parameter Disable_Oob to on.

3.8.3.2 OracleDbType Enumeration Type

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

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

Table 3-7 OracleDbType Enumeration Values

Member Name	Description
Array	Oracle Collection (VArray or Nested Table) *Not Available in ODP.NET, Managed Driver*
BFile	Oracle BFILE type
BinaryFloat	Oracle BINARY_FLOAT type
BinaryDouble	Oracle BINARY_DOUBLE type
Blob	Oracle BLOB type
Boolean	Oracle BOOLEAN type
Byte	byte type
Char	Oracle CHAR type

Table 3-7 (Cont.) OracleDbType Enumeration Values

Member Name	Description
`Clob`	Oracle `CLOB` type
`Date`	Oracle `DATE` type
`Decimal`	Oracle `NUMBER` type
`Double`	8-byte `FLOAT` type
`Int16`	2-byte `INTEGER` type
`Int32`	4-byte `INTEGER` type
`Int64`	8-byte `INTEGER` type
`IntervalDS`	Oracle `INTERVAL DAY TO SECOND` type
`IntervalYM`	Oracle `INTERVAL YEAR TO MONTH` type
`Long`	Oracle `LONG` type
`LongRaw`	Oracle `LONG RAW` type
`NChar`	Oracle `NCHAR` type
`Object`	Oracle Object type *Not Available in ODP.NET, Managed Driver*
`NClob`	Oracle `NCLOB` type
`NVarchar2`	Oracle `NVARCHAR2` type
`Raw`	Oracle `RAW` type
`Ref`	Oracle `REF` type *Not Available in ODP.NET, Managed Driver*
`RefCursor`	Oracle `REF CURSOR` type
`Single`	4-byte `FLOAT` type
`TimeStamp`	Oracle `TIMESTAMP` type
`TimeStampLTZ`	Oracle `TIMESTAMP WITH LOCAL TIME ZONE` type
`TimeStampTZ`	Oracle `TIMESTAMP WITH TIME ZONE` type
`Varchar2`	Oracle `VARCHAR2` type
`XmlType`	Oracle `XMLType` type

Note:

PL/SQL `LONG`, `LONG RAW`, `RAW`, and `VARCHAR` data types can be bound with a size up to 32512 bytes.

3.8.3.3 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.

3.8.3.3.1 Inference of DbType from OracleDbType

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

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

OracleDbType	System.Data.DbType
Array	Object
BFile	Object
Blob	Object
BinaryFloat	Single
BinaryDouble	Double
Boolean	Boolean
Byte	Byte
Char	StringFixedLength
Clob	Object
Date	Date
Decimal	Decimal
Double	Double
Int16	Int16
Int32	Int32
Int64	Int64
IntervalDS	Object
IntervalYM	Int64
Long	String
LongRaw	Binary
NChar	StringFixedLength
NClob	Object
NVarchar2	String
Object	Object
Raw	Binary
Ref	Object
RefCursor	Object
Single	Single
TimeStamp	DateTime
TimeStampLTZ	DateTime
TimeStampTZ	DateTime
Varchar2	String

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

OracleDbType	System.Data.DbType
`XmlType`	`String`

3.8.3.3.2 Inference of OracleDbType from DbType

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

Table 3-9 Inference of OracleDbType from DbType

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

3.8.3.3.3 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 3-10 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 3-10 Inference of DbType and OracleDbType from Value (.NET Datatypes)

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

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 3-11 shows the inference of DbType and OracleDbType properties from the Value property when type of Value is one of Oracle.DataAccess.Types.

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

Value (Oracle.DataAccess.Types)	System.Data.DbType	OracleDbType
OracleBFile	Object	BFile
OracleBinary	Binary	Raw
OracleBlob	Object	Blob
OracleBoolean	Boolean	Boolean
OracleClob	Object	Clob

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

Value (Oracle.DataAccess.Types)	System.Data.DbType	OracleDbType
`OracleDate`	`Date`	`Date`
`OracleDecimal`	`Decimal`	`Decimal`
`OracleIntervalDS`	`Object`	`IntervalDS`
`OracleIntervalYM`	`Int64`	`IntervalYM`
`OracleRef`	`Object`	`Ref`
`OracleRefCursor`	`Object`	`RefCursor`
`OracleString`	`String`	`Varchar2`
`OracleTimeStamp`	`DateTime`	`TimeStamp`
`OracleTimeStampLTZ`	`DateTime`	`TimeStampLTZ`
`OracleTimeStampTZ`	`DateTime`	`TimeStampTZ`
`OracleXmlType`	`String`	`XmlType`

3.8.3.4 PL/SQL Associative Array Binding

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.

  Each ODP.NET array element can store up to 2 GB of characters per element or 4 GB of binary data per element

- `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

```
/* 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\n
       Param3(1) := Param2(1);
       Param3(2) := NULL;
       Param3(3) := Param2(3);
       -- copy all elements from Param1 to Param2\n
       Param2(1) := Param1(1);
```

```
        Param2(2) := Param1(2);
        Param2(3) := Param1(3);
        -- insert some values to db\n
        FOR i IN 1..3 LOOP
          insert into T1 values(i,Param2(i));
        END LOOP;
      END TestVarchar2;
    END MYPACK;
/
 */

// C#

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

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

    con.ConnectionString = "User Id=scott;Password=tiger;Data Source=oracle";
    con.Open();
    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
    Param1.Value = new string[3] {
      "First Element", "Second Element ", "Third Element "
    };
    Param2.Value = new string[3] {
      "First Element", "Second Element ", "Third Element "
    };
    Param3.Value = null;

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

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

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

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

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

    // execute the cmd
    cmd.ExecuteNonQuery();

    //print out the parameter's values
    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");
  }
}
```

3.8.3.5 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
    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();
  }
}
```

3.8.3.5.1 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.

3.8.3.5.2 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:

```
/* 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 ArrayBindCount to indicate the number of values
      cmd.ArrayBindCount = 3;

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

      prm.Direction = ParameterDirection.Input;
      prm.Value = myArrayDeptNo;
```

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

          // Execute the command
          cmd.ExecuteNonQuery();
        }
        catch (OracleException e)
        {
          Console.WriteLine("OracleException {0} occured", e.Message);
          if (e.Number == 24381)
            for (int i = 0; i < e.Errors.Count; i++)
              Console.WriteLine("Array Bind Error {0} occured at Row Number {1}",
                e.Errors[i].Message, e.Errors[i].ArrayBindIndex);

          txn.Commit();
        }
        cmd.Parameters.Clear();
        cmd.CommandText = "select count(*) from depttest";

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

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

3.8.3.5.3 OracleParameterStatus Enumeration Types

Table 3-12 lists `OracleParameterStatus` enumeration values.

Table 3-12 OracleParameterStatus Members

Member Names	Description
`Success`	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.
`NullFetched`	Indicates that a `NULL` value has been fetched from a column or an `OUT` parameter.
`NullInsert`	Indicates that a `NULL` value is to be inserted into a column.
`Truncation`	Indicates that truncation has occurred when fetching the data from the column.

3.8.4 Batch Processing

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.

3.8.5 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.

3.8.5.1 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.

3.8.5.2 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.

3.8.5.3 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.

3.8.5.4 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.

3.8.5.5 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.

3.8.6 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

3.8.6.1 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 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.

3.8.6.2 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 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`.

3.8.6.3 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`.

3.9 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 3-13 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 3-13 Value Property Type of ODP.NET Type

Oracle Native Data Type or PL/SQL Data Type	ODP.NET Type	.NET Framework Data Types
`BFILE`	`OracleBFile` class	`System.Byte[]`
`BINARY_DOUBLE`	`OracleDecimal` structure	`System.Decimal`
`BINARY_FLOAT`	`OracleDecimal` structure	`System.Decimal`
`BINARY_INTEGER` (PL/SQL only)	`OracleDecimal` structure	`System.Decimal`
`BLOB`	`OracleBlob` class	`System.Byte[]`
`BOOLEAN` (PL/SQL only)	`OracleBoolean` structure	`System.Boolean`
`CHAR`	`OracleString` structure	`System.String`
`CLOB`	`OracleClob` class	`System.String`
`DATE`	`OracleDate` structure	`System.DateTime`
`INTERVAL DAY TO SECOND`	`OracleIntervalDS` structure	`System.TimeSpan`
`INTERVAL YEAR TO MONTH`	`OracleIntervalYM` structure	`System.Int64`
`LONG`	`OracleString` structure	`System.String`
`LONG RAW`	`OracleBinary` structure	`System.Byte[]`
`NCHAR`	`OracleString` structure	`System.String`
`NCLOB`	`OracleClob` class	`System.String`
`NUMBER`	`OracleDecimal` structure	`System.Decimal`
`NVARCHAR2`	`OracleString` structure	`System.String`
`PLS_INTEGER` (PL/SQL only)	`OracleDecimal` Structure	`System.Decimal`
`RAW`	`OracleBinary` structure	`System.Byte[]`
`REF`	`OracleRef` class	`System.String`
`REF CURSOR` (PL/SQL only)	`OracleRefCursor` class	Not Applicable
`ROWID`	`OracleString` structure	`System.String`

Table 3-13 (Cont.) Value Property Type of ODP.NET Type

Oracle Native Data Type or PL/SQL Data Type	ODP.NET Type	.NET Framework Data Types
`TIMESTAMP`	`OracleTimeStamp` structure	`System.DateTime`
`TIMESTAMP WITH LOCAL TIME ZONE`	`OracleTimeStampLTZ` structure	`System.DateTime`
`TIMESTAMP WITH TIME ZONE`	`OracleTimeStampTZ` structure	`System.DateTime`
`UROWID`	`OracleString` structure	`System.String`
`VARCHAR2`	`OracleString` structure	`System.String`
`XMLType`	`OracleXmlType` class	`System.String`

3.10 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

3.10.1 Typed OracleDataReader Accessors

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

- .NET Type Accessors
- ODP.NET Type Accessors

3.10.1.1 .NET Type Accessors

Table 3-14 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 "
- "OracleDataReader Class"

Table 3-14 .NET Type Accessors

Oracle Native Data Type	.NET Type	Typed Accessor
BFILE	System.Byte[]	GetBytes
BINARY_DOUBLE	System.Double	GetDouble
BINARY_FLOAT	System.Single	GetFloat
BLOB	System.Byte[]	GetBytes
CHAR	System.String System.Char[]	GetString GetChars
CLOB	System.String System.Char[]	GetString GetChars
DATE	System.DateTime	GetDateTime
INTERVAL DAY TO SECOND	System.Timespan	GetTimeSpan
INTERVAL YEAR TO MONTH	System.Int64	GetInt64
LONG	System.String System.Char[]	GetString GetChars
LONG RAW	System.Byte[]	GetBytes
NCHAR	System.String System.Char[]	GetString GetChars
NCLOB	System.String System.Char[]	GetString GetChars
NUMBER	System.Decimal System.Byte System.Int16 System.Int32 System.Int64 System.Single System.Double	GetDecimal GetByte GetInt16 GetInt32 GetInt64 GetFloat GetDouble
NVARCHAR2	System.String System.Char[]	GetString GetChars
RAW	System.Byte[]	GetBytes
REF	System.String	GetString
ROWID	System.String System.Char[]	GetString GetChars
TIMESTAMP	System.DateTime	GetDateTime
TIMESTAMP WITH LOCAL TIME ZONE	System.DateTime	GetDateTime

Table 3-14 (Cont.) .NET Type Accessors

Oracle Native Data Type	.NET Type	Typed Accessor
TIMESTAMP WITH TIME ZONE	System.DateTime	GetDateTime
UROWID	System.String	GetString
	System.Char[]	GetChars
VARCHAR2	System.String	GetString
	System.Char[]	GetChars
XMLType	System.String	GetString
	System.Xml.XmlReader	GetXmlReader

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:

- `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.

3.10.1.2 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.

Table 3-15 lists the valid type accessors that ODP.NET uses to obtain ODP.NET types for an Oracle native type.

Table 3-15 ODP.NET Type Accessors

Oracle Native Data Type	ODP.NET Type	Typed Accessor
BFILE	OracleBFile	GetOracleBFile
BINARY_DOUBLE	OracleDecimal	GetOracleDecimal
BINARY_FLOAT	OracleDecimal	GetOracleDecimal
BLOB	OracleBlob OracleBlob OracleBinary	GetOracleBlob GetOracleBlobForUpdate GetOracleBinary
CHAR	OracleString	GetOracleString
CLOB	OracleClob OracleClob OracleString	GetOracleClob GetOracleClobForUpdate GetOracleString
DATE	OracleDate	GetOracleDate
INTERVAL DAY TO SECOND	OracleIntervalDS	GetOracleIntervalDS
INTERVAL YEAR TO MONTH	OracleIntervalYM	GetOracleIntervalYM
LONG	OracleString	GetOracleString
LONG RAW	OracleBinary	GetOracleBinary
NCHAR	OracleString	GetOracleString
NCLOB	OracleString	GetOracleString
NUMBER	OracleDecimal	GetOracleDecimal
NVARCHAR2	OracleString	GetOracleString
RAW	OracleBinary	GetOracleBinary
REF	OracleRef	GetOracleRef
ROWID	OracleString	GetOracleString
TIMESTAMP	OracleTimeStamp	GetOracleTimeStamp
TIMESTAMP WITH LOCAL TIME ZONE	OracleTimeStampLTZ	GetOracleTimeStampLTZ
TIMESTAMP WITH TIME ZONE	OracleTimeStampTZ	GetOracleTimeStampTZ
UROWID	OracleString	GetOracleString
VARCHAR2	OracleString	GetOracleString
XMLType	OracleString OracleXmlType	GetOracleString GetOracleXmlType

3.10.2 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.

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

3.10.2.1 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.

3.10.2.2 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.

3.10.3 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 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`, greater than `0`, or `-1`:

- 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`

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

3.10.3.1 Setting InitialLOBFetchSize to Zero

By default, the `InitialLOBFetchSize` property is 0. This value dictates to ODP.NET that any LOBs selected will have their client LOB data fetches deferred until after the `OracleDataReader Read`, such as when using the an accessor. Each LOB value is retrieved only at the point it is individually accessed.

The advantage of using this retrieval strategy is that it conserves client memory and bandwidth. If the LOBs selected are either very large or not necessary to be immediately consumed by the end user, or both, then the application can perform better if LOBs are retrieved as needed, rather than all at once.

3.10.3.2 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. The maximum value is 2,147,483,647 (2GB). If the total size of a selected LOB is less than this number, the entire LOB data will be read.

By pre-fetching all LOB entries in one or more database round trips, applications can perform faster by reducing round trips. This approach is most advantageous when most LOBs are either small in size, or consumed by the end user almost immediately, or both. The down side of a large fetch size is higher memory consumption.

This section discusses the ways to fetch beyond the `InitialLOBFetchSize` characters or bytes that are cached.

The remaining 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`.

3.10.3.3 Setting InitialLOBFetchSize to -1

To fetch all LOB data selected during the read operation and not be bound by a set limit per LOB, set `InitialLOBFetchSize` to `-1`. A new default behavior has been introduced for ODP.NET Release 12.1.0.2 and higher when `InitialLobFetchSize` is set to `-1`.

When `LegacyEntireLOBFetch = 0`, which is the default value, the following operations are invoked for a LOB column:

- `OracleDataReader.GetOracleClob()`: returns `OracleClob` object
- `OracleDataReader.GetOracleBlob()` : returns `OracleBlob` object
- `OracleDataReader.GetOracleClobForUpdate()`: returns `OracleClob` object
- `OracleDataReader.GetOracleBlobForUpdate()`: returns `OracleBlob` object
- `OracleDataReader.GetOracleValue()`: returns `OracleClob` object for a CLOB column
- `OracleDataReader.GetOracleValue()`: returns `OracleBlob` object for a BLOB column

- `OracleDataAdapter.Fill()` with `ProviderSpecificTypes=true`: populates `DataTable` with `OracleClob` for a `CLOB` column
- `OracleDataAdapter.Fill()` with `ProviderSpecificTypes=true`: populates `DataTable` with `OracleBlob` for a `BLOB` column

To use the old behavior, set `LegacyEntireLobFetch = 1` in the ODP.NET configuration.

When `LegacyEntireLobFetch = 1` and `InitialLOBFetchSize = -1`, `GetOracleClob`, `GetOracleClobForUpdate`, `GetOracleBlob`, and `GetOracleBlobForUpdate` methods are not supported. The following operations are invoked for a LOB column in this scenario:

- `OracleDataReader.GetOracleClob()`: throws `InvalidCastException()`
- `OracleDataReader.GetOracleBlob()`: throws `InvalidCastException()`
- `OracleDataReader.GetOracleClobForUpdate()`: throws `InvalidCastException()`
- `OracleDataReader.GetOracleBlobForUpdate()`: throws `InvalidCastException()`
- `OracleDataReader.GetOracleValue()`: returns `OracleString` object for a `CLOB` column
- `OracleDataReader.GetOracleValue()`: returns `OracleBinary` object for a `BLOB` column
- `OracleDataAdapter.Fill()` with `ProviderSpecificTypes=true`: populates `DataTable` with `OracleString` for a `CLOB` column
- `OracleDataAdapter.Fill()` with `ProviderSpecificTypes=true`: populates `DataTable` with `OracleBinary` for a `BLOB` column

For releases prior to ODP.NET 12.1.0.2, 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.

3.10.3.3.1 Methods Supported for InitialLOBFetchSize of -1 and LegacyEntireLobFetch 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` and `LegacyEntireLobFetch` property is set to `1`.

Table 3-16 lists supported and not supported methods for the `CLOB` data types.

Table 3-16 Supported OracleDataReader CLOB Methods for InitialLOBFetchSize of -1 and LegacyEntireLobFetch of 1

OracleDataReader CLOB Methods	Supported
`GetChars`	Yes
`GetString`	Yes
`GetValue`	Yes
`GetValues`	Yes
`GetOracleString`	Yes
`GetOracleValue`	Yes

Table 3-16 (Cont.) Supported OracleDataReader CLOB Methods for InitialLOBFetchSize of -1 and LegacyEntireLobFetch of 1

OracleDataReader CLOB Methods	Supported
`GetOracleValues`	Yes
`GetOracleClob`	No
`GetOracleClobForUpdate`	No

Table 3-17 lists supported and not supported methods for the `BLOB` data types.

Table 3-17 Supported OracleDataReader BLOB Methods for InitialLOBFetchSize of -1 and LegacyEntireLobFetch of 1

OracleDataReader BLOB Methods	Supported
`GetBytes`	Yes
`GetValue`	Yes
`GetValues`	Yes
`GetOracleBinary`	Yes
`GetOracleValue`	Yes
`GetOracleValues`	Yes
`GetOracleBlob`	No
`GetOracleBlobForUpdate`	No

3.10.3.4 Performance Considerations Related to the InitialLOBFetchSize Property

This section discusses the advantages and disadvantages of the various `InitialLOBFetchSize` property settings in different situations.

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.

3.10.4 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.

3.10.4.1 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 database use the same cache allocated for that `OracleDataReader` object. Therefore, changing the `FetchSize` value after the first `Read` method invocation has no effect.

3.10.4.2 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 102400 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.

3.10.4.3 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.

3.10.4.3.1 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.

3.10.4.3.2 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.

3.10.4.3.3 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.

3.11 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

3.11.1 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.

3.11.2 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.RefCursor` parameter.

3.11.3 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.

3.11.4 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`.

3.11.5 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.

3.11.6 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.

3.11.7 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

3.11.8 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:

```
/*
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;
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());
  }
}
```

3.12 Implicit REF CURSOR Binding

ODP.NET enables applications to run stored procedures with `REF CURSOR` parameters without using explicit binding for these parameters in the .NET code. ODP.NET unmanaged and managed drivers support `REF CURSOR` implicit binding through configuration done in .NET configuration files.

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 CURSOR`s 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 CURSOR`s 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

3.12.1 Specifying REF CURSOR Bind and Metadata Information in the .NET Configuration File

Specify the `REF CURSOR` information in the `oracle.dataacccess.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:

```
<add
name="SchemaName.PackageName.StoredProcedureName.RefCursor.RefCursorParameterPosition
OrName"
value="implicitRefCursor bindinfo='mode=InputOutput|Output|ReturnValue'" />
```

Use the following format to specify metadata information:

```
<add
name="SchemaName.PackageName.StoredProcedureName.RefCursorMetaData.RefCursorParameter
PositionorName.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:

```
<add
name="&quot;HrSchema&quot;.&quot;HrPackage&quot;.&quot;HrStoredProcedure&quot;.RefCur
sorMetaData . . . />
```

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 CURSOR`s 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 3-18 describes the parameters that can be included in the *`AttributesList`*.

Example 3-1 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 shows a sample `add` element that uses `metadata`.

Table 3-18 Allowed Parameters in Attributes List

Name	Type	Required/Optional for Entity Framework	Description
`ColumnName`	`System.String`	Required	The name of the column.
`ProviderType`	`Oracle.DataAccess.Client.OracleDbType`	Required	The database column type (`OracleDbType`) of the column
`NativeDataType`	`System.String`	Required	The Oracle type. For example, `NCLOB`.
`BaseColumnName`	`System.String`	Optional	The name of the column in the database if an alias is used for the column.
`BaseSchemaName`	`System.String`	Optional	The name of the schema in the database that contains the column.
`BaseTableName`	`System.String`	Optional	The name of the table or view in the database that contains the column.
`ColumnSize`	`System.Int64`	Optional	The maximum possible length of a value in the column
`NumericPrecision`	`System.Int16`	Optional	The maximum precision of the column, if the column is a numeric data type.
`NumericScale`	`System.Int16`	Optional	The maximum scale of the column, if the column is a numeric data type.
`IsUnique`	`System.Boolean`	Optional	Indicates whether or not the column is unique.
`IsKey`	`System.Boolean`	Optional	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`
`IsRowID`	`System.Boolean`	Optional	`true` if the column is a `ROWID`, otherwise `false`.
`DataType`	`System.RuntimeType`	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

Table 3-18 (Cont.) Allowed Parameters in Attributes List

Name	Type	Required/Optional for Entity Framework	Description
`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.DbType`	Optional	`System.Data.DbType`
`ObjectName`	`System.String`	Optional	Represents the name of the object.

Some of the attributes, listed in Table 3-18, 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.

Example 3-1 Using the add Element with bindinfo

```
<add name="SCOTT.TESTPROC.RefCursor.parameter1" value="implicitRefCursor
 bindinfo='mode=Output'" />
```

Example 3-2 Using the add Element with metadata

```
<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'" />
```

3.12.2 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 (
  EMPID IN NUMBER) return sys_refcursor is
  emp sys_refcursor;
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

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <oracle.dataaccess.client>
    <settings>

      <!-- 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.&quot;GetEmpAndDept&quot;.RefCursor.EMPS"
           value="implicitRefCursor bindinfo='mode=Output'" />

      <!-- The following is for SCOTT.GETEMP's EMPS REF CURSOR metadata -->
      <add name="SCOTT.&quot;GetEmpAndDept&quot;
           .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.&quot;GetEmpAndDept&quot;
```

```
             .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'" />

        <!-- The following is for SCOTT.GETEMP's DEPTS REF CURSOR metadata -->
        <add name="SCOTT.&quot;GetEmpAndDept&quot;.RefCursor.DEPTS"
             value="implicitRefCursor bindinfo='mode=Output'" />

        <add name="SCOTT.&quot;GetEmpAndDept&quot;
             .RefCursorMetaData.DEPTS.Column.0"
             value="implicitRefCursor metadata='ColumnName=DEPTNO;
             BaseColumnName=DEPTNO;BaseSchemaName=SCOTT;BaseTableName=DEPT;
             NativeDataType=number;ProviderType=Int32;ProviderDBType=Int32;
             DataType=System.Int32;ColumnSize=4;NumericPrecision=10;
             NumericScale=3;AllowDBNull=false;IsKey=true'" />

        <add name="SCOTT.&quot;GetEmpAndDept&quot;
             .RefCursorMetaData.DEPTS.Column.1"
             value="implicitRefCursor metadata='ColumnName=DNAME;
             BaseColumnName=DNAME;BaseSchemaName=SCOTT;BaseTableName=DEPT;
             NativeDataType=varchar2;ProviderType=Varchar2;
             ProviderDBType=String;DataType=System.String;
             ColumnSize=10;AllowDBNull=true'" />
      </settings>
  </oracle.dataaccess.client>
</configuration>
```

Sample Application That Uses the Configuration File

```
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 SCOT.GETEMP
      OracleCommand cmd = con.CreateCommand();
      cmd.CommandText = "SCOTT.GETEMP";
      cmd.CommandType = CommandType.StoredProcedure;
      cmd.BindByName = false;
      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);

        // 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\"";
        cmd.CommandType = CommandType.StoredProcedure;
        cmd.BindByName = true;
        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);
          }
        }
      }
    }
  }
```

3.12.3 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

3.12.3.1 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`.

3.12.3.2 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.

3.12.3.3 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.

3.12.3.4 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.

3.12.3.5 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.

3.13 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

3.13.1 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 Oracle9*i*, but should not be used in new applications.

3.13.2 Oracle Data Provider for .NET LOB Objects

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

Table 3-19 shows the proper ODP.NET object to use for a particular Oracle LOB type.

Table 3-19 ODP.NET LOB Objects

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

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.

3.13.3 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.

3.13.4 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 Oracle9*i* 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`.

3.13.5 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.

3.13.6 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.

3.14 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.

ODP.NET, Managed Driver follows XPath 1.0 specification and hence it does not support default XML namespaces. XML namespaces must be explicitly added to search or update nodes. This behavior differs from ODP.NET, Unmanaged Driver.

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

ORACLE_BASE\ORACLE_HOME\ODACsamples

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

3.14.1 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.

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`.

See Also:

- "XQuery Support"
- "OracleCommand Class"
- "OracleXmlType Class"
- "OracleXmlStream Class"
- "OracleXmlQueryProperties Class"
- "OracleXmlSaveProperties Class"
- *Oracle XML DB Developer's Guide*

3.14.2 XQuery Support

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.

3.14.3 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.

3.14.4 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

3.14.4.1 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.

3.14.4.2 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.

3.14.4.2.1 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.

3.14.4.2.2 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;
  ```

3.14.4.2.3 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/>'));
```

3.14.5 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.

3.14.6 Characters with Special Meaning in XML

The following characters in Table 3-20 have special meaning in XML. For more information, refer to the XML 1.0 specifications

Table 3-20 Characters with Special Meaning in XML

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

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

Also certain characters are not valid in XML element names. When SQL identifiers (such as column names) are mapped to XML element names, these characters are converted to a sequence of hexadecimal digits, derived from the Unicode encoding of the character, bracketed by an introductory underscore, a lowercase `x` and a trailing underscore. 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.

3.14.7 Retrieving Query Result Set as XML

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

3.14.7.1 Handling Date and Time Format

The generated XML `DATE` and `TIMESTAMP` formats are based on the standard XML Schema formats.

3.14.7.2 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 3-20), 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:

```
/* 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>
```

3.14.7.3 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\"";
```

3.14.7.4 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.

3.14.7.5 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.

```
// 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:

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

3.14.7.5.1 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.

3.14.7.6 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:

```
// 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.

3.14.7.7 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.

3.14.8 Data Manipulation Using XML

This section discusses making changes to the database data using XML.

3.14.8.1 Handling Date and Time Format

The generated XML `DATE` and `TIMESTAMP` formats are based on the standard XML Schema formats.

3.14.8.2 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:

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

The following XML document is generated:

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

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

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

3.14.8.3 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 3-20), 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:

```
// 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 version = '1.0'?>
 <ROWSET>
  <ROW>
   <id>1</id >
```

```
    <NAME>&lt;Jones&gt;</NAME>
  </ROW>
</ROWSET>
```

3.14.8.4 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\""`.

3.14.8.5 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 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.

3.14.8.6 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.

3.14.8.7 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 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:

```
/* 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;";
    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();
    }
  }
```

3.14.8.7.1 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.

3.14.8.8 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.

3.14.8.9 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.

3.14.8.10 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.

3.15 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.

Note:

ODP.NET, Managed Driver does not support UDTs and .NET Custom Types

To convert between UDTs and custom types, ODP.NET uses custom interfaces.

This section discusses the following topics:

3.15.1 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.

The term UDT is used interchangeably with Oracle object types and abstract data types (ADTs).

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.

3.15.2 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:

3.15.2.1 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.DataAccess.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.DataAccess.Types.IOracleCustomTypeFactory` interface and the `Oracle.DataAccess.Types.IOracleArrayTypeFactory` interface.

  – To create an array type that represents an Oracle Collection, a custom type factory class must implement the `Oracle.DataAccess.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.

3.15.2.2 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`

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.

3.15.3 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. If a .NET application dynamically loads .NET assemblies, which contain .NET classes that Oracle UDTs are mapped to, then the mapping between .NET classes and Oracle UDTs must be configured using a .NET config file.

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 in Visual Studio 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"

3.15.3.1 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 `NUMBER`s.

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.

3.15.3.2 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.

3.15.3.2.1 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

3.15.3.2.2 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:

```
 <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>
```

3.15.3.3 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.

3.15.4 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.

3.15.5 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[]`.

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 3-21 Attribute Mappings Between UDTs and Custom Object Types

Type of UDT Attribute or Element	.NET Type	ODP.NET Type
`BFILE` #1	`System.Byte[]`	`OracleBFile`
`BINARY FLOAT`	`System.Byte, System.Int16, System.Int32, System.Int64, System.Single, System.Double, System.Decimal`	`OracleDecimal`
`BINARY DOUBLE`	`System.Byte, System.Int16, System.Int32, System.Int64, System.Single, System.Double, System.Decimal`	`OracleDecimal`
`BLOB`	`System.Byte[]`	`OracleBlob`
`CHAR`	`System.Char[], System.String`	`OracleString`
`CLOB`	`System.Char[], System.String`	`OracleClob`
`DATE`	`System.DateTime`	`OracleDate`
`INTERVAL DAY TO SECOND`	`System.TimeSpan,`	`OracleIntervalDS`
`INTERVAL YEAR TO MONTH`	`System.Int64`	`OracleIntervalYM`
`LONG RAW`	`System.Byte[]`	`OracleBinary`
`NCHAR`	`System.Char[], System.String`	`OracleString`
`NCLOB`	`System.Char[], System.String`	`OracleClob`

Table 3-21 (Cont.) Attribute Mappings Between UDTs and Custom Object Types

Type of UDT Attribute or Element	.NET Type	ODP.NET Type
Nested Table	*custom type*, *.NET type[]*, or *custom type*[]	*ODP Type[]*
NUMBER	System.Byte, System.Int16, System.Int32, System.Int64, System.Single, System.Double, System.Decimal	OracleDecimal
NVARCHAR2	System.Char[], System.String	OracleString
Object Type	*custom type*	N/A
RAW	System.Byte[]	OracleBinary
REF	System.String	OracleRef
TIMESTAMP	System.DateTime	OracleTimeStamp
TIMESTAMP WITH LOCAL TIME ZONE	System.DateTime	OracleTimeStampLTZ
TIMESTAMP WITH TIME ZONE	System.DateTime	OracleTimeStampTZ
VARCHAR2	System.Char[], System.String	OracleString
VARRAY	*custom type*, *.NET type[]*, or *custom type*[]	*ODP Type[]*

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.

3.15.6 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

OracleDataReader method/ property invocation	Column Data Type	Custom Type Mapping	Value Returned for Oracle UDT	NULL Value Returned for Oracle UDT
`Item[index]`, `Item[name]`, `GetValue()`, `GetValues()`	`Object`, `Collection`	none	Exception thrown	Exception thrown
`Item[index]`, `Item[name]`, `GetValue()`, `GetValues()`	`Object`	*`schema.type`*	*`custom object`*	`DBNull.Value`
`Item[index]`, `Item[name]`, `GetValue()`, `GetValues()`	`Collection`	*`schema.type`*	*`custom object`* | *`custom object[]`* | *`.NET Type[]`* | *`PS object[]`*	`DBNull.Value`
`Item[index]`, `Item[name]`, `GetValue()`, `GetValues()`	`REF`	none | *`schema.type`*	*`string`* (HEX)	`DBNull.Value`
`GetString()`	`REF`	none | *`schema.type`*	`string` (HEX)	Exception thrown
`GetProviderSpecificValue()`, `GetProviderSpecificValues()`, `GetOracleValue()`, `GetOracleValues()`	`Object`, `Collection`	*`schema.type`*	*`custom object`*	*`custom type`*`.Null`
`GetProviderSpecificValue()`, `GetProviderSpecificValues()`, `GetOracleValue()`, `GetOracleValues()`	`Collection`	*`schema.type`*	*`custom object[]`* | *`.NET Type[]`* | *`PS object[]`*	`null`
`GetProviderSpecificValue()`, `GetProviderSpecificValues()`, `GetOracleValue()`, `GetOracleValues()`, `GetOracleRef()`	`REF`	none | *`schema.type`*	`OracleRef`	`OracleRef.Null`
`GetOracleString()`	`REF`	none | *`schema.type`*	`OracleString` (HEX)	`OracleString.Null`

3.15.7 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

OracleDataReader Method/Property invocation	Column Data Type	Custom Type Mapping	Return Value
`GetFieldType(index)`	`Object`, `Collection`	none	Exception thrown
`GetFieldType(index)`	`Object`	*schema.type*	`typeof(`*custom type*`)`
`GetFieldType(index)`	`Collection`	*schema.type*	`typeof(`*custom type*`)` | `typeof(`*custom type[]*`))` | `typeof(`*.NET type[]*`))` | `typeof(`*PS type[]*`)`
`GetFieldType(index)`	`REF`	none | *schema.type*	`typeof(string)`
`GetProviderSpecificFieldType(index)`	`Object`, `Collection`	none	Exception thrown
`GetProviderSpecificFieldType(index)`	`Object`,	*schema.type*	`typeof(`*custom type*`)`
`GetProviderSpecificFieldType(index)`	`Collection`	*schema.type*	`typeof(`*custom type*`)` | `typeof(`*custom type[]*`))` | `typeof(`*.NET type[]*`))` | `typeof(`*PS type[]*`)`
`GetProviderSpecificFieldType(index)`	`REF`	none | *schema.type*	`typeof(OracleRef)`

3.15.8 Oracle UDT Parameter Binding with OracleParameter

This section discusses using UDT output and input parameter bindings with an `OracleParameter` object.

See Also:

"Parameter Binding"

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

3.15.8.1 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.

3.15.8.2 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

OracleParameter. Value	OracleParameter. DbType or OracleParameter. OracleDbType	OracleParameter . UdtTypeName	Custom Type Mappings	Oracle Type converted to before Binding
custom object | *custom object[]* |*.NET object[]* | *PS object[]* | `String` (HEX) | `OracleString`(HEX) | `OracleRef`	`DbType.Object` | `OracleDbType.Object` | `OracleDbType.Array` | `OracleDbType.Ref` |	not set	none | *schema.type*	Exception thrown
custom object[] |*.NET object[]* | *PS object[]*	`DbType.Object` | `OracleDbType.Object` | `OracleDbType.Array`	*schema.type*	none	Exception thrown
custom object	`DbType.Object`	*schema.type*	*schema.type*	Specified UDT is instantiated. Value is bound as Object or Collection, based on the `UdtTypeName` property

Table 3-24 (Cont.) Valid Ways to Bind Input Parameters for Oracle UDTs

OracleParameter. Value	OracleParameter. DbType or OracleParameter. OracleDbType	OracleParameter . UdtTypeName	Custom Type Mappings	Oracle Type converted to before Binding
custom object	`OracleDbType.Object`	*schema.type*	*schema.type*	Specified UDT is instantiated. *schema.type* must represent an object.
custom object	`OracleDbType.Array`	*schema.type*	*schema.type*	Specified UDT is instantiated. *schema.type* must represent a collection.
.NET object[] | *PS object[]* | *custom object[]*	`DbType.Object` | `OracleDbType.Array`	*schema.type*	*schema.type*	UDT specified by `OracleParameter.UdtTypeName` is instantiated.
.NET object[] | *PS object[]* | *custom object[]*	`OracleDbType.Object`	*schema.type*	none | *schema.type*	Exception thrown
custom object |*.NET object[]* | *PS object[]* *custom object[]*	`OracleDbType.Ref`	*schema.type*	none | *schema.type*	Exception thrown
`String` (HEX) | `OracleString` (HEX) | `OracleRef`	`DbType.Object` | `OracleDbType.Object` | `OracleDbType.Array`	*schema.type*	none | *schema.type*	Exception thrown
Char[] (HEX) | `String` (HEX) | `OracleString` (HEX) | `OracleRef`	`OracleDbType.Ref`	*schema.type*	none | *schema.type*	A `REF`

3.15.8.3 UDT Output 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 3-25 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.

Table 3-25 Valid Ways to Bind Output Parameters for Oracle UDTs

Type returned from Oracle	OracleParameter. DbType	OracleParameter. UdtTypeName	Custom Type Mappings	Type converted to
`Object/ Collection/REF`	`DbType.Object` | `OracleDbType.Object` | `OracleDbType.Array` | `OracleDbType.Ref`	not set	none | *schema.type*	Exception thrown

Table 3-25 (Cont.) Valid Ways to Bind Output Parameters for Oracle UDTs

Type returned from Oracle	OracleParameter. DbType	OracleParameter. UdtTypeName	Custom Type Mappings	Type converted to
`Object/ Collection`	`DbType.Object` | `OracleDbType.Object` | `OracleDbType.Array`	*`schema.type`*	none	Exception thrown
`Object`	`DbType.Object` | `OracleDbType.Object`	*`schema.type`*	*`schema.type`*	*`custom object`*
`Object`	`OracleDbType.Array` | `OracleDbType.Ref`	*`schema.type`*	none | *`schema.type`*	Exception thrown
`Collection`	`OracleDbType.Array` | `DbType.Object`	*`schema.type`*	*`schema.type`*	*`custom object`* | *`custom object[]`* | *`.NET object[]`* | *`PS object[]`*
`Collection`	`OracleDbType.Ref` | `OracleDbType.Object`	*`schema.type`*	none | *`schema.type`*	Exception thrown
`REF`	`DbType.Object` | `OracleDbType.Object` | `OracleDbType.Array`	*`schema.type`*	none | *`schema.type`*	Exception thrown
`REF`	`OracleDbType.Ref`	*`schema.type`*	none | *`schema.type`*	`OracleRef`

3.15.9 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 3-26 Types that Populate the DataSet with ADO.NET 2.0

Oracle Column Type	ReturnProvider-SpecificTypes Property	Custom Type Mappings	DataSet Column Type	DataSet Column Value	DataSet Column Null Value
`Object / Collection`	`False/True`	none	Exception thrown	Exception thrown	Exception thrown
`Object / Collection`	`False`	schema.type	`typeof(`*`custom type`*`)`	*`custom object`*	`DbNull.Value`
`Object / Collection`	`True`	schema.type	`typeof(`*`custom type`*`)`	*`custom object`*	*`custom object`*`.Null`
`Collection`	`False`	schema.type	`typeof(`*`custom type[]`*`)` | `typeof(`*`.NET type[]`*`)` | `typeof(`*`PS type[]`*`)`	*`.NET type[]`* | *`PS object[]`* | *`custom object[]`*	`DbNull.Value`

Table 3-26 (Cont.) Types that Populate the DataSet with ADO.NET 2.0

Oracle Column Type	ReturnProvider-SpecificTypes Property	Custom Type Mappings	DataSet Column Type	DataSet Column Value	DataSet Column Null Value
`Collection`	`True`	schema.type	`typeof(`*`custom type[]`*`)`| `typeof(`*`.NET type[]`*`)` | `typeof(`*`PS type[]`*`)`	*`.NET type[]`* | *`PS object[]`* | *`custom object[]`*	`null`
`REF`	`False`	none | schema.type	`typeof(string)`	`string`/HEX	`DbNull.Value`
`REF`	`True`	none | schema.type	`typeof(OracleRef)`	`OracleRef`	`OracleRef.Null`

3.15.10 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.

3.15.11 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.

3.15.11.1 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.

3.15.11.2 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.

3.16 Bulk Copy

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.

Note:

ODP.NET, Managed Driver does not support Bulk Copy.

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.

3.16.1 Data Types Supported by Bulk Copy

Bulk Copy supports the following Oracle database data types:

- `NUMBER`

- `BINARY_DOUBLE`
- `BINARY_FLOAT`
- `CHAR`
- `NCHAR`
- `VARCHAR2`
- `NVARCHAR2`
- `LONG`
- `CLOB`
- `BLOB`
- `DATE`
- `TIMESTAMP`
- `TIMESTAMP WITH TIME ZONE`
- `TIMESTAMP WITH LOCAL TIME ZONE`
- `INTERVAL YEAR TO MONTH`
- `INTERVAL DAY TO SECOND`

Bulk copy does not support overwrites.

3.16.2 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.

3.16.3 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.

3.16.4 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.

3.16.5 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.

3.17 Oracle Database Advanced Queuing Support

Oracle Database Advanced Queuing (AQ) provides database-integrated message queuing functionality. Oracle Database 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).

Note:

ODP.NET, Managed Driver does not support the AQ .NET classes.

As Oracle Database AQ is implemented in database tables, all operational benefits of high availability, scalability, and reliability are also applicable to queue data. Oracle Database AQ supports standard database features such as recovery, restart, and security.

The following items discuss Oracle Database 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.

- 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.

- 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.

- 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.

- 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.

- 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.

- Buffered Messaging

  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 Database 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 Database 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.

3.17.1 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 3-27 Mapping AQ Features with their ODP.NET Implementation

Functionality	ODP.NET Implementation
Create a Message	Create an `OracleAQMessage` object
Enqueue a single message	Specify the message as `OracleAQMessage`, queue as `OracleAQQueue` and enqueue options on `OracleAQQueue`, call `OracleAQQueue.`Enqueue
Enqueue multiple messages	Specify the messages as an `OracleAQMessage` array in `OracleAQQueue.`EnqueueArray
Dequeue a single message	Specify dequeue options on `OracleAQQueue` and call `OracleAQQueue.`Dequeue
Dequeue multiple messages	Call `OracleAQQueue.`DequeueArray
Listen for messages on Queue(s)	Call `OracleAQQueue.`Listen.To listen on multiple queues use static Listen method of `OracleAQQueue`
Message Notifications	Use `OracleAQQueue.`MessageAvailable Event along with the `NotificationConsumers` property

Note:

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

3.17.1.1 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
      {
        // Close/Dispose objects
        queue.Dispose();
        con.Close();
        con.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();
    }
  }
}
```

3.18 Continuous Query 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. Continuous Query Notification was previously known as Database Change Notification.

Note:

Continuous Query 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.

Note:

If a registered object is dropped from the database and a new one is created with the same name in the same schema, re-registration is required to receive notifications for the newly created object.

See Also:

Firewalls, such as Windows Firewall, may be set up to block TCP network ports, which blocks incoming database notifications. Ensure the firewall is configured so that database applications can use the designated port for Continuous Query Notification.

Beginning with Oracle Database 11*g* and ODP.NET 11*g* (11.1), Continuous Query 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 continuous query 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 continuous query notification option and choose the one that best suits their requirements.

If a large number of rows are modified at once, consuming significant shared pool resources, the application will not receive any change notifications with specific row information that had undergone changes. Rather, it will receive a notification with `OracleNotificationEventArgs.Info` property set to `OracleNotificationInfo.Error`.

This section contains the following topics:

- Continuous Query Notification Classes
- Supported Operations
- Requirements of Notification Registration
- Using Continuous Query Notification
- Best Practice Guidelines and Performance Considerations

3.18.1 Continuous Query 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 continuous query 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"
> – "OracleNotificationRequest Class"
> – "OracleNotificationEventArgs Class"

3.18.2 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 Continuous Query 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, `ROWID`s are not returned in the `OracleNotificationEventArgs` object that is passed to the continuous query 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 continuous query 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.

3.18.3 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
```

3.18.4 Using Continuous Query 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.

3.18.4.1 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.

3.18.4.2 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 continuous query notification feature.

```
// 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;

//This sample shows the continuous query notification feature in ODP.NET.
//Application specifies to get a notification when emp table is updated.
//When emp table is updated, the application will get a notification
//through an event handler.
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.OnMyNotificaton);

        // 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;
      }
    }
  }
```

3.18.5 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.

3.19 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`

Note:

ODP.NET, Managed Driver does not support Safe Type Mapping.

This section includes the following topics:

- Comparison Between Oracle Data Types and .NET Types
- SafeMapping Property

3.19.1 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 3-28 Oracle NUMBER to .NET Decimal Comparisons

Value Limits	Oracle NUMBER	.NET Decimal
Maximum	9.99999999999999999999999999999999999 e125	79,228,162,514,264,337,593,543,950,335
Minimum	-9.999999999999999999999999999999999999 e125	-79,228,162,514,264,337,593,543,950,335

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.

Table 3-29 Oracle Date to .NET DateTime Comparisons

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

Oracle TimeStamp Type to .NET DateTime Type

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 3-30 Oracle TimeStamp to .NET DateTime Comparisons

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

Oracle INTERVAL DAY TO SECOND to .NET TimeSpan

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 3-31 Oracle INTERVAL DAY TO SECOND to .NET TimeSpan Comparisons

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

3.19.2 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.

3.19.2.1 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

```
// C#
```

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

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

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

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

    // Get the connection from the adapter
    OracleConnection connection = adapter.SelectCommand.Connection;

    // Create the safe type mapping for the adapter
    // which can safely map column data to byte arrays, where
    // 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"]);
  }
}
```

3.20 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.

3.21 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:

3.21.1 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.
- 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.

3.21.2 Configuring PrimaryKey and Constraints Properties

If the minimal conditions described in "What Constitutes Uniqueness in DataRow Objects?" 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.

3.21.3 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.

3.22 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.

Note:

- ODP.NET, Managed Driver is not `NLS_LANG` sensitive. It is only .NET locale sensitive.
- ODP.NET, Managed Driver does not support thread-based globalization.

See Also:

"OracleGlobalization Class"

This section includes the following:

- Globalization Settings
- Globalization-Sensitive Operations

3.22.1 Globalization Settings

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

- Client Globalization Settings
- Session Globalization Settings
- Thread-Based Globalization Settings

3.22.1.1 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:

```
// C#

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.

3.22.1.2 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:

```
// C#

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();
  }
}
```

3.22.1.3 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. CurrentThread.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:

```
// C#

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

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

    // construct OracleDate from a string using the DateFormat specified.
    OracleDate date = new OracleDate("1999-DEC-01");

    // Set a different DateFormat for the thread
    info.DateFormat = "MM/DD/YYYY";
    OracleGlobalization.SetThreadInfo(info);

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

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

3.22.2 Globalization-Sensitive Operations

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

3.22.2.1 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[]`.

3.22.2.2 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.

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.

3.22.2.3 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.

3.22.3 ODP.NET Managed and Unmanaged Drivers Differences

ODP.NET, Managed and Unmanaged Drivers set the default session time zone differently. While the session time zone for unmanaged ODP.NET uses an hour offset, managed ODP.NET uses the region identifier for setting its session time zone. As a result, managed ODP.NET is sensitive to daylight savings in scenarios where the timestamp LTZ values have to be converted from or to the session time zone.

There are two methods to resolve this difference if needed. For ODP.NET, Unmanaged Driver, the application explicitly sets the region identifier with the environment variable `ORA_SDTZ`. For example, `set ORA_SDTZ = <Region ID>`. If `ORA_SDTZ` variable is set, Oracle Client considers this value as the session time zone. The second method is to execute an alter session command to set the session time zone property to the region identifier.

3.23 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, `TraceFileLocation`, `TraceLevel`, and `TraceOption` must be set appropriately either in the Windows Registry or in an XML configuration file. ODP.NET, Managed and Unmanaged Drivers support the XML configuration file. Windows Registry settings are available for ODP.NET, Unmanaged Driver only.

In ODAC 12*c* Release 4, ODP.NET now uses new directories to write trace files to by default.

- ODP.NET, Managed Driver: `<`*`Windows user temporary folder`*`>\ODP.NET\managed\trace`
- ODP.NET, Unmanaged Driver: <Windows user temporary folder>`\ODP.NET\unmanaged\trace`

The Windows user temporary folder is determined by your local Windows settings, such as your Windows `TMP` or `TEMP` environment variable. Typically, it can be `C:\temp` or `C:\Users\<`*`user name`*`>\AppData\Local\Temp`. ODP.NET will create an entry in the Windows event log where the trace was created anytime it creates a trace file.

For ODP.NET, Unmanaged Driver specifically, `TraceFileLocation` is now supported similar to ODP.NET, Managed Driver. `TraceFileLocation` defines the directory where the trace files will be created. Neither `TraceFileName` nor `TraceFileLocation` will be created by default in the Windows Registry.

3.24 Database Application Migration: SQL Translation Framework

A key part of migrating non-Oracle database applications to an Oracle Database requires converting non-Oracle SQL statements to SQL statements that can be processed by an Oracle Database. SQL conversion is generally a manual and laborious process. To minimize the effort, Oracle Database 12*c* introduces SQL Translation Framework which takes non-Oracle SQL statements from client applications and then translates them at run-time for the Oracle Database to execute.

The SQL Translation Framework can be used to map non-Oracle stored procedure to Oracle stored procedures to ensure successful execution of those stored procedures when migrating to Oracle Database.

Currently, SQL Translation Framework is available for Sybase Adaptive Server Enterprise and Microsoft SQL Server. There is limited support for IBM DB2.

Note:

SQL Translation Framework is only supported by ODP.NET, Unmanaged Driver. ODP.NET, Managed Driver does not support this feature.

3.24.1 The SQL Translation Profile

The SQL Translation Profile is a database object that contains the set of captured non-Oracle SQL statements, and their translations or translation errors. The SQL Translation Profile is used to review, approve, and modify translations. A profile is associated to a single translator. However, a translator can be used in one or more SQL Translation Profiles. Typically, there is one SQL Translation Profile per application, otherwise applications can share translated queries. You can export profiles among various databases.

1. Configuring the SQL Translation Profile Name

   The default translation profile name for SQL Translation Framework can be configured through the `app/web/machine` .NET configuration file. If configured, connections, by default will automatically be set to the specified profile when the connection is initially created.

2. Changing the SQL Translation Profile Name

   ODP.NET supports setting the profile name through the .NET config file, logon trigger, or database service. ODP.NET does not support using `ALTER SESSION` from an application to set the profile name.

3. Forcing Translation

   Applications are strictly prohibited to execute the following SQL which forces translation of all SQL's on the database:

   ```
   ALTER SESSION SET events = '10601 trace name context forever, level 32'
   ```

4. Connection Related Error Mapping

   Connection Related Error Mapping can be configured through the .NET configuration file. Please note that this error mapping strictly applies to errors which could be thrown before the connection is successfully established. Once the database connection is established successfully, then these error mapping will be completely ignored and further error translation will be provided through the error mapping configured in the database.

   The rules to choose an error mapping section in the configuration file are as follows:

   a. ODP.NET uses the error mapping section which matches the configured `userId`, `dataSource`, and profile, where `userId` and `dataSource` matches the corresponding values in the connection string and profile matches the `defaultProfile` configuration setting.

   b. If no error mapping section is found from 4.a.), then ODP.NET uses the error mapping section which matches the `userId`, `dataSource`, and profile similar to 4.a.), but with the profile that matches with the `defaultErrorMappingProfile` configuration setting.
   c. If still no error mapping section is found, then ODP.NET uses the global mapping, that is, `<ErrorMapping profile="*">`, if configured.

5. Stored Procedure Mapping.

   Application must map their native stored procedure names to the corresponding Oracle stored procedure names on the translation profile in the database. The following procedure can be used to setup the mapping in the database.

```
DBMS_SQL_TRANSLATOR.REGISTER_SQL_TRANSLATION(
    PROFILE_NAME    VARCHAR2    IN
    SQL_TEXT        CLOB        IN
    TRANSLATED_TEXT CLOB        IN      DEFAULT
    ENABLE          BOOLEAN     IN      DEFAULT)
```

   Example of stored procedure mapping:

```
DBMS_SQL_TRANSLATOR.REGISTER_SQL_TRANSLATION('profile_name',
 'native_sp_name',
 'oracle_sp_name');
```

4

ADO.NET Entity Framework and LINQ to Entities

This section describes ADO.NET Entity Framework and LINQ to Entities. Entity Framework is a framework for providing object-relational mapping service on data models.

This section contains these topics:

4.1 Overview of Entity Framework

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.

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.
- Code First is supported starting with Entity Framework 6 and higher.
- 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 be generated from Oracle database schemas. Schemas can be generated from entity data models. These Oracle entity data models can be queried and manipulated using Visual Studio and ODP.NET. Oracle supports Code First, 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`.

4.2 Language Integrated Query and Entity SQL

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 SQL is a language that enables querying of Entity Framework conceptual models. It allows querying Entity Framework entities and relationships in a format that is similar to SQL. ODP.NET supports querying Oracle databases through Entity SQL.

LINQ and Entity SQL syntax are generally data source neutral.

4.3 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 4-1 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 4-1 Mapping of Oracle Data Types and EDM Types

Oracle Data Types	EDM Types (Primitive-TypeKind)	Provider Type Attributes: Name and Value	EDM Type Facets
`Bfile`	`Binary`	• `Equal Comparable: False` • `Order Comparable: False`	EDM Type Facets for Bfile
`Binary_Double`	`Double`	• `Equal Comparable: True` • `Order Comparable: True`	Not Applicable
`Binary_Float`	`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

Table 4-1 (Cont.) Mapping of Oracle Data Types and EDM Types

Oracle Data Types	EDM Types (Primitive-TypeKind)	Provider Type Attributes: Name and Value	EDM Type Facets
`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
`Long Raw`	`Binary`	• `Equal Comparable: False` • `Order Comparable: False`	EDM Type Facets for Long Raw
`NChar`	`String`	• `Equal Comparable: True` • `Order Comparable: True`	EDM Type Facets for NChar
`NClob`	`String`	• `Equal Comparable: False` • `Order Comparable: False`	EDM Type Facets for NClob
`Nested Table`		Not Applicable	Not Applicable and Not Supported

`Number(1,0)`
`Number(2,0)`
`Number(3,0)`
`Number(4,0)`
`Number(5,0)`
	`Int16`	• `Equal Comparable: True`
		• `Order Comparable: True`
			Not Applicable

`Number(6,0)`
`Number(7,0)`
`Number(8,0)`
`Number(9,0)`
`Number(10,0)`
	`Int32`	• `Equal Comparable: True`
		• `Order Comparable: True`
			Not Applicable

`Number(11,0)`
`Number(12,0)`
`Number(13,0)`
`Number(14,0)`
`Number(15,0)`
`Number(16,0)`
`Number(17,0)`
`Number(18,0)`
`Number(19,0)`
	`Int64`	• `Equal Comparable: True`
		• `Order Comparable: True`
			Not Applicable

Table 4-1 (Cont.) Mapping of Oracle Data Types and EDM Types

Oracle Data Types	EDM Types (Primitive-TypeKind)	Provider Type Attributes: Name and Value	EDM Type Facets
`Number` (all other cases)	`Decimal`	• `Equal Comparable: True` • `Order Comparable: True`	EDM Type Facets for Number
`NVarchar2`	`String`	• `Equal Comparable: True` • `Order Comparable: True`	EDM Type Facets for NVarchar2
`Object`		Not Applicable	Not Applicable and Not Supported
`Raw`	`Binary`	• `Equal Comparable: True` • `Order Comparable: True`	EDM Type Facets for Raw
`Raw(16)`	`Guid`	• `Equal Comparable: True` • `Order Comparable: True`	Not Applicable
`Ref`		Not Applicable	Not Applicable and Not Supported
`ROWID`	`String`	• `Equal Comparable: True` • `Order Comparable: True`	EDM Type Facets for ROWID
`Smallint`	`Int16`	• `Equal Comparable: True` • `Order Comparable: True`	Not Applicable
`Timestamp`	`DateTime`	• `Equal Comparable: True` • `Order Comparable: True`	EDM Type Facets for Timestamp
`Timestamp with Local Time Zone`	`DateTime`	• `Equal Comparable: True` • `Order Comparable: True`	EDM Type Facets for Timestamp with Local Time Zone
`Timestamp with Time Zone`	`DateTimeOffset`	• `Equal Comparable: True` • `Order Comparable: True`	EDM Type Facets for Timestamp with Time Zone
`UROWID` (size)	`Binary`	• `Equal Comparable: True` • `Order Comparable: True`	EDM Type Facets for UROWID

Table 4-1 (Cont.) Mapping of Oracle Data Types and EDM Types

Oracle Data Types	EDM Types (Primitive-TypeKind)	Provider Type Attributes: Name and Value	EDM Type Facets
`Varchar2`	`String`	• `Equal Comparable: True` • `Order Comparable: True`	EDM Type Facets for Varchar2
`VArray`		Not Applicable	Not Applicable and Not Supported
`XMLType`	`String`	• `Equal Comparable: False` • `Order Comparable: False`	EDM Type Facets for XMLType

4.3.1 EDM Type Facets

The following sections enumerate the EDM type facets for the preceding Oracle data types. The first column of each table displays the EDM type facet names for the Oracle data type. Subsequent columns list the facet attribute names and displays their respective values.

EDM Type Facets for Bfile

Table 4-2 EDM Type Facets for Bfile

Facet name	Attributes Name and Value

`MaxLength`	`DefaultValue: 2147483648`
	`Constant: True`

`FixedLength`	`DefaultValue: False`
	`Constant: True`

EDM Type Facets for Blob

Table 4-3 EDM Type Facets for Blob

Facet name	Attributes Name and Value

`MaxLength`	`DefaultValue: 2147483648`
	`Constant: True`

`FixedLength`	`DefaultValue: False`
	`Constant: True`

EDM Type Facets for Char

Table 4-4 EDM Type Facets for Char

Facet Name	Attributes Name and Value

`MaxLength`	`Minimum: 1`
	`Maximum: 2000`
	`DefaultValue: 2000`
	`Constant: False`

`Unicode`	`DefaultValue: False`
	`Constant: True`

`FixedLength`	`DefaultValue: True`
	`Constant: True`

EDM Type Facets for Clob

Table 4-5 EDM Type Facets for Clob

Facet Name	Attributes Name and Value

`MaxLength`	`DefaultValue: 2147483647`
	`Constant: True`

`Unicode`	`DefaultValue: False`
	`Constant: True`

`FixedLength`	`DefaultValue: False`
	`Constant: True`

EDM Type Facets for Date

Table 4-6 EDM Type Facets for Date

Facet Name	Attributes Name and Value

`Precision`	`Constant: True`
	`DefaultValue: 0`

EDM Type Facets for Float

Table 4-7 EDM Type Facets for Float

Facet name	Attributes Name and Value

`Precision`	`Minimum: 0`
	`Maximum: 126`
	`DefaultValue: 0`
	`Constant: False`

Table 4-7 (Cont.) EDM Type Facets for Float

Facet name	Attributes Name and Value

`Scale`	`Minimum: 0`
	`Maximum: 38`
	`DefaultValue: 0`
	`Constant: False`

EDM Type Facets for Interval Day To Second

Table 4-8 EDM Type Facets for Interval Day To Second

Facet name	Attributes Name and Value

`Precision`	`Minimum: 1`
	`Maximum: 251`
	`DefaultValue: 251`
	`Constant: False`

`Scale`	`Minimum: 0`
	`Maximum: 9`
	`DefaultValue: 0`
	`Constant: False`

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 4-9 EDM Type Facets for Interval Year To Month

Facet name	Attributes Name and Value

`Precision`	`Minimum: 1`
	`Maximum: 250`
	`DefaultValue: 250`
	`Constant: False`

`Scale`	`Minimum: 0`
	`Maximum: 9`
	`DefaultValue: 0`
	`Constant: False`

EDM Type Facets for Long

Table 4-10 EDM Type Facets for Long

Facet name	Attributes Name and Value

`MaxLength`	`DefaultValue: 2147483647`
	`Constant: True`

`Unicode`	`DefaultValue: False`
	`Constant: True`

`FixedLength`	`DefaultValue: False`
	`Constant: True`

EDM Type Facets for Long Raw

Table 4-11 EDM Type Facets for Long Raw

Facet name	Attributes Name and Value

`MaxLength`	`DefaultValue: 2147483647`
	`Constant: True`

`FixedLength`	`DefaultValue: False`
	`Constant: True`

EDM Type Facets for NChar

Table 4-12 EDM Type Facets for NChar

Facet name	Attributes Name and Value

`MaxLength`	`Minimum: 1`
	`Maximum: 1000`
	`DefaultValue: 1000`
	`Constant: False`

`Unicode`	`DefaultValue: True`
	`Constant: True`

`FixedLength`	`DefaultValue: True`
	`Constant: True`

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

Table 4-13 EDM Type Facets for NClob

Facet name	Attributes Name and Value

`MaxLength`	`DefaultValue`: `2147483647`
	`Constant`: `True`

`Unicode`	`DefaultValue`: `True`
	`Constant`: `True`

`FixedLength`	`DefaultValue`: `False`
	`Constant`: `True`

EDM Type Facets for Number

Table 4-14 EDM Type Facets for Number

Facet name	Attributes Name and Value

`Precision`	`Minimum`: `1`
	`Maximum`: `38`
	`DefaultValue`: `38`
	`Constant`: `False`

`Scale`	`Minimum`: `0`
	`Maximum`: `38`
	`DefaultValue`: `0`
	`Constant`: `False`

EDM Type Facets for NVarchar2

Table 4-15 EDM Type Facets for NVarchar2

Facet name	Attributes Name and Value

`MaxLength`	`Minimum`: `1`
	`Maximum`: `2000`
	`DefaultValue`: `2000`
	`Constant`: `False`

Table 4-15 (Cont.) EDM Type Facets for NVarchar2

Facet name	Attributes Name and Value

`Unicode`	`DefaultValue: True`
	`Constant: True`

`FixedLength`	`DefaultValue: False`
	`Constant: True`

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 4-16 EDM Type Facets for Raw

Facet name	Attributes Name and Value

`MaxLength`	`Minimum: 1`
	`Maximum: 2000`
	`Constant: False`

`FixedLength`	`DefaultValue: False`
	`Constant: True`

EDM Type Facets for ROWID

Table 4-17 EDM Type Facets for ROWID

Facet name	Attributes Name and Value

`MaxLength`	`DefaultValue: 18`
	`Constant: True`

`Unicode`	`DefaultValue: False`
	`Constant: True`

`FixedLength`	`DefaultValue: True`
	`Constant: True`

EDM Type Facets for Timestamp

Table 4-18 EDM Type Facets for Timestamp

Facet name	Attributes Name and Value

`Precision`	`Minimum: 0`
	`Maximum: 9`
	`DefaultValue: 6`
	`Constant: False`

EDM Type Facets for Timestamp with Local Time Zone

Table 4-19 EDM Type Facets for Timestamp with Local Time Zone

Facet name	Attributes Name and Value

`Precision`	`Minimum: 0`
	`Maximum: 9`
	`DefaultValue: 6`
	`Constant: False`

EDM Type Facets for Timestamp with Time Zone

Table 4-20 EDM Type Facets for Timestamp with Time Zone

Facet name	Attributes Name and Value

`Precision`	`Minimum: 0`
	`Maximum: 9`
	`DefaultValue: 6`
	`Constant: False`

EDM Type Facets for UROWID

Table 4-21 EDM Type Facets for UROWID

Facet name	Attributes Name and Value

`MaxLength`	`DefaultValue: 4000`
	`Constant: True`

`FixedLength`	`DefaultValue: True`
	`Constant: True`

EDM Type Facets for Varchar2

Table 4-22 EDM Type Facets for Varchar2

Facet name	Attributes Name and Value

`MaxLength`	`Minimum`: `1`
	`Maximum`: `4000`
	`DefaultValue`: `4000`
	`Constant`: `False`

`Unicode`	`DefaultValue`: `False`
	`Constant`: `True`

`FixedLength`	`DefaultValue`: `False`
	`Constant`: `True`

EDM Type Facets for XMLType

Table 4-23 EDM Type Facets for XMLType

Facet name	Attributes Name and Value

`MaxLength`	`DefaultValue`: `2147483647`
	`Constant`: `True`

`Unicode`	`DefaultValue`: `True`
	`Constant`: `True`

`FixedLength`	`DefaultValue`: `False`
	`Constant`: `True`

4.4 Oracle Number Default Data Type Mapping and Customization

This section describes the default number mapping behavior and how to customize it for your application. You can configure a custom mapping in the .NET configuration file to override the default mapping for each Oracle `NUMBER(p,0)`, which represents integer values.

Oracle `NUMBER` data types that represent integers do not have a matching .NET integer data type with exactly the same range of acceptable values. ODP.NET uses a default mapping that ensures any .NET integer type values can be stored within the Oracle database without requiring custom data type mapping. However, it is possible that Oracle `NUMBER(p,0)` column data can be larger than what a .NET data type can hold when retrieving values from the database.

For example, in Entity Framework 6, Oracle `NUMBER(3,0)` has a default mapping to .NET Byte. Oracle `NUMBER(3,0)` can store a value up to `999`, while a .NET `BYTE` can store up to the value of `255`. If you expect the Oracle data to exceed `255`, modify the mapping to a larger numeric data type, such as a .NET `Int16`. Setting up this custom mapping allows you to consume the data in .NET without encountering an error. When such a custom mapping is used, be cautious not to insert a .NET `Int16` value beyond what an Oracle `NUMBER(3,0)` column can hold. Trying to insert `Int16.MaxValue` (i.e. `32,767`) into a `NUMBER(3,0)` column will cause an Oracle Database error.

4.4.1 Entity Framework 5 and Earlier Mapping and Customization

Example 4-1 shows an ODP.NET, Unmanaged Driver sample `app.config` file that uses custom mapping to map the `Number(1,0)` Oracle data type to the `bool` EDM type. For example, `Number(1,0)`, which is mapped to `Int16` by default, can be custom mapped to the .NET `Bool` or .NET `Byte` type. This example maps `Number(3,0)` to byte, and sets the maximum precisions for the `Int16`, `Int32`, and `Int64` data types to `4`, `9`, and `18` respectively.

Example 4-2 shows the same changes as Example 4-1, but using the traditional ODP.NET, Unmanaged Driver `app.config` format.

Example 4-3 shows a ODP.NET, Managed Driver sample `app.config` file.

Example 4-1, Example 4-2, and Example 4-3 customizes the mappings as follows:

Oracle Type	Default EDM Type	Custom EDM Type
`Number(1,0)`	`Int16`	`bool`
`Number(2,0)` to `Number(3,0)`	`Int16`	`byte`
`Number(4,0)`	`Int16`	`Int16`
`Number(5,0)`	`Int16`	`Int32`
`Number(6,0)` to `Number(9,0)`	`Int32`	`Int32`
`Number(10,0)`	`Int32`	`Int64`
`Number(11,0)` to `Number(18,0)`	`Int64`	`Int64`
`Number(19,0)`	`Int64`	`Decimal`

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.

Example 4-1 First Sample ODP.NET, Unmanaged Driver Application Configuration File to Custom Map the Number (p,0) Data Type

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <oracle.unmanageddataaccess.client>
    <version number="*">
       <edmMappings>
           <edmMapping dataType="number">
               <add name="bool" precision="1"/>
               <add name="byte" precision="3" />
               <add name="int16" precision="4" />
               <add name="int32" precision="9" />
               <add name="int64" precision="18" />
          </edmMapping>
       </edmMappings>
     </version>
  </oracle.unmanageddataaccess.client>
</configuration>
```

Example 4-2 Second Sample ODP.NET, Unmanaged Driver Application Configuration File to Custom Map the Number (p,0) Data Type

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <connectionStrings>
  </connectionStrings>
  <oracle.dataaccess.client>
    <settings>
      <add name="bool" value="edmmapping number(1,0)" />
      <add name="byte" value="edmmapping number(3,0)" />
      <add name="int16" value="edmmapping number(4,0)" />
      <add name="int32" value="edmmapping number(9,0)" />
      <add name="int64" value="edmmapping number(18,0)" />
    </settings>
  </oracle.dataaccess.client>
</configuration>
```

Example 4-3 Sample ODP.NET, Managed Driver Application Configuration File to Custom Map the Number Data Type

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <oracle.manageddataaccess.client>
    <version number="*">
       <edmMappings>
           <edmMapping dataType="number">
               <add name="bool" precision="1"/>
               <add name="byte" precision="3" />
               <add name="int16" precision="4" />
               <add name="int32" precision="9" />
               <add name="int64" precision="18" />
          </edmMapping>
       </edmMappings>
     </version>
  </oracle.manageddataaccess.client>
</configuration>
```

4.4.2 Entity Framework 6 Mapping and Customization

ODP.NET 12.1.0.2 introduces a new .NET configuration setting format for both managed and unmanaged ODP.NET. This new setting format applies only for use with Entity Framework 6 and Entity Data Model mappings, including Code First, Database First, and Model First use cases. Developers can continue using the existing ODP.NET format for non-Entity Framework 6 applications.

This new format unifies how ODP.NET, Managed and Unmanaged Drivers sets their configuration values and supports auto-completion.

The following is an example of an `edmMappings` section for ODP.NET, Managed Driver:

```
<oracle.manageddataaccess.client>
  <version number="*">
    <edmMappings>
      <edmNumberMapping>
        <add NETType="bool" MinPrecision="1" MaxPrecision="1" DBType="Number" />
        <add NETType="byte" MinPrecision="2" MaxPrecision="3" DBType="Number" />
        <add NETType="int16" MinPrecision="4" MaxPrecision="5" DBType="Number" />
        <add NETType="int32" MinPrecision="6" MaxPrecision="10" DBType="Number" />
        <add NETType="int64" MinPrecision="11" MaxPrecision="19" DBType="Number" />
      </edmNumberMapping>
    </edmMappings>
  </version>
</oracle.manageddataaccess.client>
```

Where:

- `DBType` is the Oracle Database data type
- `NETType` is the .NET data type that the Oracle data type maps to
- `MinPrecision` is the minimum range the Oracle data type will map to the .NET type
- `MaxPrecision` is the maximum range the Oracle data type will map to the .NET type

The following is an example of an `edmmappings` section for ODP.NET, Unmanaged Driver. It is exactly same format as the managed driver with the exception of the opening and closing tags.

```
<oracle.unmanageddataaccess.client>
  <version number="*">
    <edmMappings>
      <edmNumberMapping>
        <add NETType="bool" MinPrecision="1" MaxPrecision="1" DBType="Number" />
        <add NETType="byte" MinPrecision="2" MaxPrecision="3" DBType="Number" />
        <add NETType="int16" MinPrecision="4" MaxPrecision="5" DBType="Number" />
        <add NETType="int32" MinPrecision="6" MaxPrecision="10" DBType="Number" />
        <add NETType="int64" MinPrecision="11" MaxPrecision="19" DBType="Number" />
      </edmNumberMapping>
    </edmMappings>
  </version>
</oracle.unmanageddataaccess.client>
```

4.4.2.1 New Default Mappings

For Entity Framework 6, ODP.NET 12.1.0.2 introduces new default mappings that apply to Code First, Database First, and Model First scenarios. These changes were necessary to support Code First interoperability.

- .NET `Booleans` map to Oracle `Number(1,0)` and vice-versa by default
- .NET `Bytes` map to Oracle `Number(2,0)` and `Number(3,0)` and vice-versa by default

This default behavior can be changed by providing an alternative data type mapping by configuring the section of the .NET config file.

4.4.3 Data Type Mapping and Customization Process

To enable custom mapping, add the mapping information to the .NET config file *prior* to EDM creation.

If the EDM was created already before providing the mapping information, then you can *modify* the mappings either through the Visual Studio tools or manually. Using Visual Studio, go to the EDM Model Browser page. Right-click on the table(s) requiring new data type mapping and select **Table Mapping** from the pop-up menu. The **Mapping Details** window will appear usually at the bottom of your screen. Update **Column Mappings** as desired.

If you need to *add* or *delete* mappings, find the **Type** values in the CSDL mapping section of your project's existing EDMX file. Add or delete those **Type** values to the .NET data types you want the application to use. In the example below, the property name types for `BOOLCOL` and `BYTECOL` are added to the CSDL and mapped to Boolean and Byte, respectively.

Example Mapping Before CSDL Customization:

```
<Property Name="INT16COL" Type="Int16" Nullable="false" />
```

Example Mapping After CSDL Customization:

```
<Property Name="BOOLCOL" Type="Boolean" Nullable="false" />
<Property Name="BYTECOL" Type="Byte" Nullable="false" />
<Property Name="INT16COL" Type="Int16" Nullable="false" />
```

You can employ combinations of these customization possibilities depending on your planned mapping changes. If *many* tables and *many* columns require mapping changes, it is most efficient to delete the EDMX file and regenerate the data model. If a *few* tables and *many* columns require changes, then delete the affected tables, save the EDMX file, and select **Update Model from Database...** to include those tables again. If only a *single* table and *one or two* columns require changes, then modify the EDMX either manually or by using the **Mapping Details** window.

Note:

When using the EDM wizard to create a complex type from a function import, any custom EDM type mappings specified will not be applied automatically. The EDM wizard uses the default type mappings. Developers must then manually edit the resulting complex type. Developers begin this process after the complex type is generated. Any type declaration (field, property, constructor parameter, etc.) in the complex object which has an undesired type mapping, such as Decimal rather than Boolean, should be manually edited to the desired type.

4.4.4 StoreGeneratedPattern Enumeration

The following sections describe the Identity attribute and the Virtual column.

4.4.4.1 Identity Attribute

Oracle Database 12*c* (12.1) and later versions support table or view Identity attribute columns. Oracle has three Identity attribute types. When the EDM wizard generates a data model from an Oracle Identity attribute-containing table or view, ODP.NET will set the value of `StoreGeneratedPattern` to `Identity` in the `.edmx` file for any of three Oracle Identity types. The Identity attribute-associated column will use the server-generated value during `INSERT`: hence, application developers no longer need to create a sequence nor trigger. If the .NET application attempts to set the Identity attribute itself, this value will be ignored.

For Oracle Database 11*g* Release 2 (11.2) and earlier versions that do not support Identity columns, application developers can manually set `StoreGeneratedPattern` to `Identity` in columns through the entity model designer Properties after model generation, then create an `INSERT` trigger. Depending on the data type, a sequence may not be necessary if a server function, such as `sys_guid()`, can generate the value for the column.

4.4.4.2 Virtual Column

Oracle Database 11*g* (11.1) and later versions can store expressions directly in base tables as Virtual columns, also known as Generated columns. Virtual columns cannot be inserted into or updated. ODP.NET will not automatically set `StoreGeneratedPattern` to `Computed` in the EF model for Virtual columns. To avoid errors, application developers need to add or change the value of `StoreGeneratedPattern` to `Computed` for Virtual columns after the model generation. Once done, Virtual columns are excluded from `INSERTs` and `UPDATEs` upon calling `SaveChanges()`.

4.4.5 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. One specific scenario is when Visual Studio opens a project with an existing custom mapping that now generates errors when those errors did not exist before. 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.

4.4.6 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.

4.5 Migrating Existing Entity Framework 5 Applications to Entity Framework 6

To migrate existing Database First Entity Framework 5 applications to Entity Framework 6, use the following instructions. The first four steps are generic to all Entity Framework applications. The last four steps are specific to Oracle deployments.

1. Uninstall Entity Framework 5 in Visual Studio Package Manager Console. For example,

   ```
   Uninstall-Package EntityFramework
   ```

2. Install Entity Framework 6 in Package Manager Console. For example,

   ```
   Install-Package EntityFramework -Version 6.0.2
   ```

   This step adds Entity Framework 6 to the `configSections` entry and adds a new section called `entityFramework`.

3. Delete the following namespaces from your application:

   ```
   // C#
   using System.Data.EntityClient;
   using System.Data.Objects;
   ```

4. Add the following namespaces to your application:

   ```
   // C#
   using System.Data.Entity.Core.EntityClient;
   using System.Data.Entity.Core.Objects;
   ```

5. Add the Oracle Entity Framework 6 provider configuration information to the .NET config file in the `providers` section. Modify the ODP.NET version if using a version besides 6.121.2.0. If you installed the ODP.NET NuGet package, you can skip this step as the NuGet install has already added made this change.

   ```
   <provider invariantName="Oracle.DataAccess.Client"
   type="Oracle.DataAccess.EntityFramework.EFOracleProviderServices,Oracle.DataAcces
   s.EntityFramework, Version=6.121.2.0, Culture=neutral,
   PublicKeyToken=89b483f429c47342" />

   <provider invariantName="Oracle.ManagedDataAccess.Client"
   type="Oracle.ManagedDataAccess.EntityFramework.EFOracleProviderServices,Oracle.Ma
   ```

```
nagedDataAccess.EntityFramework, Version=6.121.2.0, Culture=neutral,
PublicKeyToken=89b483f429c47342" />
```

6. Add the `Oracle.ManagedDataAccess.EntityFramework` or `Oracle.DataAccess.EntityFramework` assembly as a reference to the project.
7. Modify the Oracle data type to .NET data type mappings as required by your application. See "Entity Framework 6 Mapping and Customization" for more details.
8. Rebuild the application.

4.6 Code First

Using the Entity Framework Code First modeling path, developers define the application domain model using source code rather than working directly with a designer or an XML-based configuration file. The classes defined within the source code become the model. The Code First model path offers an alternative to the existing Entity Framework Database First and Model First paths. Within Code First, the classes defined in code that comprise the model are known as Plain Old CLR Objects (POCOs). This name derives from the fact that these classes have no dependency upon Entity Framework itself and are independent of it.

Oracle's support for the Code First modeling path enables .NET developers to take advantage of Oracle Database benefits.

4.6.1 Mapping of .NET Types to Oracle Types

When using the Code First path, the model is defined by the application's classes and properties. The property data types need to be mapped to the Oracle Database table data types. The following table lists the default mapping of supported .NET types to Oracle types as well as how to map a String property to non-default Oracle types:

Table 4-24 Mapping of .NET Data Types to Oracle Data Types

.NET Data Type	Oracle Data Type	Mapping Method
`Boolean`	`number(1, 0)`	Use EDM Mapping Note: Requires use of EDM Mapping configuration. Reference the EDM Mapping sections in the documentation for additional information.
`Byte`	`number(3, 0)`	Use EDM Mapping Note: Requires use of EDM Mapping configuration. Reference the EDM Mapping sections in the documentation for additional information.
`Byte[]`	`blob`	Default

Table 4-24 (Cont.) Mapping of .NET Data Types to Oracle Data Types

.NET Data Type	Oracle Data Type	Mapping Method
`Int16`	`number(5, 0)`	Default Note: The default mapping of integer types may be specified in the EDM Mapping configuration. Reference the EDM Mapping sections in the documentation for additional information.
`Int32`	`number(10, 0)`	Default Note: The default mapping of integer types may be specified in the EDM Mapping configuration. Reference the EDM Mapping sections in the documentation for additional information.
`Int64`	`number(19, 0)`	Default Note: The default mapping of integer types may be specified in the EDM Mapping configuration. Reference the EDM Mapping sections in the documentation for additional information.
`Decimal`	`number(18, 2)`	Default
`Single`	`binary_float`	Default
`Double`	`binary_double`	Default
`Guid`	`raw(16)`	Default
`DateTime`	`date`	Default
`DateTimeOffset`	`timestamp with time zone`	Default
`String`	`nclob`	Default
`String`	`clob`	Set Unicode to false using `IsUnicode()` fluent API
`String`	`nvarchar2`	Set Max Length to <= `2000` using `HasMaxLength()` fluent API or `MaxLength` data annotation
`String`	`varchar2`	Set Max Length to <= `4000` using `HasMaxLength()` fluent API or `MaxLength` data annotation and set Unicode to false using `IsUnicode()` fluent API

Table 4-24 (Cont.) Mapping of .NET Data Types to Oracle Data Types

.NET Data Type	Oracle Data Type	Mapping Method
`String`	`nchar`	Set Max Length to <= `1000` using HasMaxLength() fluent API or MaxLength annotation and Set Column Type to `NCHAR` using HasColumnType() fluent API or Column data annotation
`String`	`char`	Set Max Length to <= `2000` using `HasMaxLength()` fluent API or `MaxLength` annotation and Set Column Type to `CHAR` using `HasColumnType()` fluent API or `Column` data annotation
`String`	`Long`	Set Column Type to `LONG` using `HasColumnType()` fluent API or `Column` data annotation Note: The long data type is deprecated and not recommended.
String	rowid	Set Column Type to `ROWID` using `HasColumnType()` fluent API or `Column` data annotation
String	urowid	Set Column Type to `UROWID` using `HasColumnType()` fluent API or `Column` data annotation

Note:

The character based columns, namely, `CHAR`, `NCHAR`, `VARCHAR2`, `NVARCHAR2` will be created using character semantics to be able to store the specified `Max Length` amount of characters. However, due to the Oracle database limit, these columns can store only up to `4000` bytes. As such, these columns may not be able to store `4000` characters even if `Max Length` is set to `4000` characters since one character may require multiple number of bytes of storage, depending on the data and the database character set. If the character data can be longer than `4000` bytes, it may be more appropriate to use `CLOB` or `NCLOB` column.

Influencing the Oracle Data Type Characteristics

The type mappings listed in the previous table represent the mappings that occur by default or what is known as convention in Entity Framework. As illustrated with the `String` type, you can influence the resulting Oracle Data Type for a property as well as characteristics of that data type. There are two Entity Framework methods to influence the resulting Oracle Data Type: Data Annotations and the Code First Fluent API. Data Annotations permit you to explicitly mark a class property with one or more attributes, whereas the Code First Fluent API permits you to use code rather than attributes to achieve the same goal. For additional information regarding the use of Data

Annotations and the Code First Fluent API refer to the MSDN Entity Framework documentation.

The following table illustrates the available functionality:

Table 4-25 Mapping of Data Annotations and the Code First Fluent APIs

Data Annotation	Fluent API	Purpose	Applies To
`Key`	`HasKey`	Set a property as the Primary Key.	All Scalar Types
`Required`	`IsRequired`	Set the database column as `NOT NULL`.	All
`MaxLength`	`HasMaxLength`	Specifies the maximum length of the property.	`String`
`NotMapped`	`Ignore`	Indicates the property is not mapped to a database column.	All
`ConcurrencyCheck`	`IsConcurrencyToken`	Indicates the column should be used for optimistic concurrency checking. Note: Do not use with an unbounded (no maximum length specified) string property as this will create a LOB column. Use of a LOB column in the concurrency check will result in an `ORA-00932: inconsistent datatypes` error.	All
`TimeStamp`	`IsRowVersion`	Indicates to create the column as a `rowversion` column.	Not Supported
`Column`	`HasColumnType`	Indicates the provider-specific type to use for the database column. Note: Must be a legal compatible type. For example a `Date` property is not legal to map to a number column. Use the `TypeName` property with the `Column` Data Annotation to specify the type.	All

Table 4-25 (Cont.) Mapping of Data Annotations and the Code First Fluent APIs

Data Annotation	Fluent API	Purpose	Applies To
N/A	`IsUnicode`	Indicates to create the column as an N-type, that is, `nvarchar2` or `nclob`. Default is true. Note: There is no Data Annotation equivalent for `IsUnicode`.	`String`
N/A	`HasPrecision`	Indicates the precision and scale for a decimal property. Note: There is no Data Annotation equivalent for `HasPrecision`.	`Decimal`

4.6.2 Code First Migrations

The Oracle Data Provider for .NET supports Code First Migrations functionality. The use of Code First Migrations with Oracle Database is supported through the Package Manager Console window migrations commands. For information on these commands, refer to the MSDN Code First Migrations documentation:

http://msdn.microsoft.com/en-us/data/jj591621.aspx

Code First Migrations utilizes a table known as the Migration History table for tracking migration operations as well as model changes. ODP.NET creates this table, by default, in the user schema specified in the context connection string. This table is named **__MigrationHistory**.

This table can be created in another user schema besides the user specified in the context connection string. This is accomplished through a process known as Migration History Table Customization, which is described in the following MSDN documentation.

http://msdn.microsoft.com/en-us/data/dn456841

Note:

- Changing the user schema for the table is the only supported customization.
- Code First Automatic Migrations is limited to working with the `dbo` schema only. Due to this limitation it is recommended to use code-based migrations, that is, add explicit migrations through the Add-Migration command.

4.6.2.1 Code First Migrations With No Supporting Code Migration File

When using Code First Migrations with ODP.NET, the migration history table may be dropped if no supporting code migration file existed prior to updating the database. Developers should ensure the supporting code migration file has been added prior to updating the database.

The following steps can remove the migration history table:

1. Execute application to create database objects
2. **Enable-Migrations** in the Package Manager Console
3. Make code change to POCO
4. **Update-Database** in the Package Manager Console

The following steps ensure the code migration file is created:

1. Execute application to create database objects
2. **Enable-Migrations** in the Package Manager Console
3. Make code change to POCO
4. **Add-Migration** in the Package Manager Console. This step will create the necessary code migration file.
5. **Update-Database** in the Package Manager Console

4.6.3 Code First Database Initialization

ODP.NET supports the following Code First Database Initializer methods:

- `CreateDatabaseIfNotExists` (default if none specified)
- `DropCreateDatabaseAlways`
- `DropCreateDatabaseIfModelChanges`
- `NullDatabaseInitializer`
- `MigrateDatabaseToLatestVersion`

These methods are documented on MSDN.

Due to differences in how Oracle and SQL Server define a database, database initialization actions work on all of the Oracle objects in the model. An Oracle Database is not created or dropped, rather the objects that compose the model are considered to be the database for these operations.

4.6.4 Oracle Database Object Creation

In order to support the client application, ODP.NET will create and maintain the required database objects. The following are the database objects created and maintained by the provider:

- Table
- Table Column
- Primary Key

- Foreign Key
- Index
- Sequence
- Trigger

> **Note:**
>
> Sequences and triggers may be created in Oracle Database 11*g* Release 2 and earlier databases to support identity columns.

For objects which directly relate to a client application object, namely, a table which represents an application class and a table column which represents a class property, the object names used are those provided by the client. These object names must conform to the object identifier length limits for Oracle Database. For example, if a class name length exceeds the valid object identifier length in Oracle Database then the `ORA-00972: identifier is too long` exception will be raised at object creation time.

For the remaining objects, ODP.NET utilizes a name generation algorithm if the supplied name length exceeds the database identifier length limit. If the supplied name length does not exceed the database limit the name is used as-is. In all cases, the object name is created as a quoted identifier in order to preserve case and any special characters which may be part of the identifier.

In cases where the provider generates a name to comply with database identifier length limits, the name is composed of the following underscore separated elements:

- A substring of the original name (from the first character)
- A numeric suffix value calculated from the original name

The following example illustrates the results of the name generation algorithm using a simple POCO in the client application:

```
public class LongSamplePocoTestClassName
{
  [Key]
  public int Id { get; set; }

  [MaxLength(64)]
  public string Name { get; set; }
}
```

The default name for the Primary Key for the resulting table will be:

```
PK_LongSamplePocoTestClassNames
```

As this name contains 31 characters (single byte per character), it violates the database identifier restrictions. The rewritten Primary Key name will resemble the following value:

```
PK_LongSamplePocoTes_730795129
```

The algorithm is designed to utilize as many characters as possible from the original name such that the new name does not violate the identifier length restrictions.

Controlling Table Name and Owner

Through the use of Data Annotations or the Entity Framework Fluent API you may control the table name, as well as the table owner. For example, you may choose to explicitly set the table name to conform to your organization's naming standards or if you do not wish to, use the name Entity Framework provides. The `Table` Data Annotation is used to control both the table name and the owner. When using the Fluent API, the `ToTable` method is used to control the table name and the owner within the `OnModelCreating` override in your class which derives from `DbContext`.

The following examples use an incomplete class definition to illustrate these actions.

Setting the table name using a Data Annotation:

```
[Table("Employee")]
public class Employee
```

Setting the table name using the Fluent API:

```
protected override void OnModelCreating(DbModelBuilder modelBuilder)
{
  modelBuilder.Entity<Employee>().ToTable("Employee");
}
```

Setting the table name and the owner using a Data Annotation:

```
[Table("Employee ", Schema="TESTUSER")]
public class Employee
```

Setting the table name and the owner using the Fluent API:

```
protected override void OnModelCreating(DbModelBuilder modelBuilder)
{
  modelBuilder.Entity<Employee>().ToTable("Employee", "TESTUSER");
}
```

Note:

When using Data Annotations or the Fluent API as above to set the owner, it is required to also set the name.

Setting the Default Table Owner

Rather than set the table owner for each user table, Entity Framework 6 and higher allows you to set the default owner to be used. This is done by invoking the `HasDefaultSchema` method within the `OnModelCreating` override in your class, which derives from `DbContext`.

For example, the following code will cause all user tables to be created within the `TESTUSER` schema by default:

```
protected override void OnModelCreating(DbModelBuilder modelBuilder)
{
  modelBuilder.HasDefaultSchema("TESTUSER");
}
```

Note:

The owner name is case-sensitive.

4.6.5 Using the Default Connection Factory

The default connection factory allows ODP.NET connections to be created by providing an Oracle connection string to the `DbContext` constructor. For example, the following entry could be used to configure the ODP.NET, Managed Driver default connection factory:

```
<defaultConnectionFactory
type="Oracle.ManagedDataAccess.EntityFramework.OracleConnectionFactory,
Oracle.ManagedDataAccess.EntityFramework,
Version=6.121.2.0,
Culture=neutral,
PublicKeyToken=89b483f429c47342" />
```

When using the default connection factory, the application supplies an Oracle connection string to the `DbContext` base constructor as follows:

```
public class TestContext : DbContext
{
  public TestContext()
    : base("<connection string>")
  {
  }
}
```

Where `<connection string>` is the ODP.NET connection string. This allows the application to connect to the database using code similar to the following:

```
using (var ctx = new TestContext())
{
  ...
}
```

For additional information please see the MSDN documentation for the `IDbConnectionFactory` interface:

`http://msdn.microsoft.com/en-us/library/system.data.entity.infrastructure.idbconnectionfactory%28v=vs.113%29.aspx`

4.7 Unsupported Entity Framework Features

The following items are not supported by the current release of the provider:

- Mapping Code First Insert, Update, Delete operations to Stored Procedures
- TimeStamp/RowVersion properties
- Custom Configuration
- Spatial Types
- Table-valued functions

- Asynchronous Query and Save
- Connection Resiliency
- Oracle synonyms

5 Oracle Data Provider for .NET Stored Procedures

This section 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 for Microsoft Windows

This section 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

5.1 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.

```
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;

      // Create and open a context connection
      OracleConnection conn = new OracleConnection();
      if( OracleConnection.IsAvailable == true )
      {
        conn.ConnectionString = "context connection=true";
      }
      else
      {
        //set connection string for a normal client connection
        conn.ConnectionString = "user id=scott;password=tiger;" +
          "data source=oracle";
      }
      conn.Open();

      // Create and execute a command
      OracleCommand cmd = conn.CreateCommand();
      cmd.CommandText = "SELECT DEPTNO FROM EMP WHERE EMPNO = :1";
      cmd.Parameters.Add(":1",OracleDbType.Int32,empno,
      System.Data.ParameterDirection.Input);
      OracleDataReader rdr = cmd.ExecuteReader();
      if (rdr.Read())
        deptno = (uint)rdr.GetInt32(0);
      rdr.Close();
      cmd.Dispose();
      conn.Close();
      return deptno;
    } // GetDeptNo
  } // CLRLibrary1
```

5.2 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.

Note:

ODP.NET, Managed Driver does not support .NET stored procedures.

5.2.1 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.

5.2.2 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)

```
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(
                "context connection=true");
            conn.Open();
            // Create new command object from connection context
            OracleCommand Cmd = conn.CreateCommand();
            Cmd.CommandText = "INSERT INTO EMP( EMPNO, ENAME, JOB," +
                "MGR, HIREDATE, SAL, COMM, DEPTNO ) " +
                "VALUES ( :1, :2, 'ANALYST', 7566, " +
                "'06-DEC-04', 5000, 0, 20 )";
            Cmd.Parameters.Add( ":1", OracleDbType.Int32,
                EmpNo, ParameterDirection.Input );
            Cmd.Parameters.Add( ":2", OracleDbType.Varchar2,
                EmpName, ParameterDirection.Input );
            Cmd.ExecuteNonQuery();
        }
    }
}
```

Example (Client Application)

The example enters new employee, Bernstein, employee number 7950, into the `EMP` table.

```
// 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;
         }
        }
```

5.2.3 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.

5.2.4 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.

5.3 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 5-1 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 headings 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 5-1 API Support Comparison Between Client Application and .NET Stored Procedure

Class or Class Members	Client Application	Implicit Connection/Explicit Connection
OnChangeEventHandler Delegate -all members	Yes	No/No
OracleDependency Class -all members	Yes	No/No
OracleNotificationEventArgs Class -all members	Yes	No/No

Table 5-1 (Cont.) API Support Comparison Between Client Application and .NET Stored Procedure

Class or Class Members	Client Application	Implicit Connection/Explicit Connection
OracleNotificationRequest Class	Yes	No/No
-all members		
OracleFailoverEventArgs Class	Yes	No/No
-all members		
OracleFailoverEventHandler Delegate	Yes	No/No
-all members		
OracleTransaction Class	Yes	No/No
-all members		
OracleCommand Class	Yes	
-Transaction Property		No: Always returns null /No: Always returns null.
OracleConnection Class		
-ConnectionTimeout Property	Yes	Yes: Implicit database connection always returns `0`/Yes
-DataSource Property	Yes	Yes: Implicit database connection always returns an empty string/Yes
-BeginTransaction Method	No	No/Yes
-ChangeDatabase Method	Yes	No/No
-Clone Method	Yes	No/Yes
-EnlistDistributedTransaction Method	Yes	No/No
-OpenWithNewPassword Method	Yes	No/Yes
-Failover Event	Yes	No/No
-OracleFailoverEventHandler Delegate	Yes	No/No
ODP.NET Enumerations		
-FailoverEvent Enumeration	Yes	No/No
-FailoverReturnCode Enumeration	Yes	No/No
-FailoverType Enumeration	Yes	No/No
-OracleNotificationInfo Enumeration	Yes	No/No
-OracleNotificationSource Enumeration	Yes	No/No
-OracleNotificationType Enumeration	Yes	No/No

6

Oracle Data Provider for .NET Classes

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

- OracleParameterStatus Enumeration

6.1 OracleClientFactory Class

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

Class Inheritance

```
System.Object

  System.Data.Common.DbProviderFactory

    Oracle.DataAccess.Client.OracleClientFactory
```

Declaration

```
// C#
public sealed class OracleClientFactory : DbProviderFactory
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver
Assembly	`Oracle.DataAccess.dll`	`Oracle.ManagedDataAccess.dll`
Namespace	`Oracle.DataAccess.Client`	`Oracle.ManagedDataAccess.Client`
.NET Framework	3.5, 4.5, 4.6	4.5, 4.6

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 =
            DbProviderFactories.GetFactory("Oracle.DataAccess.Client");

    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);
      }
    }
  }
```

6.1.1 OracleClientFactory Members

`OracleClientFactory` members are listed in the following tables.

OracleClientFactory Field

The `OracleClientFactory` field is listed in Table 6-1

Table 6-1 OracleClientFactory Field

Property	Description
Instance	Gets an instance of the `OracleClientFactory` class

OracleClientFactory Constructor

The `OracleClientFactory` constructor is listed in Table 6-2

Table 6-2 OracleClientFactory Constructor

Property	Description
OracleClientFactory Constructor	Instantiates a new instance of `OracleClientFactory` class

OracleClientFactory Public Properties

The `OracleClientFactory` public properties are listed in Table 6-3.

Table 6-3 OracleClientFactory Public Properties

Property	Description
CanCreateDataSourceEnumerator	Indicates whether or not the `CreateDataSourceEnumerator` method is supported

OracleClientFactory Public Methods

`OracleClientFactory` Public Methods are listed in Table 6-4.

Table 6-4 OracleClientFactory Public Method

Method	Description
CreateCommand	Returns a `DbCommand` object that represents an `OracleCommand` object
CreateCommandBuilder	Returns a `DbCommandBuilder` object that represents an `OracleCommandBuilder` object
CreateConnection	Returns a `DbConnection` object that represents an `OracleConnection` object
CreateConnectionStringBuilder	Returns a `DbConnectionStringBuilder` object that represents an `OracleConnectionStringBuilder` object
CreateDataAdapter	Returns a `DbDataAdapter` object that represents an `OracleDataAdapter` object
CreateDataSourceEnumerator	Returns a `DbDataSourceEnumerator` object that represents an `OracleDataSourceEnumerator` object
CreateParameter	Returns a `DbParameter` object that represents an `OracleParameter` object
CreatePermission	Returns a `CodeAccessPermission` object that represents an `OraclePermission` object

6.1.2 OracleClientFactory Field

The `OracleClientFactory` field is listed in Table 6-5

Table 6-5 OracleClientFactory Field

Property	Description
Instance	Gets an instance of the `OracleClientFactory` class

6.1.2.1 Instance

The `Instance` field gets an instance of the OracleClientFactory class. This can be used to retrieve strongly typed data objects.

Declaration

```
// C#
public static readonly OracleClientFactory Instance
```

6.1.3 OracleClientFactory Constructor

The `OracleClientFactory` constructor creates a new instances of the `OracleClientFactory` class.

Declaration

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

6.1.4 OracleClientFactory Public Properties

The `OracleClientFactory` public properties are listed in Table 6-6.

Table 6-6 OracleClientFactory Public Properties

Property	Description
CanCreateDataSourceEnumerator	Indicates whether or not the `CreateDataSourceEnumerator` method is supported

6.1.4.1 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.

6.1.5 OracleClientFactory Public Methods

The `OracleClientFactory` public method is listed in Table 6-7.

Table 6-7 OracleClientFactory Public Method

Method	Description
CreateCommand	Returns a `DbCommand` object that represents an `OracleCommand` object
CreateCommandBuilder	Returns a `DbCommandBuilder` object that represents an `OracleCommandBuilder` object
CreateConnection	Returns a `DbConnection` object that represents an `OracleConnection` object

Table 6-7 (Cont.) OracleClientFactory Public Method

Method	Description
CreateConnectionStringBuilder	Returns a `DbConnectionStringBuilder` object that represents an `OracleConnectionStringBuilder` object
CreateDataAdapter	Returns a `DbDataAdapter` object that represents an `OracleDataAdapter` object
CreateDataSourceEnumerator	Returns a `DbDataSourceEnumerator` object that represents an `OracleDataSourceEnumerator` object
CreateParameter	Returns a `DbParameter` object that represents an `OracleParameter` object
CreatePermission	Returns a `CodeAccessPermission` object that represents an `OraclePermission` object

6.1.5.1 CreateCommand

This method returns a `DbCommand` object that represents an `OracleCommand` object.

Declaration

```
// C#
public override DbCommand CreateCommand();
```

Return Value

A `DbCommand` object that represents an `OracleCommand` object.

6.1.5.2 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.

6.1.5.3 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.

6.1.5.4 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.

6.1.5.5 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.

6.1.5.6 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.

6.1.5.7 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.

6.1.5.8 CreatePermission

This method returns a `CodeAccessPermission` object that represents an `OraclePermission` object.

Declaration

```
// C#
public override System.Security.CodeAccessPermission CreatePermission(
  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.

6.2 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

        Oracle.DataAccess.Client.OracleCommand
```

Declaration

```
// C#
public sealed class OracleCommand : DbCommand, ICloneable
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver
Assembly	`Oracle.DataAccess.dll`	`Oracle.ManagedDataAccess.dll`
Namespace	`Oracle.DataAccess.Client`	`Oracle.ManagedDataAccess.Client`
.NET Framework	3.5, 4.5, 4.6	4.5, 4.6

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();
  }
}
```

6.2.1 OracleCommand Members

`OracleCommand` members are listed in the following tables.

OracleCommand Constructors

`OracleCommand` constructors are listed in Table 6-8.

Table 6-8 OracleCommand Constructors

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

OracleCommand Static Methods

The `OracleCommand` static method is listed in Table 6-9.

Table 6-9 OracleCommand Static Method

Method	Description
`Equals`	Inherited from `System.Object` (Overloaded)

OracleCommand Properties

`OracleCommand` properties are listed in Table 6-10.

Table 6-10 OracleCommand Properties

Property	Description
AddRowid	Adds the `ROWID` as part of the select list
AddToStatementCache	Causes executed statements to be cached, when the property is set to `true` and statement caching is enabled

Table 6-10 (Cont.) OracleCommand Properties

Property	Description
ArrayBindCount	Specifies if the array binding feature is to be used and also specifies the maximum number of array elements to be bound in the `Value` property
ArrayBindRowsAffected	Returns the number of affected rows for each iteration while executing a DML using array binding
BindByName	Specifies the binding method in the collection
CommandText	Specifies the SQL statement or stored procedure to run against the Oracle database or the XML data used to store changes to the Oracle database
CommandTimeout	Specifies the number of seconds the command is allowed to execute before terminating the execution with an exception
CommandType	Specifies the command type that indicates how the `CommandText` property is to be interpreted
Connection	Specifies the `OracleConnection` object that is used to identify the connection to execute a command
`Container`	Inherited from `System.ComponentModel.Component`
DesignTimeVisible	Specifies whether or not the `OracleCommand` object is visible on designer controls.
FetchSize	Specifies the size of `OracleDataReader`'s internal cache to store result set data
ImplicitRefCursors	Specifies an array of `OracleRefCursors` mapped to an implicit resultset returned by the stored procedure. *Not available in the ODP.NET, Managed Driver*
InitialLOBFetchSize	Specifies the amount of data that the `OracleDataReader` initially fetches for LOB columns
InitialLONGFetchSize	Specifies the amount of data that the `OracleDataReader` initially fetches for `LONG` and `LONG RAW` columns
Notification	Indicates that there is a notification request for the command
NotificationAutoEnlist	Indicates whether or not to register for a continuous query notification with the database automatically when the command is executed
Parameters	Specifies the parameters for the SQL statement or stored procedure
RowSize	Specifies the amount of memory needed by the `OracleDataReader` internal cache to store one row of data
Site	Inherited from `System.ComponentModel.Component`
Transaction	Specifies the `OracleTransaction` object in which the `OracleCommand` executes *Not supported in a .NET stored procedure*

Table 6-10 (Cont.) OracleCommand Properties

Property	Description
UpdatedRowSource	Specifies how query command results are applied to the row being updated *Not supported in a .NET stored procedure*
UseEdmMapping	Indicates whether or not the command object utilizes the Entity Data Model mapping configuration values
XmlCommandType	Specifies the type of XML operation on the `OracleCommand`
XmlQueryProperties	Specifies the properties that are used when an XML document is created from the result set of a SQL query statement
XmlSaveProperties	Specifies the properties that are used when an XML document is used to save changes to the database

OracleCommand Public Methods

`OracleCommand` public methods are listed in Table 6-11.

Table 6-11 OracleCommand Public Methods

Public Method	Description
Cancel	Attempts to cancels a command that is currently executing on a particular connection
Clone	Creates a copy of `OracleCommand` object
`CreateObjRef`	Inherited from `System.MarshalByRefObject`
CreateParameter	Creates a new instance of `OracleParameter` class
Dispose	Releases any resources or memory allocated by the object
`Equals`	Inherited from `System.Object` (Overloaded)
ExecuteNonQuery	Executes a SQL statement or a command using the `XmlCommandType` and `CommandText` properties and returns the number of rows affected
ExecuteReader	Executes a command (Overloaded)
ExecuteScalar	Returns the first column of the first row in the result set returned by the query
ExecuteStream	Executes a command using the `XmlCommandType` and `CommandText` properties and returns the results in a new `Stream` object
ExecuteToStream	Executes a command using the `XmlCommandType` and `CommandText` properties and appends the results as an XML document to the existing `Stream`
ExecuteXmlReader	Executes a command using the `XmlCommandType` and `CommandText` properties and returns the result as an XML document in a .NET `XmlTextReader` object
`GetHashCode`	Inherited from `System.Object`

Table 6-11 (Cont.) OracleCommand Public Methods

Public Method	Description
`GetLifetimeService`	Inherited from `System.MarshalByRefObject`
`GetType`	Inherited from `System.Object`
`InitializeLifetimeService`	Inherited from `System.MarshalByRefObject`
Prepare	*This method is a no-op*
`ToString`	Inherited from `System.Object`

6.2.2 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.

6.2.2.1 OracleCommand()

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

Declaration

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

Remarks

Default constructor.

6.2.2.2 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.

6.2.2.3 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.

6.2.3 OracleCommand Static Methods

The `OracleCommand` static method is listed in Table 6-12.

Table 6-12 OracleCommand Static Method

Method	Description
`Equals`	Inherited from `System.Object` (Overloaded)

6.2.4 OracleCommand Properties

`OracleCommand` properties are listed in Table 6-13.

Table 6-13 OracleCommand Properties

Property	Description
AddRowid	Adds the `ROWID` as part of the select list
AddToStatementCache	Causes executed statements to be cached, when the property is set to `true` and statement caching is enabled
ArrayBindCount	Specifies if the array binding feature is to be used and also specifies the maximum number of array elements to be bound in the `Value` property
ArrayBindRowsAffected	Returns the number of affected rows for each iteration while executing a DML using array binding
BindByName	Specifies the binding method in the collection

Table 6-13 (Cont.) OracleCommand Properties

Property	Description
CommandText	Specifies the SQL statement or stored procedure to run against the Oracle database or the XML data used to store changes to the Oracle database
CommandTimeout	Specifies the number of seconds the command is allowed to execute before terminating the execution with an exception
CommandType	Specifies the command type that indicates how the `CommandText` property is to be interpreted
Connection	Specifies the `OracleConnection` object that is used to identify the connection to execute a command
Container	Inherited from `System.ComponentModel.Component`
DesignTimeVisible	Specifies whether or not the `OracleCommand` object is visible on designer controls.
FetchSize	Specifies the size of `OracleDataReader`'s internal cache to store result set data
ImplicitRefCursors	Specifies an array of `OracleRefCursors` mapped to an implicit resultset returned by the stored procedure. *Not available in the ODP.NET, Managed Driver*
InitialLOBFetchSize	Specifies the amount of data that the `OracleDataReader` initially fetches for LOB columns
InitialLONGFetchSize	Specifies the amount that of data the `OracleDataReader` initially fetches for `LONG` and `LONG RAW` columns
Notification	Indicates that there is a notification request for the command
NotificationAutoEnlist	Indicates whether or not to register for a continuous query notification with the database automatically when the command is executed
Parameters	Specifies the parameters for the SQL statement or stored procedure
RowSize	Specifies the amount of memory needed by the `OracleDataReader` internal cache to store one row of data
Site	Inherited from `System.ComponentModel.Component`
Transaction	Specifies the `OracleTransaction` object in which the `OracleCommand` executes *Not supported in a .NET stored procedure*
UpdatedRowSource	Specifies how query command results are applied to the row being updated *Not supported in a .NET stored procedure*
UseEdmMapping	Indicates whether or not the command object utilizes the Entity Data Model mapping configuration values
XmlCommandType	Specifies the type of XML operation on the `OracleCommand`
XmlQueryProperties	Specifies the properties that are used when an XML document is created from the result set of a SQL query statement

Table 6-13 (Cont.) OracleCommand Properties

Property	Description
XmlSaveProperties	Specifies the properties that are used when an XML document is used to save changes to the database

6.2.4.1 AddRowid

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

Declaration

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

Property Value

`bool`

Remarks

Default is `false`.

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

6.2.4.2 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);

    // 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();
  }
}
```

6.2.4.3 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.

6.2.4.4 ArrayBindRowsAffected

This property returns the number of affected rows for each iteration while executing a DML using array binding.

Declaration

```
// C#
public long[] ArrayBindRowsAffected ;
```

Property Value

A long type

6.2.4.5 BindByName

This property specifies the binding method in the collection.

Declaration

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

Property Value

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

Remarks

Default = false.

BindByName is ignored under the following conditions:

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

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

- ExecuteNonQuery
- ExecuteXmlReader
- ExecuteStream
- ExecuteToStream

6.2.4.6 CommandText

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

Declaration

```
// C#
public 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.

6.2.4.7 CommandTimeout

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

Declaration

```
// C#
public override int CommandTimeout {get; set;}
```

Property Value

`int`

Implements

`IDbCommand.CommandTimeout`

Exceptions

`InvalidArgument` - 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`.

6.2.4.8 CommandType

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

Declaration

```
// 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.

6.2.4.9 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`

6.2.4.10 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.

6.2.4.11 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.

6.2.4.12 ImplicitRefCursors

This property returns an array of `OracleRefCursors`, where each `OracleRefCursor` maps to an implicit resultset returned by the stored procedure.

Declaration

```
// C#
public OracleRefCursor[] ImplicitRefCursors {get; set;}
```

Property Value

An array of `OracleRefCursors`.

Remarks

This property is populated only when the stored procedure is executed through `ExecuteNonQuery` and it does not get populated in any other scenarios.

6.2.4.13 InitialLOBFetchSize

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

Declaration

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

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.

Default = 0.

The maximum value supported for `InitialLOBFetchSize` is 2 GB.

`GetOracleBlob` and `GetOracleClob` methods can be used to retrieve any LOBs no matter the `InitialLOBFetchSize` value.

6.2.4.14 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.

6.2.4.15 Notification

This instance property indicates that there is a notification request for the command.

Declaration

```
// 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.

6.2.4.16 NotificationAutoEnlist

This instance property indicates whether or not to register for a continuous query 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 continuous query notification request automatically, when the command is executed. If `NotificationAutoEnlist` is set to `true`, and the `Notification` property is set appropriately, a continuous query notification request is registered automatically; otherwise, no continuous query 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`.

6.2.4.17 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.

6.2.4.18 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 runtime, 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()`.

ODP.NET now supports values up to 32K for `VARCHAR2`, `NVARCHAR2` or `RAW` type columns in its calculation of `RowSize` value.

6.2.4.19 Transaction

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

Declaration

```
// C#
public OracleTransaction Transaction {set; 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`.

6.2.4.20 UpdatedRowSource

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

Declaration

```
// 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`.

6.2.4.21 UseEdmMapping

This property Indicates whether or not the `OracleCommand` object utilizes the Entity Data Model mapping configuration values.

Declaration

```
// C#
public bool UseEdmMapping
```

Property Value

A `bool`.

Remarks

Default is `false`.

The `UseEdmMapping` property allows user to explicitly specify that the `OracleCommand` object should use the Entity Data Model mapping configuration values. This enables use of Entity Framework Multiple Result Sets feature.

6.2.4.22 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.

6.2.4.23 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.

6.2.4.24 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.

6.2.5 OracleCommand Public Methods

`OracleCommand` public methods are listed in Table 6-14.

Table 6-14 OracleCommand Public Methods

Public Method	Description
Cancel	Attempts to cancels a command that is currently executing on a particular connection
Clone	Creates a copy of `OracleCommand` object
`CreateObjRef`	Inherited from `System.MarshalByRefObject`
CreateParameter	Creates a new instance of `OracleParameter` class
Dispose	Releases any resources or memory allocated by the object
`Equals`	Inherited from `System.Object` (Overloaded)
ExecuteNonQuery	Executes a SQL statement or a command using the `XmlCommandType` and `CommandText` properties and returns the number of rows affected
ExecuteReader	Executes a command (Overloaded)
ExecuteScalar	Returns the first column of the first row in the result set returned by the query
ExecuteStream	Executes a command using the `XmlCommandType` and `CommandText` properties and returns the results in a new `Stream` object
ExecuteToStream	Executes a command using the `XmlCommandType` and `CommandText` properties and appends the results as an XML document to the existing `Stream`
ExecuteXmlReader	Executes a command using the `XmlCommandType` and `CommandText` properties and returns the result as an XML document in a .NET `XmlTextReader` object
`GetHashCode`	Inherited from `System.Object`

Table 6-14 (Cont.) OracleCommand Public Methods

Public Method	Description
`GetLifetimeService`	Inherited from `System.MarshalByRefObject`
`GetType`	Inherited from `System.Object`
`InitializeLifetimeService`	Inherited from `System.MarshalByRefObject`
Prepare	*This method is a no-op*
`ToString`	Inherited from `System.Object`

6.2.5.1 Cancel

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

Declaration

```
// 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
// 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);
    }
    Console.WriteLine("ExecuteEvent.Set()");
    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()");
    CancelEvent.Set();
  }

  // Execution of the command with a potential of cancelling
  public void ExecuteWithinLimitedTime()
  {
    for (int i = 0; i < 5; i++)
    {
      Monitor.Enter(typeof(CancelSample));
      try
      {
        Console.WriteLine("Executing " + this.cmd.CommandText);
        ExecuteEvent.Reset();
        CancelEvent.Reset();
        t1 = new Thread(new ThreadStart(this.Execute));
        t2 = new Thread(new ThreadStart(this.Cancel));
        t1.Start();
        t2.Start();
      }
      finally
      {
        WaitHandle.WaitAll(ExecuteAndCancel);
        Monitor.Exit(typeof(CancelSample));
      }
    }
    FinishedEvent.Set();
  }
  [MTAThread]
  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;
      ExecutionCompleteEvents[2] = test3.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());
    }
  }
}
```

6.2.5.2 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.

6.2.5.3 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`

6.2.5.4 Dispose

This method releases any resources or memory allocated by the object.

Declaration

```
// C#
public void Dispose();
```

Implements

`IDisposable`

Remarks

The `Dispose` method also closes the `OracleCommand` object.

6.2.5.5 ExecuteNonQuery

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

Declaration

```
// C#
public 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

```
// C#

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

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

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

    cmd.CommandText = "update emp set sal = sal + .01 where empno=7934";

    // Auto-commit changes
    int rowsUpdated = cmd.ExecuteNonQuery();

    if (rowsUpdated > 0)
    {
      cmd.CommandText = "select sal from emp where empno=7934";
      sal = cmd.ExecuteScalar();
      Console.WriteLine("Employee sal after update: " + sal);
    }

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

Requirements

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

6.2.5.6 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.

6.2.5.7 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 a 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".

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

Example

```
// C#

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();
  }
}
```

6.2.5.8 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 `OracleRefCursors` for more details. See "OracleRefCursor Class".

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".

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`.

6.2.5.9 ExecuteScalar

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

Declaration

```
// C#
public 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

```
// C#

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

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

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

    object count = cmd.ExecuteScalar();

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

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

6.2.5.10 ExecuteStream

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

Declaration

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

Return Value

A `Stream`.

Remarks

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

- `XmlCommandType = OracleXmlCommandType.None`

  `ExecuteStream` throws an `InvalidOperationException`.

- `XmlCommandType = OracleXmlCommandType.Query`

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

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

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

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

6.2.5.11 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`.

6.2.5.12 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.

6.2.5.13 Prepare

This method is not supported.

6.3 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

        OracleDataAccess.Client.OracleCommandBuilder
```

Declaration

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

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver
Assembly	Oracle.DataAccess.dll	Oracle.ManagedDataAccess.dll
Namespace	Oracle.DataAccess.Client	Oracle.ManagedDataAccess.Client
.NET Framework	3.5, 4.5, 4.6	4.5, 4.6

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");
  }
}
```

6.3.1 OracleCommandBuilder Members

`OracleCommandBuilder` members are listed in the following tables.

OracleCommandBuilder Constructors

`OracleCommandBuilder` constructors are listed in Table 6-15.

Table 6-15 OracleCommandBuilder Constructors

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

OracleCommandBuilder Static Methods

`OracleCommandBuilder` static methods are listed in Table 6-16.

Table 6-16 OracleCommandBuilder Static Methods

Method	Description
DeriveParameters	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
`Equals`	Inherited from `System.Object` (Overloaded)

OracleCommandBuilder Properties

`OracleCommandBuilder` properties are listed in Table 6-17.

Table 6-17 OracleCommandBuilder Properties

Property	Description
`Container`	Inherited from `System.ComponentModel.Component`
CaseSensitive	Indicates whether or not double quotes are used around Oracle object names when generating SQL statements
CatalogLocation	*Not Supported*
CatalogSeparator	*Not Supported*
ConflictOption	*Not Supported*
DataAdapter	Indicates the `OracleDataAdapter` for which the SQL statements are generated
QuotePrefix	Specifies the beginning character or characters used to specify database objects whose names contain special characters such as spaces or reserved words
QuoteSuffix	Specifies the ending character or characters used to specify database objects whose names contain special characters such as spaces or reserved words

Table 6-17 (Cont.) OracleCommandBuilder Properties

Property	Description
SchemaSeparator	Specifies the character to be used for the separator between the schema identifier and other identifiers
`Site`	Inherited from `System.ComponentModel.Component`

OracleCommandBuilder Public Methods

`OracleCommandBuilder` public methods are listed in Table 6-18.

Table 6-18 OracleCommandBuilder Public Methods

Public Method	Description
`CreateObjRef`	Inherited from `System.MarshalByRefObject`
`Dispose`	Inherited from `System.ComponentModel.Component`
`Equals`	Inherited from `System.Object` (Overloaded)
GetDeleteCommand	Gets the automatically generated `OracleCommand` object that has the SQL statement (`CommandText`) perform deletions on the database (Overloaded)
`GetHashCode`	Inherited from `System.Object`
GetInsertCommand	Gets the automatically generated `OracleCommand` object that has the SQL statement (`CommandText`) perform insertions on the database (Overloaded)
`GetLifetimeService`	Inherited from `System.MarshalByRefObject`
`GetType`	Inherited from `System.Object`
GetUpdateCommand	Gets the automatically generated `OracleCommand` object that has the SQL statement (`CommandText`) perform updates on the database (Overloaded)
`InitializeLifetimeService`	Inherited from `System.MarshalByRefObject`
QuoteIdentifier	Returns the correct quoted form of the provided unquoted identifier, with any embedded quotes in the identifier properly escaped
RefreshSchema	Refreshes the database schema information used to generate `INSERT`, `UPDATE`, or `DELETE` statements
UnquoteIdentifier	Returns the correct unquoted form of the provided quoted identifier, removing any escape notation for quotes embedded in the identifier
`ToString`	Inherited from `System.Object`

OracleCommandBuilder Events

The `OracleCommandBuilder` event is listed in Table 6-19.

Table 6-19 OracleCommandBuilder Events

Event Name	Description
`Disposed`	Inherited from `System.ComponentModel.Component`

6.3.2 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.

6.3.2.1 OracleCommandBuilder()

This constructor creates an instance of the `OracleCommandBuilder` class.

Declaration

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

Remarks

Default constructor.

6.3.2.2 OracleCommandBuilder(OracleDataAdapter)

This constructor creates an instance of the `OracleCommandBuilder` class and sets the `DataAdapter` property to the provided `OracleDataAdapter` object.

Declaration

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

Parameters

- *da*

  The `OracleDataAdapter` object provided.

6.3.3 OracleCommandBuilder Static Methods

`OracleCommandBuilder` static methods are listed in Table 6-20.

Table 6-20 OracleCommandBuilder Static Methods

Method	Description
DeriveParameters	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
`Equals`	Inherited from `System.Object` (Overloaded)

6.3.3.1 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.

`DeriveParameters` incurs a database round-trip to retrieve parameter metadata prior to executing the stored procedure/function. It 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.

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

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

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

6.3.4 OracleCommandBuilder Properties

`OracleCommandBuilder` properties are listed in Table 6-21.

Table 6-21 OracleCommandBuilder Properties

Property	Description
`Container`	Inherited from `System.ComponentModel.Component`
CaseSensitive	Indicates whether or not double quotes are used around Oracle object names when generating SQL statements
CatalogLocation	*Not Supported*
CatalogSeparator	*Not Supported*
ConflictOption	*Not Supported*
DataAdapter	Indicates the `OracleDataAdapter` for which the SQL statements are generated

Table 6-21 (Cont.) OracleCommandBuilder Properties

Property	Description
QuotePrefix	Specifies the beginning character or characters used to specify database objects whose names contain special characters such as spaces or reserved words
QuoteSuffix	Specifies the ending character or characters used to specify database objects whose names contain special characters such as spaces or reserved words
SchemaSeparator	Specifies the character to be used for the separator between the schema identifier and other identifiers
`Site`	Inherited from `System.ComponentModel.Component`

6.3.4.1 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`

6.3.4.2 CatalogLocation

This property is not supported.

Declaration

```
// C#
public override CatalogLocation CatalogLocation {get; set;}
```

Exceptions

`NotSupportedException` - This property is not supported.

Remarks

This property is not supported.

6.3.4.3 CatalogSeparator

This property is not supported.

Declaration

```
// C#
public override string CatalogSeparator {get; set;}
```

Exceptions

`NotSupportedException` - This property is not supported.

Remarks

This property is not supported.

6.3.4.4 ConflictOption

This property is not supported.

Declaration

```
// C#
public override string ConflictOption {get; set;}
```

Exceptions

`NotSupportedException` - This property is not supported.

Remarks

This property is not supported.

6.3.4.5 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`

6.3.4.6 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

```
// 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.

6.3.4.7 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

```
// 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.

6.3.4.8 SchemaSeparator

This property specifies the character to be used for the separator between the schema identifier and other identifiers.

Declaration

```
// 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

```
// C#

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

class SchemaSeperatorSample
{
  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);
    }
  }
}
```

6.3.5 OracleCommandBuilder Public Methods

`OracleCommandBuilder` public methods are listed in Table 6-22.

Table 6-22 OracleCommandBuilder Public Methods

Public Method	Description
`CreateObjRef`	Inherited from `System.MarshalByRefObject`
`Dispose`	Inherited from `System.ComponentModel.Component`
`Equals`	Inherited from `System.Object` (Overloaded)
GetDeleteCommand	Gets the automatically generated `OracleCommand` object that has the SQL statement (`CommandText`) perform deletions on the database (Overloaded)
`GetHashCode`	Inherited from `System.Object`
GetInsertCommand	Gets the automatically generated `OracleCommand` object that has the SQL statement (`CommandText`) perform insertions on the database (Overloaded)

Table 6-22 (Cont.) OracleCommandBuilder Public Methods

Public Method	Description
`GetLifetimeService`	Inherited from `System.MarshalByRefObject`
`GetType`	Inherited from `System.Object`
GetUpdateCommand	Gets the automatically generated `OracleCommand` object that has the SQL statement (`CommandText`) perform updates on the database (Overloaded)
`InitializeLifetimeService`	Inherited from `System.MarshalByRefObject`
QuoteIdentifier	Returns the correct quoted form of the provided unquoted identifier, with any embedded quotes in the identifier properly escaped
RefreshSchema	Refreshes the database schema information used to generate `INSERT`, `UPDATE`, or `DELETE` statements
UnquoteIdentifier	Returns the correct unquoted form of the provided quoted identifier, removing any escape notation for quotes embedded in the identifier
`ToString`	Inherited from `System.Object`

6.3.5.1 GetDeleteCommand

Gets the automatically generated `OracleCommand` object that has the SQL statement (`CommandText`) perform deletions on the database

Overload List

- 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`.

- GetDeleteCommand(bool)

  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`.

6.3.5.2 GetDeleteCommand()

This method gets the automatically generated `OracleCommand` object that has the SQL statement (`CommandText`) perform deletions on the database when an application calls `Update()` on the `OracleDataAdapter`.

Declaration

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

Return Value

An `OracleCommand`.

Exceptions

`ObjectDisposedException` - The `OracleCommandBuilder` object is already disposed.

`InvalidOperationException` - Either the `SelectCommand` or the `DataAdapter` property is null, or the primary key cannot be retrieved from the `SelectCommand` property of the `OracleDataAdapter`.

6.3.5.3 GetDeleteCommand(bool)

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(bool useColumnsForParameterNames);
```

Parameters

- `useColumnsForParameterNames`

  If true, the method generates parameter names matching column names if possible. If false, the method binds parameters by position.

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`.

Remarks

If the bool is `true`, the method generates parameter names matching column names if possible. If `false`, the method binds parameters by position.

6.3.5.4 GetInsertCommand

Gets the automatically generated `OracleCommand` object that has the SQL statement (`CommandText`) perform insertions on the database

Overload List

- 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`.
- GetInsertCommand(bool)

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`.

6.3.5.5 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`.

6.3.5.6 GetInsertCommand(bool)

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(bool useColumnsForParameterNames);
```

Parameters

- `useColumnsForParameterNames`

  If true, the method generates parameter names matching column names if possible. If false, the method binds parameters by position.

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`.

Remarks

If the bool is `true`, the method generates parameter names matching column names if possible. If `false`, the method binds parameters by position.

6.3.5.7 GetUpdateCommand

Gets the automatically generated `OracleCommand` object that has the SQL statement (`CommandText`) perform updates on the database

Overload List

- 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`.
- GetUpdateCommand(bool)

  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`.

6.3.5.8 GetUpdateCommand()

This method gets the automatically generated `OracleCommand` object that has the SQL statement (`CommandText`) perform updates on the database when an application calls `Update()` on the `OracleDataAdapter`.

Declaration

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

Return Value

An `OracleCommand`.

Exceptions

`ObjectDisposedException` - The `OracleCommandBuilder` object is already disposed.

`InvalidOperationException` - Either the `SelectCommand` or the `DataAdapter` property is null, or the primary key cannot be retrieved from the `SelectCommand` property of the `OracleDataAdapter`.

6.3.5.9 GetUpdateCommand(bool)

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(bool useColumnsForParameterNames);
```

Parameters

- `useColumnsForParameterNames`

  If true, the method generates parameter names matching column names if possible. If false, the method binds parameters by position.

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`.

Remarks

If the bool is `true`, the method generates parameter names matching column names if possible. If `false`, the method binds parameters by position.

6.3.5.10 QuoteIdentifier

This method returns the correct quoted form of the provided unquoted identifier, with any embedded quotes in the identifier properly escaped.

Declaration

```
// 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);
  }
}
```

6.3.5.11 RefreshSchema

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

Declaration

```
// C#
public override void RefreshSchema();
```

Remarks

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

6.3.5.12 UnquoteIdentifier

This method returns the correct unquoted form of the provided quoted identifier, removing any escape notation for quotes embedded in the identifier.

Declaration

```
// 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

```
// C#

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);

    //quoteIdentifier of "US\"ER" is (\"US\"\"ER\")
    Console.WriteLine("QuotedIdentifier is {0}" , quoteIdentifier);
    string unquoteIdentifier = builder.UnquoteIdentifier(quoteIdentifier);

    //And its unquoteIdentifier is US\"ER
    Console.WriteLine("UnquotedIdentifier is {0}" , unquoteIdentifier);
  }
}
```

6.3.6 OracleCommandBuilder Events

The `OracleCommandBuilder` event is listed in Table 6-23.

Table 6-23 OracleCommandBuilder Event

Event Name	Description
`Disposed`	Inherited from `System.ComponentModel.Component`

6.4 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

        Oracle.DataAccess.Client.OracleConnection
```

Declaration

```
// C#
public sealed class OracleConnection : DbConnection, IDbConnection, ICloneable
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver
Assembly	`Oracle.DataAccess.dll`	`Oracle.ManagedDataAccess.dll`
Namespace	`Oracle.DataAccess.Client`	`Oracle.ManagedDataAccess.Client`
.NET Framework	3.5, 4.5, 4.6	4.5, 4.6

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();
      }
    }
```

6.4.1 OracleConnection Members

`OracleConnection` members are listed in the following tables.

OracleConnection Constructors

`OracleConnection` constructors are listed in Table 6-24.

Table 6-24 OracleConnection Constructors

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

OracleConnection Static Properties

The `OracleConnection` static property is listed in Table 6-26.

Table 6-25 OracleConnection Static Property

Property	Description
IsAvailable	Indicates whether or not the implicit database connection is available for use

OracleConnection Static Methods

The `OracleConnection` static methods are listed in Table 6-26.

Table 6-26 OracleConnection Static Methods

Method	Description
`Equals`	Inherited from `System.Object` (Overloaded)
ClearPool	Clears the connection pool that is associated with the provided `OracleConnection` object. *Not supported in a .NET stored procedure*
ClearAllPools	Clears all connections from all the connection pools *Not supported in a .NET stored procedure*

OracleConnection Properties

`OracleConnection` properties are listed in Table 6-27.

Table 6-27 OracleConnection Properties

Property	Description
ActionName	Specifies the action name for the connection
ClientId	Specifies the client identifier for the connection
ClientInfo	Specifies the client information for the connection
ConnectionString	Specifies connection information used to connect to an Oracle database
ConnectionTimeout	Indicates the maximum amount of time that the `Open` method can take to obtain a pooled connection before the request is terminated
ConnectionType	Determines whether a particular connection object is associated with a TimesTen database connection, an Oracle database connection, or no physical connection *Not available in ODP.NET, Managed Driver*
`Container`	Inherited from `System.ComponentModel.Component`
Database	*Not Supported*
DatabaseDomainName	Specifies the name of the database domain to which the connection is set
DatabaseName	Specifies the name of the database to which the connection is set
DataSource	Specifies the Oracle Net Services Name, Connect Descriptor, or an easy connect naming that identifies the database to which to connect
HostName	Specifies the name of the host to which the connection is set
InstanceName	Specifies the name of the instance to which the connection is set
ModuleName	Specifies the module name for the connection
ServerVersion	Specifies the version number of the Oracle database to which the `OracleConnection` has established a connection
ServiceName	Specifies the name of the service to which the connection is set
`Site`	Inherited from `System.ComponentModel.Component`
State	Specifies the current state of the connection
StatementCacheSize	Specifies the current size of the statement cache associated with this connection

OracleConnection Public Methods

`OracleConnection` public methods are listed in Table 6-28.

Table 6-28 OracleConnection Public Methods

Public Method	Description
BeginTransaction	Begins a local transaction (Overloaded) *Not supported in a .NET stored procedure for context connection*
ChangeDatabase	*Not Supported*
Clone	Creates a copy of an `OracleConnection` object *Not supported in a .NET stored procedure*

Table 6-28 (Cont.) OracleConnection Public Methods

Public Method	Description
Close	Closes the database connection
CreateCommand	Creates and returns an `OracleCommand` object associated with the `OracleConnection` object
`CreateObjRef`	Inherited from `System.MarshalByRefObject`
`Dispose`	Inherited from `System.ComponentModel.Component`
EnlistDistributedTransaction	Enables applications to explicitly enlist in a specified distributed transaction *Not supported in a .NET stored procedure*
EnlistTransaction	Enables applications to enlist in a specified distributed transaction *Not supported in a .NET stored procedure*
`Equals`	Inherited from `System.Object` (Overloaded)
FlushCache	Flushes all updates and deletes made through `REF` objects retrieved using this connection *Not available in ODP.NET, Managed Driver*
`GetHashCode`	Inherited from `System.Object`
`GetLifetimeService`	Inherited from `System.MarshalByRefObject`
GetSchema	Returns schema information for the data source of the `OracleConnection`
GetSessionInfo	Returns or refreshes the property values of the `OracleGlobalization` object that represents the globalization settings of the session (Overloaded)
`GetType`	Inherited from `System.Object`
`InitializeLifetimeService`	Inherited from `System.MarshalByRefObject`
Open	Opens a database connection with the property settings specified by the `ConnectionString`
OpenWithNewPassword	Opens a new connection with the new password *Not supported in a .NET stored procedure for context connection*
PurgeStatementCache	Flushes the Statement Cache by closing all open cursors on the database, when statement caching is enabled
SetSessionInfo	Alters the session's globalization settings with the property values provided by the `OracleGlobalization` object
SetShardingKey(OracleShardingKey, OracleShardingKey)	Enables applications to set the sharding key and super sharding key before requesting a connection *Not available in ODP.NET, Managed Driver*
`ToString`	Inherited from `System.Object`

OracleConnection Events

`OracleConnection` events are listed in Table 6-29.

Table 6-29 OracleConnection Events

Event Name	Description
Disposed	Inherited from System.ComponentModel.Component
Failover	An event that is triggered when an Oracle failover occurs *Not supported in a .NET stored procedure* *Not Available in ODP.NET, Managed Driver*
HAEvent	An event that is triggered when an HA event occurs.
InfoMessage	An event that is triggered for any message or warning sent by the database
StateChange	An event that is triggered when the connection state changes

6.4.2 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.

6.4.2.1 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

6.4.2.2 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

6.4.3 OracleConnection Static Properties

The `OracleConnection` static property is listed in Table 6-30.

Table 6-30 OracleConnection Static Property

Property	Description
IsAvailable	Indicates whether or not the implicit database connection is available for use

6.4.3.1 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 runtime 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" 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!");
  }
}
```

6.4.4 OracleConnection Static Methods

The `OracleConnection` static methods are listed in Table 6-31.

Table 6-31 OracleConnection Static Methods

Method	Description
`Equals`	Inherited from `System.Object` (Overloaded)

Table 6-31 (Cont.) OracleConnection Static Methods

Method	Description
ClearPool	Clears the connection pool that is associated with the provided `OracleConnection` object. *Not supported in a .NET stored procedure*
ClearAllPools	Clears all connections from all the connection pools *Not supported in a .NET stored procedure*

6.4.4.1 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.

Beginning with ODP.NET 12*c* Release 1 (12.1), `ClearPool` does not automatically repopulate the pool with new connections. This prevents the pool from being repopulated with invalid connections if client remains unable to connect with the database server. Developers programmatically control when the pool is repopulated by calling `OracleConnection.Open()`, which will repopulate the pool with at least the Min Pool Size number of connections.

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!");
  }
}
```

6.4.4.2 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!");
  }
}
```

6.4.5 OracleConnection Properties

`OracleConnection` properties are listed in Table 6-32

Table 6-32 OracleConnection Properties

Property	Description
ActionName	Specifies the action name for the connection
ClientId	Specifies the client identifier for the connection
ClientInfo	Specifies the client information for the connection
ConnectionString	Specifies connection information used to connect to an Oracle database
ConnectionTimeout	Indicates the maximum amount of time that the `Open` method can take to obtain a pooled connection before the request is terminated
ConnectionType	Determines whether a particular connection object is associated with a TimesTen database connection, an Oracle database connection, or no physical connection *Not available in ODP.NET, Managed Driver*
`Container`	Inherited from `System.ComponentModel.Component`
Database	*Not Supported*
DatabaseDomainName	Specifies the name of the database domain to which the connection is set
DatabaseName	Specifies the name of the database to which the connection is set

Table 6-32 (Cont.) OracleConnection Properties

Property	Description
DataSource	Specifies the Oracle Net Services Name, Connect Descriptor, or an easy connect naming that identifies the database to which to connect
HostName	Specifies the name of the host to which the connection is set
InstanceName	Specifies the name of the instance to which the connection is set
ModuleName	Specifies the module name for the connection
ServerVersion	Specifies the version number of the Oracle database to which the `OracleConnection` has established a connection
ServiceName	Specifies the name of the service to which the connection is set
`Site`	Inherited from `System.ComponentModel.Component`
State	Specifies the current state of the connection
StatementCacheSize	Specifies the current size of the statement cache associated with this connection

6.4.5.1 ActionName

This property specifies the action name for the connection.

Declaration

```
// 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.

6.4.5.2 ClientId

This property specifies the client identifier for the connection.

Declaration

```
// 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.

6.4.5.3 ClientInfo

This property specifies the client information for the connection.

Declaration

```
// 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.

6.4.5.4 ConnectionString

This property specifies connection information used to connect to an Oracle database.

Declaration

```
// 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 6-33 lists the supported connection string attributes.

Table 6-33 Supported Connection String Attributes

Connection String Attribute	Description	Default Value
`Application Continuity`	Enables database requests to automatically replay transactional or non-transactional operations in a non-disruptive and rapid manner in the event of a severed database session, which results in a recoverable error. *Not available in ODP.NET, Managed Driver*	`true`

Table 6-33 (Cont.) Supported Connection String Attributes

Connection String Attribute	Description	Default Value
`Connection Lifetime`	Minimum life time (in seconds) of the connection. This attribute specifies the lifetime of the connection in seconds. Before the `Connection` is placed back into the pool upon a `Close()` or `Dispose()` call, the lifetime of the connection is checked. If the lifetime of the connection exceeds this property value, then the connection is destroyed. If this property value is `0`, then the connection lifetime is never checked.	`0`
`Connection Timeout`	Minimum time (in seconds) to wait for a free connection from the pool. This attribute specifies the minimum amount of time (in seconds) that the `Open()` method must 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.)	`15`
`Context Connection`	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 `"context connection"` when it is set to `true`. *Supported in a .NET stored procedure only*	`false`
`Data Source`	Oracle Net Services Name, Connect Descriptor, or an easy connect naming that identifies the database to which to connect.	empty string
`DBA Privilege`	Administrative privileges `SYSDBA` or `SYSOPER`. This connection string attribute only accepts `SYSDBA` or `SYSOPER` as the attribute value. It is case-insensitive.	empty string
`Decr Pool Size`	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.	`1`

Table 6-33 (Cont.) Supported Connection String Attributes

Connection String Attribute	Description	Default Value
`Enlist`	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`.	`true`
`HA Events`	Enables ODP.NET connection pool to proactively remove connections from the pool when an Oracle database service, service member, or node goes down. This feature can be used with Global Data Services, including Oracle RAC, Data Guard, GoldenGate, and single instance deployments. "`pooling=true`" must also be set This attribute can be set to `true`, `false`, `yes`, or `no`.	`true`
`Load Balancing`	Enables ODP.NET connection pool to balance work requests across Oracle database instances based on the load balancing advisory and service goal. This feature can be used with Global Data Services, including Oracle RAC, Active Data Guard, and GoldenGate. "`pooling=true`" must also be set. This attribute can be set to `true`, `false`, `yes`, or `no`.	`true`
`Incr Pool Size`	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.	`5`
`Max Pool Size`	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.	`100`
`Metadata Pooling`	Caches metadata information. This attribute indicates whether or not metadata information for executed queries are cached for improved performance.	`True`

Table 6-33 (Cont.) Supported Connection String Attributes

Connection String Attribute	Description	Default Value
`Min Pool Size`	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.	`1`
`Password`	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 11*g* release 1 (11.1) and later.	empty string
`Persist Security Info`	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`.	`false`
`Pooling`	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`.	`true`
`Promotable Transaction`	Promotable to distributed transaction or not. If `"promotable"` is specified, the first and all subsequent connections opened in the same `TransactionScope` enlist in the same distributed transaction. If `"local"` is specified, the first connection opened in the `TransactionScope` uses a local transaction.	promotable
`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

Table 6-33 (Cont.) Supported Connection String Attributes

Connection String Attribute	Description	Default Value
`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`
`Statement Cache Size`	Statement cache enabled and cache size set size, that is, the maximum number of statements that can be cached. A value greater than zero enables statement caching and sets the cache size to itself. This value should not be greater than the value of the `OPEN_CURSORS` parameter set in the `init.ora` database configuration file.	0
`Self Tuning`	Enables or disables self-tuning for the connection. If self-tuning is enabled, then the `StatementCacheSize` settings in the registry, configuration files, and connection string are ignored. If self-tuning is disabled, then a `StatementCacheSize` value of 0 is used unless StatementCachSize is specified in the registry, configuration file, or connection string.	`true`
`User Id`	Oracle user name. This attribute specifies the Oracle user name. The case of this attribute value is preserved if it is surrounded by double quotes. For the user to connect to an Oracle database using operating system authentication, set the `User Id` to `"/"`. Any `Password` attribute setting is ignored in this case.	empty string
`Validate Connection`	Validation of connections coming from the pool. Validation causes a round-trip to the database for each connection. Therefore, it should only be used when necessary.	`false`

6.4.5.5 ConnectionTimeout

This property indicates the minimum amount of time that the `Open` method can take to obtain a pooled connection before the request is terminated.

Declaration

```
// C#
public override int ConnectionTimeout {get;}
```

Property Value

The minimum time allowed for a pooled connection request, in seconds.

Implements

`IDbConnection`

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`.

6.4.5.6 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.

6.4.5.7 Database

This property is not supported.

Declaration

```
// 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.

6.4.5.8 DatabaseDomainName

This property specifies the name of the database domain that this connection is connected to.

Declaration

```
// C#
public string DatabaseDomainName {get;}
```

Property Value

The database domain that this connection is connected to.

6.4.5.9 DatabaseName

This property specifies the name of the database that this connection is connected to.

Declaration

```
// C#
public string DatabaseName {get;}
```

Property Value

The database that this connection is connected to.

6.4.5.10 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

```
// C#
public override string DataSource {get;}
```

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.

6.4.5.11 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.

6.4.5.12 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.

6.4.5.13 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.

6.4.5.14 ServerVersion

This property specifies the version number of the Oracle database to which the `OracleConnection` has established a connection.

Declaration

```
// 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.

6.4.5.15 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.

6.4.5.16 State

This property specifies the current state of the connection.

Declaration

```
// 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`.

6.4.5.17 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.

6.4.6 OracleConnection Public Methods

`OracleConnection` public methods are listed in Table 6-34.

Table 6-34 OracleConnection Public Methods

Public Method	Description
BeginTransaction	Begins a local transaction (Overloaded) *Not supported in a .NET stored procedure for context connection*
ChangeDatabase	*Not Supported*
Clone	Creates a copy of an `OracleConnection` object *Not supported in a .NET stored procedure*
Close	Closes the database connection
CreateCommand	Creates and returns an `OracleCommand` object associated with the `OracleConnection` object
`CreateObjRef`	Inherited from `System.MarshalByRefObject`
`Dispose`	Inherited from `System.ComponentModel.Component`
EnlistDistributedTransaction	Enables applications to explicitly enlist in a specified distributed transaction *Not supported in a .NET stored procedure*
EnlistTransaction	Enables applications to enlist in a specified distributed transaction *Not supported in a .NET stored procedure*
`Equals`	Inherited from `System.Object` (Overloaded)
FlushCache	Flushes all updates and deletes made through `REF` objects retrieved using this connection *Not available in ODP.NET, Managed Driver*
`GetHashCode`	Inherited from `System.Object`

Table 6-34 (Cont.) OracleConnection Public Methods

Public Method	Description
`GetLifetimeService`	Inherited from `System.MarshalByRefObject`
GetSchema	Returns schema information for the data source of the `OracleConnection`
GetSessionInfo	Returns or refreshes the property values of the `OracleGlobalization` object that represents the globalization settings of the session (Overloaded)
`GetType`	Inherited from `System.Object`
`InitializeLifetimeService`	Inherited from `System.MarshalByRefObject`
Open	Opens a database connection with the property settings specified by the `ConnectionString`
OpenWithNewPassword	Opens a new connection with the new password *Not supported in a .NET stored procedure for context connection*
PurgeStatementCache	Flushes the Statement Cache by closing all open cursors on the database, when statement caching is enabled
SetSessionInfo	Alters the session's globalization settings with the property values provided by the `OracleGlobalization` object
SetShardingKey(OracleShardingKey, OracleShardingKey)	Enables applications to set the sharding key and super sharding key before requesting a connection *Not available in ODP.NET, Managed Driver*
`ToString`	Inherited from `System.Object`

6.4.6.1 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.

6.4.6.2 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.

6.4.6.3 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 Not Supported 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();
  }
}
```

6.4.6.4 ChangeDatabase

This method is not supported.

Declaration

```
// 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.

6.4.6.5 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();
  }
}
```

6.4.6.6 Close

This method closes the connection to the database.

Declaration

```
// C#
public override void Close();
```

Implements

`IDbConnection`

Remarks

Performs the following:

- Rolls back any pending local transactions that are not yet committed. Distributed transactions will rely on the distributed transaction coordinator on whether roll back is necessary.
- 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()`.

6.4.6.7 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();
  }
}
```

6.4.6.8 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 `ITranasction` 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 distrubuted 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();
      }
    }
  }
```

6.4.6.9 EnlistTransaction

This method enlists the connection to the specified transaction.

Declaration

```
// C#
public override void EnlistTransaction(Transaction transaction)
```

Parameters

- *transaction*

  A `System.Transactions.Transaction` object.

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 `"true"` 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. The `"enlist"` attribute should be set to `"false"` or "`dynamic`" only if the connection will never enlist in a transaction.

6.4.6.10 FlushCache

This method flushes all updates and deletes made through `REF` objects retrieved using this connection.

Declaration

```
// 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.

6.4.6.11 GetSchema

`GetSchema` methods return schema information for the data source of the `OracleConnection`.

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.

6.4.6.12 GetSchema()

This method returns schema information for the data source of the `OracleConnection`.

Declaration

```
// 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

```
// 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);
      }
    }
  }
}
```

6.4.6.13 GetSchema (string collectionName)

This method returns schema information for the data source of the `OracleConnection` using the specified string for the collection name.

Declaration

```
// 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");

        //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);
      }
    }
  }
}
```

6.4.6.14 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

```
// 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#

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);
      }
    }
  }
}
```

6.4.6.15 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.

6.4.6.16 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();
  }
}
```

6.4.6.17 GetSessionInfo(OracleGlobalization)

This method refreshes the provided `OracleGlobalization` object with the globalization settings of the session.

Declaration

```
// C#
public void GetSessionInfo(OracleGlobalization oraGlob);
```

Parameters

- *`oraGlob`*

  The `OracleGlobalization` object to be updated.

6.4.6.18 Open

This method opens a connection to an Oracle database.

Declaration

```
// C#
public overide void Open();
```

Implements

`IDbConnection`

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - The connection is already opened or the connection string is null or empty.

Remarks

The connection is obtained from the pool if connection pooling is enabled. Otherwise, a new connection is established.

It is possible that the pool does not contain any unused connections when the `Open()` method is invoked. In this case, a new connection is established.

If no connections are available within the specified connection timeout value, when the `Max Pool Size` is reached, an `OracleException` is thrown.

6.4.6.19 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.

6.4.6.20 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();
  }
}
```

6.4.6.21 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.DateFormat, cmd.ExecuteScalar());

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

6.4.6.22 SetShardingKey(OracleShardingKey, OracleShardingKey)

This instance method enables applications to set the sharding key and the super sharding key before requesting a connection.

Declaration

```
// C#
public void SetShardingKey(OracleShardingKey shardKey, OracleShardingKey
superShardingKey);
```

Exceptions

`InvalidArgumentException` – An invalid Oracle sharding key is supplied.

`InvalidOperationException` – The method is invoked when the connection is in an `Open` state.

Remarks

This method sets the sharding key and the super sharding key that is to be used for returning the proper connection upon the `Open` method invocation.

This method can only be invoked when the connection is in a `Closed` state.

Example

```
// C#

using System;
using Oracle.DataAccess.Client;

class Sharding
{
  static void Main()
  {
    OracleConnection con = new OracleConnection("user id=hr;password=hr;Data
Source=orcl;");
    //Setting a shard key
    OracleShardingKey shardingKey = new OracleShardingKey(OracleDbType.Int32, 123);
    //Setting a second shard key value for a composite key
    shardingKey.SetShardingKey(OracleDbType.Varchar2, "gold");
    //Creating and setting the super shard key
    OracleShardingKey superShardingKey = new OracleShardingKey();
    superShardingKey.SetShardingKey(OracleDbType.Int32, 1000);

    //Setting super sharding key and sharding key on the connection
    con.SetShardingKey(shardingKey, superShardingKey);
    con.Open();

    //perform SQL query
  }
}
```

6.4.7 OracleConnection Events

`OracleConnection` events are listed in Table 6-35.

Table 6-35 OracleConnection Events

Event Name	Description
`Disposed`	Inherited from `System.ComponentModel.Component`
Failover	An event that is triggered when an Oracle failover occurs *Not supported in a .NET stored procedure* *Not available in ODP.NET, Managed Driver*
HAEvent	An event that is triggered when an HA event occurs.
InfoMessage	An event that is triggered for any message or warning sent by the database
StateChange	An event that is triggered when the connection state changes

6.4.7.1 Failover

This event is triggered when an Oracle failover occurs.

Declaration

```
// C#
public event OracleFailoverEventHandler Failover;
```

Event Data

The event handler receives an `OracleFailoverEventArgs` object which exposes the following properties containing information about the event.

- `FailoverType`

  Indicates the type of the failover.
- `FailoverEvent`

  Indicates the state of the failover.

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.

6.4.7.2 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.

6.4.7.3 InfoMessage

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

Declaration

```
// C#
public event OracleInfoMessageEventHandler InfoMessage;
```

Event Data

The event handler receives an `OracleInfoMessageEventArgs` object which exposes the following properties containing information about the event.

- `Errors`

  The collection of errors generated by the data source.
- `Message`

  The error text generated by the data source.
- `Source`

  The name of the object that generated the error.

Remarks

In order to respond to warnings and messages from the database, the client should create an `OracleInfoMessageEventHandler` delegate to listen to this event.

6.4.7.4 StateChange

This event is triggered when the connection state changes.

Declaration

```
// 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`.

6.5 OracleConnectionStringBuilder Class

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

Class Inheritance

```
System.Object

  System.Data.Common.DbConnectionStringBuilder

    Oracle.DataAccess.Client.OracleConnectionStringBuilder
```

Declaration

```
// C#
public sealed class OracleConnectionStringBuilder : DbConnectionStringBuilder
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver
Assembly	`Oracle.DataAccess.dll`	`Oracle.ManagedDataAccess.dll`
Namespace	`Oracle.DataAccess.Client`	`Oracle.ManagedDataAccess.Client`
.NET Framework	3.5, 4.5, 4.6	4.5, 4.6

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 = '"scoTT"'` 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>`.

5. 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

```
// C#

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

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 nott 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} \n"
          ,keys.Current ,values.Current);
      }
    }
  }
```

6.5.1 OracleConnectionStringBuilder Members

`OracleConnectionStringBuilder` members are listed in the following tables.

OracleConnectionStringBuilder Constructors

`OracleConnectionStringBuilder` constructors are listed in Table 6-36.

Table 6-36 OracleConnectionStringBuilder Constructors

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

OracleConnectionStringBuilder Public Properties

`OracleConnectionStringBuilder` instance properties are listed in Table 6-37.

Table 6-37 OracleConnectionStringBuilder Public Properties

Properties	Description
`BrowsableConnectionString`	Inherited from `System.Data.Common.DbConnectionStringBuilder`
ConnectionLifeTime	Specifies the value corresponding to the `Connection Lifetime` attribute in the `ConnectionString` property
`ConnectionString`	Inherited from `System.Data.Common.DbConnectionStringBuilder`
ConnectionTimeout	Specifies the value corresponding to the `Connection Timeout` attribute in the `ConnectionString` property
ContextConnection	Specifies the value corresponding to the `Context Connection` attribute in the `ConnectionString` property
`Count`	Inherited from `System.Data.Common.DbConnectionStringBuilder`
DataSource	Specifies the value corresponding to the `Data Source` attribute in the `ConnectionString` property
DBAPrivilege	Specifies the value corresponding to the `DBA Privilege` attribute in the `ConnectionString` property
DecrPoolSize	Specifies the value corresponding to the `Decr Pool Size` attribute in the `ConnectionString` property
Enlist	Specifies the value corresponding to the `Enlist` attribute in the `ConnectionString` property
HAEvents	Specifies the value corresponding to the `HA Events` attribute in the `ConnectionString` property
IncrPoolSize	Specifies the value corresponding to the `Incr Pool Size` attribute in the `ConnectionString` property
IsFixedSize	Indicates whether or not the Connection String Builder has a fixed size

Table 6-37 (Cont.) OracleConnectionStringBuilder Public Properties

Properties	Description
`IsReadOnly`	Inherited from `System.Data.Common.DbConnectionStringBuilder`
Item	Specifies the value associated with the specified attribute
Keys	Specifies a collection of attributes contained in the Connection String Builder
LoadBalancing	Specifies the value corresponding to the `Load Balancing` attribute in the `ConnectionString` property
MaxPoolSize	Specifies the value corresponding to the `Max Pool Size` attribute in the `ConnectionString` property
MetadataPooling	Specifies the value that corresponds to the `Metadata Pooling` attribute in the `ConnectionString` property
MinPoolSize	Specifies the value corresponding to the `Min Pool Size` attribute in the `ConnectionString` property
Password	Specifies the value corresponding to the `Password` attribute in the `ConnectionString` property
PersistSecurityInfo	Specifies the value corresponding to the `Persist Security Info` attribute in the `ConnectionString` property
Pooling	Specifies the value corresponding to the `Pooling` attribute in the `ConnectionString` property
PromotableTransaction	Specifies the value corresponding to the `PromotableTransaction` attribute in the `ConnectionString` property *This property has been deprecated in 12.2.0.1. It will be desupported in a future release.*
ProxyPassword	Specifies the value corresponding to the `Proxy User Id` attribute in the `ConnectionString` property
ProxyUserId	Specifies the value corresponding to the `Proxy User Id` attribute in the `ConnectionString` property
SelfTuning	Specifies the value corresponding to the `Self Tuning` attribute in the `ConnectionString` property
StatementCachePurge	Specifies the value corresponding to the `Statement Cache Purge` attribute in the `ConnectionString` property
StatementCacheSize	Specifies the value corresponding to the `Statement Cache Size` attribute in the `ConnectionString` property
UserID	Specifies the value corresponding to the `User Id` attribute in the `ConnectionString` property
ValidateConnection	Specifies the value corresponding to the `Validate Connection` attribute in the `ConnectionString` property
Values	Specifies a collection of values contained in the Connection String Builder

OracleConnectionStringBuilder Public Methods

`OracleConnectionStringBuilder` instance methods are listed in Table 6-38.

Table 6-38 OracleConnectionStringBuilder Public Methods

Methods	Description
`Add`	Inherited from `System.Data.Common.DbConnectionStringBuilder`
Clear	Clears the connection string contents
ContainsKey	Indicates whether or not a specific attribute in the connection string is supported by ODP.NET
`EquivalentTo`	Inherited from `System.Data.Common.DbConnectionStringBuilder`
Remove	Removes the entry corresponding to the specified attribute from the connection string
`ShouldSerialize`	Inherited from `System.Data.Common.DbConnectionStringBuilder`
`ToString`	Inherited from `System.Data.Common.DbConnectionStringBuilder`
TryGetValue	Returns the value corresponding to the supplied attribute, as an output parameter

6.5.2 OracleConnectionStringBuilder Constructors

`OracleConnectionStringBuilder` constructors instantiate new instances of the `OracleConnectionStringBuilder` class.

Overload List:

- OracleConnectionStringBuilder()

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

- OracleConnectionStringBuilder(string)

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

6.5.2.1 OracleConnectionStringBuilder()

This constructor instantiates a new instance of the `OracleConnectionStringBuilder` class.

Declaration

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

Remarks

The `ConnectionString` property is empty after the object is created.

6.5.2.2 OracleConnectionStringBuilder(string)

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

Declaration

```
// C#
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.

6.5.3 OracleConnectionStringBuilder Public Properties

`OracleConnectionStringBuilder` public properties are listed in Table 6-39.

Table 6-39 OracleConnectionStringBuilder Public Properties

Properties	Description
`BrowsableConnectionString`	Inherited from `System.Data.Common.DbConnectionStringBuilder`
ConnectionLifeTime	Specifies the value corresponding to the `Connection Lifetime` attribute in the `ConnectionString` property
`ConnectionString`	Inherited from `System.Data.Common.DbConnectionStringBuilder`
ConnectionTimeout	Specifies the value corresponding to the `Connection Timeout` attribute in the `ConnectionString` property
ContextConnection	Specifies the value corresponding to the `Context Connection` attribute in the `ConnectionString` property
`Count`	Inherited from `System.Data.Common.DbConnectionStringBuilder`
DataSource	Specifies the value corresponding to the `Data Source` attribute in the `ConnectionString` property
DBAPrivilege	Specifies the value corresponding to the `DBA Privilege` attribute in the `ConnectionString` property
DecrPoolSize	Specifies the value corresponding to the `Decr Pool Size` attribute in the `ConnectionString` property

Table 6-39 (Cont.) OracleConnectionStringBuilder Public Properties

Properties	Description
Enlist	Specifies the value corresponding to the `Enlist` attribute in the `ConnectionString` property
HAEvents	Specifies the value corresponding to the `HA Events` attribute in the `ConnectionString` property
IncrPoolSize	Specifies the value corresponding to the `Incr Pool Size` attribute in the `ConnectionString` property
IsFixedSize	Indicates whether or not the Connection String Builder has a fixed size
`IsReadOnly`	Inherited from `System.Data.Common.DbConnectionStringBuilder`
Item	Specifies the value associated with the specified attribute
Keys	Specifies a collection of attributes contained in the Connection String Builder
LoadBalancing	Specifies the value corresponding to the `Load Balancing` attribute in the `ConnectionString` property
MaxPoolSize	Specifies the value corresponding to the `Max Pool Size` attribute in the `ConnectionString` property
MetadataPooling	Specifies the value that corresponds to the `Metadata Pooling` attribute in the `ConnectionString` property
MinPoolSize	Specifies the value corresponding to the `Min Pool Size` attribute in the `ConnectionString` property
Password	Specifies the value corresponding to the `Password` attribute in the `ConnectionString` property
PersistSecurityInfo	Specifies the value corresponding to the `Persist Security Info` attribute in the `ConnectionString` property
Pooling	Specifies the value corresponding to the `Pooling` attribute in the `ConnectionString` property
PromotableTransaction	Specifies the value corresponding to the `PromotableTransaction` attribute in the `ConnectionString` property *This property has been deprecated in 12.2.0.1. It will be desupported in a future release.*
ProxyPassword	Specifies the value corresponding to the `Proxy User Id` attribute in the `ConnectionString` property
ProxyUserId	Specifies the value corresponding to the `Proxy User Id` attribute in the `ConnectionString` property
SelfTuning	Specifies the value corresponding to the `Self Tuning` attribute in the `ConnectionString` property
StatementCachePurge	Specifies the value corresponding to the `Statement Cache Purge` attribute in the `ConnectionString` property
StatementCacheSize	Specifies the value corresponding to the `Statement Cache Size` attribute in the `ConnectionString` property

Table 6-39 (Cont.) OracleConnectionStringBuilder Public Properties

Properties	Description
UserID	Specifies the value corresponding to the `User Id` attribute in the `ConnectionString` property
ValidateConnection	Specifies the value corresponding to the `Validate Connection` attribute in the `ConnectionString` property
Values	Specifies a collection of values contained in the Connection String Builder

6.5.3.1 ConnectionLifeTime

This property specifies the value corresponding to the `Connection LifeTime` attribute in the `ConnectionString` property.

Declaration

```
// 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.

6.5.3.2 ConnectionTimeout

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.

6.5.3.3 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.

6.5.3.4 DataSource

This property specifies the value corresponding to the `Data Source` attribute in the `ConnectionString` property.

Declaration

```
// 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.

6.5.3.5 DBAPrivilege

This property specifies the value corresponding to the `DBA Privilege` attribute in the `ConnectionString` property.

Declaration

```
// 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.

6.5.3.6 DecrPoolSize

This property specifies the value corresponding to the `Decr Pool Size` attribute in the `ConnectionString` property.

Declaration

```
// 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.

6.5.3.7 Enlist

This property specifies the value corresponding to the `Enlist` attribute in the `ConnectionString` property.

Declaration

```
 // 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.

6.5.3.8 HAEvents

This property specifies the value corresponding to the `HA Events` 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.

6.5.3.9 IncrPoolSize

This property specifies the value corresponding to the `Incr Pool Size` 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`.

Remarks

When an `OracleConnectionStringBuilder` instance is created, this property is set to the default value of the corresponding connection string attribute.

6.5.3.10 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.

6.5.3.11 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.

6.5.3.12 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.

6.5.3.13 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.

6.5.3.14 MaxPoolSize

This property specifies the value corresponding to the `Max Pool Size` attribute in the `ConnectionString` property.

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.

6.5.3.15 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.

6.5.3.16 MinPoolSize

This property specifies the value corresponding to the `Min Pool Size` attribute in the `ConnectionString` property.

Declaration

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

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.

6.5.3.17 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.

6.5.3.18 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.

6.5.3.19 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.

6.5.3.20 PromotableTransaction

This property specifies the value corresponding to the `PromotableTransaction` attribute in the `ConnectionString` property.

Declaration

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

Property Value

A string that represents the value of the supplied attribute

6.5.3.21 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.

6.5.3.22 ProxyUserId

This property specifies the value corresponding to the `Proxy User Id` 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.

6.5.3.23 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.

6.5.3.24 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.

6.5.3.25 StatementCacheSize

This property specifies the value corresponding to the `Statement Cache Size` attribute in the `ConnectionString` property.

Declaration

```
// 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.

6.5.3.26 UserID

This property specifies the value corresponding to the `User Id` attribute in the `ConnectionString` property.

Declaration

```
// 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.

6.5.3.27 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.

6.5.3.28 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.

6.5.4 OracleConnectionStringBuilder Public Methods

`OracleConnectionStringBuilder` public methods are listed in Table 6-40.

Table 6-40 OracleConnectionStringBuilder Public Methods

Methods	Description
`Add`	Inherited from `System.Data.Common.DbConnectionStringBuilder`
Clear	Clears the connection string contents
ContainsKey	Indicates whether or not a specific attribute in the connection string is supported by ODP.NET
`EquivalentTo`	Inherited from `System.Data.Common.DbConnectionStringBuilder`
Remove	Removes the entry corresponding to the specified attribute from the connection string
`ShouldSerialize`	Inherited from `System.Data.Common.DbConnectionStringBuilder`
`ToString`	Inherited from `System.Data.Common.DbConnectionStringBuilder`
TryGetValue	Returns the value corresponding to the supplied attribute, as an output parameter

6.5.4.1 Clear

This method clears the connection string contents.

Declaration

```
// C#
public override void Clear();
```

Remarks

All key/value pairs are removed from the `OracleConnectionStringBuilder` object and the `ConnectionString` property is set to Empty.

6.5.4.2 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.

6.5.4.3 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.

6.5.4.4 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.

6.6 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.DataAdapter
      System.Data.Common.DbDataAdapter
        Oracle.DataAccess.Client.OracleDataAdapter
```

Declaration

```
// C#
public sealed class OracleDataAdapter : DbDataAdapter, IDbDataAdapter
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver
Assembly	`Oracle.DataAccess.dll`	`Oracle.ManagedDataAccess.dll`
Namespace	`Oracle.DataAccess.Client`	`Oracle.ManagedDataAccess.Client`
.NET Framework	3.5, 4.5, 4.6	4.5, 4.6

Thread Safety

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

Example

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

```
// C#

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");
    }
  }
```

See Also:

- "Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"
- OracleDataAdapter Members
- OracleDataAdapter Constructors
- OracleDataAdapter Static Methods
- OracleDataAdapter Properties
- OracleDataAdapter Public Methods
- OracleDataAdapter Events

6.6.1 OracleDataAdapter Members

`OracleDataAdapter` members are listed in the following tables.

OracleDataAdapter Constructors

`OracleDataAdapter` constructors are listed in Table 6-41.

Table 6-41 OracleDataAdapter Constructors

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

OracleDataAdapter Static Methods

The `OracleDataAdapter` static method is listed in Table 6-42.

Table 6-42 OracleDataAdapter Static Method

Method	Description
`Equals`	Inherited from `System.Object` (Overloaded)

OracleDataAdapter Properties

`OracleDataAdapter` properties are listed in Table 6-43.

Table 6-43 OracleDataAdapter Properties

Property	Description
`AcceptChangesDuringFill`	Inherited from `System.Data.Common.DataAdapter`
`Container`	Inherited from `System.ComponentModel.Component`
`ContinueUpdateOnError`	Inherited from `System.Data.Common.DataAdapter`
DeleteCommand	A SQL statement or stored procedure to delete rows from an Oracle database
IdentityInsert	Determines whether or not to insert identity column values in the `DataSet` into the database when the `Update` method is invoked. *Not available in the ODP.NET, Managed Driver*
IdentityUpdate	Determines whether or not to update identity column values in the `DataSet` into the database when the `Update` method is invoked. *Not available in the ODP.NET, Managed Driver*
InsertCommand	A SQL statement or stored procedure to insert new rows into an Oracle database
`MissingMappingAction`	Inherited from `System.Data.Common.DataAdapter`
`MissingSchemaAction`	Inherited from `System.Data.Common.DataAdapter`
Requery	Determines whether or not the `SelectCommand` is reexecuted on the next call to `Fill`
ReturnProviderSpecificTypes	Determines if the `Fill` method returns ODP.NET-specific values or .NET common language specification values
SafeMapping	Creates a mapping between column names in the result set to .NET types, to preserve the data *Not available in the ODP.NET, Managed Driver*
SelectCommand	A SQL statement or stored procedure that returns a single or multiple result set
`Site`	Inherited from `System.ComponentModel.Component`
`TableMappings`	Inherited from `System.Data.Common.DataAdapter`
UpdateBatchSize	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
UpdateCommand	A SQL statement or stored procedure to update rows from the `DataSet` to an Oracle database

OracleDataAdapter Public Methods

`OracleDataAdapter` public methods are listed in Table 6-44.

Table 6-44 OracleDataAdapter Public Methods

Public Method	Description
`CreateObjRef`	Inherited from `System.MarshalByRefObject`
`Dispose`	Inherited from `System.ComponentModel.Component`
`Equals`	Inherited from `System.Object` (Overloaded)
Fill	Adds or refreshes rows in the `DataSet` to match the data in the Oracle database (Overloaded)
`FillSchema`	Inherited from `System.Data.Common.DbDataAdapter`
`GetFillParameters`	Inherited from `System.Data.Common.DbDataAdapter`
`GetHashCode`	Inherited from `System.Object`
`GetLifetimeService`	Inherited from `System.MarshalByRefObject`
`GetType`	Inherited from `System.Object`
`InitializeLifetimeService`	Inherited from `System.MarshalByRefObject`
`ToString`	Inherited from `System.Object`
`Update`	Inherited from `System.Data.Common.DbDataAdapter`

OracleDataAdapter Events

`OracleDataAdapter` events are listed in Table 6-45.

Table 6-45 OracleDataAdapter Events

Event Name	Description
`Disposed`	Inherited from `System.ComponentModel.Component`
`FillError`	Inherited from `System.Data.Common.DbDataAdapter`
RowUpdated	This event is raised when row(s) have been updated by the `Update()` method
RowUpdating	This event is raised when row data are about to be updated to the database

6.6.2 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`.

6.6.2.1 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`

6.6.2.2 OracleDataAdapter(OracleCommand)

This constructor creates an instance of an `OracleDataAdapter` class with the provided `OracleCommand` as the `SelectCommand`.

Declaration

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

Parameters

- *`selectCommand`*

  The `OracleCommand` that is to be set as the `SelectCommand` property.

Remarks

Initial values are set for the following `OracleDataAdapter` properties as indicated:

- `MissingMappingAction = MissingMappingAction.Passthrough`
- `MissingSchemaAction = MissingSchemaAction.Add`

6.6.2.3 OracleDataAdapter(string, OracleConnection)

This constructor creates an instance of an `OracleDataAdapter` class with the provided `OracleConnection` object and the command text for the `SelectCommand`.

Declaration

```
// C#
public OracleDataAdapter(string selectCommandText, OracleConnection
    selectConnection);
```

Parameters

- *`selectCommandText`*

  The string that is set as the `CommandText` of the `SelectCommand` property of the `OracleDataAdapter`.

- *`selectConnection`*

  The `OracleConnection` to connect to the Oracle database.

Remarks

The `OracleDataAdapter` opens and closes the connection, if it is not already open. If the connection is open, it must be explicitly closed.

Initial values are set for the following `OracleDataAdapter` properties as indicated:

- `MissingMappingAction = MissingMappingAction.Passthrough`
- `MissingSchemaAction = MissingSchemaAction.Add`

6.6.2.4 OracleDataAdapter(string, string)

This constructor creates an instance of an `OracleDataAdapter` class with the provided connection string and the command text for the `SelectCommand`.

Declaration

```
// C#
public OracleDataAdapter(string selectCommandText, string
   selectConnectionString);
```

Parameters

- *`selectCommandText`*

  The string that is set as the `CommandText` of the `SelectCommand` property of the `OracleDataAdapter`.

- *`selectConnectionString`*

  The connection string.

Remarks

Initial values are set for the following `OracleDataAdapter` properties as indicated:

- `MissingMappingAction = MissingMappingAction.Passthrough`
- `MissingSchemaAction = MissingSchemaAction.Add`

6.6.3 OracleDataAdapter Static Methods

The `OracleDataAdapter` static method is listed in Table 6-46.

Table 6-46 OracleDataAdapter Static Method

Method	Description
`Equals`	Inherited from `System.Object` (Overloaded)

6.6.4 OracleDataAdapter Properties

`OracleDataAdapter` properties are listed in Table 6-47.

Table 6-47 OracleDataAdapter Properties

Property	Description
`AcceptChangesDuringFill`	Inherited from `System.Data.Common.DataAdapter`
`Container`	Inherited from `System.ComponentModel.Component`
`ContinueUpdateOnError`	Inherited from `System.Data.Common.DataAdapter`
DeleteCommand	A SQL statement or stored procedure to delete rows from an Oracle database
IdentityInsert	Determines whether or not to insert identity column values in the `DataSet` into the database when the `Update` method is invoked. *Not available in the ODP.NET, Managed Driver*
IdentityUpdate	Determines whether or not to update identity column values in the `DataSet` into the database when the `Update` method is invoked. *Not available in the ODP.NET, Managed Driver*
InsertCommand	A SQL statement or stored procedure to insert new rows into an Oracle database
`MissingMappingAction`	Inherited from `System.Data.Common.DataAdapter`
`MissingSchemaAction`	Inherited from `System.Data.Common.DataAdapter`
Requery	Determines whether or not the `SelectCommand` is reexecuted on the next call to `Fill`
ReturnProviderSpecificTypes	Determines if the `Fill` method returns ODP.NET-specific values or .NET common language specification values
SafeMapping	Creates a mapping between column names in the result set to .NET types, to preserve the data *Not Available in ODP.NET, Managed Driver*
SelectCommand	A SQL statement or stored procedure that returns a single or multiple result set
`Site`	Inherited from `System.ComponentModel.Component`
`TableMappings`	Inherited from `System.Data.Common.DataAdapter`

Table 6-47 (Cont.) OracleDataAdapter Properties

Property	Description
UpdateBatchSize	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
UpdateCommand	A SQL statement or stored procedure to update rows from the `DataSet` to an Oracle database

6.6.4.1 DeleteCommand

This property is a SQL statement or stored procedure to delete rows from an Oracle database.

Declaration

```
// C#
public OracleCommand DeleteCommand {get; set;}
```

Property Value

An `OracleCommand` used during the `Update` call to delete rows from tables in the Oracle database, corresponding to the deleted rows in the `DataSet`.

Remarks

Default = `null`

If there is primary key information in the `DataSet`, the `DeleteCommand` can be automatically generated using the `OracleCommandBuilder`, if no command is provided for this.

6.6.4.2 IdentityInsert

When inserting `DataSet` data into the database, this property indicates whether the database generates the inserted row's identity column value or `DataSet` supplies this value.

Declaration

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

Property Value

When set to `true`, ODP.NET inserts `DataSet` identity column values into the database. When set to false, the database determines the inserted identity column values.

Remarks

This property applies only to identity columns of type `GENERATED BY DEFAULT` and `GENERATED BY DEFAULT ON NULL`. Identity column of type `GENERATED ALWAYS` will ignore this property and will always use database generated values.

When set to false, the server will generate an identity value for the row. That generated identity value returns back to the client to update the `DataSet` value.

When this property is set to true for the `GENERATED BY DEFAULT` case and the application attempts to insert a `NULL` value into the database's identity column, the `NOT NULL` constraint is violated and an error occurs. ODP.NET will then allow the database to generate the identity column value and return the generated value to the `DataSet`.

The default value for this property is `false`.

6.6.4.3 IdentityUpdate

When updating `DataSet` data into the database, this property indicates whether to replace the database's identity column values with values of the `DataSet` or leave the current values unchanged.

Declaration

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

Property Value

When set to true, ODP.NET updates the database identity column values with the values of the `DataSet`. When set to false, the database identity columns are left unchanged.

Remarks

This property applies only to identity columns of type `GENERATED BY DEFAULT` and `GENERATED BY DEFAULT ON NULL`. In the case of type `GENERATED ALWAYS`, this property will be ignored and the database will always retain its current identity values.

When set to false, the existing identity column value in the server is returned to the `DataSet`.

When this property is set to true for the `GENERATED BY DEFAULT` and `GENERATED BY DEFAULT ON NULL` cases and the application attempts to update the database's identity column with a `NULL` value, the `NOT NULL` constraint is violated and an error occurs. ODP.NET then does not update the identity column value and instead returns the existing identity column value of the database to the `DataSet`.

The default value for this property is `false`.

6.6.4.4 InsertCommand

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

Declaration

```
// C#
public OracleCommand InsertCommand {get; set;}
```

Property Value

An `OracleCommand` used during the `Update` call to insert rows into a table, corresponding to the inserted rows in the `DataSet`.

Remarks

Default = `null`

If there is primary key information in the `DataSet`, the `InsertCommand` can be automatically generated using the `OracleCommandBuilder`, if no command is provided for this property.

6.6.4.5 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`.

6.6.4.6 ReturnProviderSpecificTypes

This property determines if the `Fill` method returns ODP.NET-specific values or .NET common language specification compliant values.

Declaration

```
// C#
public Boolean ReturnProviderSpecificTypes {get; set;}
```

Property Value

A value that indicates whether or not the `Fill` method returns ODP.NET-specific values.

Starting with ODP.NET 12.1.0.2, when set to `true` and `LegacyEntireLOBFetch = 0` (default), BLOB and CLOB column values are represented in the `DataTable` as `OracleBlob` and `OracleClob`, respectively.

A value of `false` indicates that the `Fill` method returns .NET common language specification compliant values. The default is `false`.

6.6.4.7 SafeMapping

This property creates a mapping between column names in the result set to .NET types that represent column values in the `DataSet`, to preserve the data.

Declaration

```
// C#
public Hashtable SafeMapping {get; set;}
```

Property Value

A 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".

6.6.4.8 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.

6.6.4.9 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.

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.

6.6.4.10 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.

6.6.5 OracleDataAdapter Public Methods

`OracleDataAdapter` public methods are listed in Table 6-48.

Table 6-48 OracleDataAdapter Public Methods

Public Method	Description
`CreateObjRef`	Inherited from `System.MarshalByRefObject`
`Dispose`	Inherited from `System.ComponentModel.Component`
`Equals`	Inherited from `System.Object` (Overloaded)
Fill	Adds or refreshes rows in the `DataSet` to match the data in the Oracle database (Overloaded)
`FillSchema`	Inherited from `System.Data.Common.DbDataAdapter`
`GetFillParameters`	Inherited from `System.Data.Common.DbDataAdapter`
`GetHashCode`	Inherited from `System.Object`
`GetLifetimeService`	Inherited from `System.MarshalByRefObject`
`GetType`	Inherited from `System.Object`
`InitializeLifetimeService`	Inherited from System.`MarshalByRefObject`
`ToString`	Inherited from `System.Object`
`Update`	Inherited from `System.Data.Common.DbDataAdapter`

6.6.5.1 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.

6.6.5.2 Fill(DataTable, OracleRefCursor)

This method adds or refreshes rows in the specified `DataTable` to match those in the provided `OracleRefCursor` object.

Declaration

```
// C#
public int Fill(DataTable dataTable, OracleRefCursor refCursor);
```

Parameters

- *`dataTable`*

  The `DataTable` object being populated.
- *`refCursor`*

  The `OracleRefCursor` that rows are being retrieved from.

Return Value

The number of rows added to or refreshed in the `DataTable`.

Exceptions

`ArgumentNullException` - The *`dataTable`* or *`refCursor`* parameter is null.

`InvalidOperationException` - The `OracleRefCursor` is already being used to fetch data.

`NotSupportedException` - The `SafeMapping` type is not supported.

Remarks

No schema or key information is provided, even if the `Fill` method is called with `MissingSchemaAction` set to `MissingSchemaAction.AddWithKey`.

6.6.5.3 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`.

6.6.5.4 Fill(DataSet, string, OracleRefCursor)

This method adds or refreshes rows in the specified source table of the `DataSet` to match those in the provided `OracleRefCursor` object.

Declaration

```
// C#
public int Fill(DataSet dataSet, string srcTable, OracleRefCursor
   refCursor);
```

Parameters

- *`dataSet`*

  The `DataSet` object being populated.
- *`srcTable`*

  The name of the source table used in the table mapping.
- *`refCursor`*

  The `OracleRefCursor` that rows are being retrieved from.

Return Value

Returns the number of rows added or refreshed into the `DataSet`.

Exceptions

`ArgumentNullException` - The *`dataSet`* or *`refCursor`* parameter is null.

`InvalidOperationException` - The `OracleRefCursor` is already being used to fetch data or the source table name is invalid.

`NotSupportedException` - The `SafeMapping` type is not supported.

Remarks

No schema or key information is provided, even if the `Fill` method is called with `MissingSchemaAction` set to `MissingSchemaAction.AddWithKey`.

6.6.5.5 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`.

6.6.6 OracleDataAdapter Events

`OracleDataAdapter` events are listed in Table 6-49.

Table 6-49 OracleDataAdapter Events

Event Name	Description
`Disposed`	Inherited from `System.ComponentModel.Component`
`FillError`	Inherited from `System.Data.Common.DbDataAdapter`
RowUpdated	This event is raised when row(s) have been updated by the `Update()` method
RowUpdating	This event is raised when row data are about to be updated to the database

6.6.6.1 RowUpdated

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

Declaration

```
// C#
public event OracleRowUpdatedEventHandler RowUpdated;
```

Event Data

The event handler receives an `OracleRowUpdatedEventArgs` object which exposes the following properties containing information about the event.

- `Command`

  The `OracleCommand` executed during the `Update`.
- `Errors` (inherited from `RowUpdatedEventArgs`)

  The exception, if any, is generated during the `Update`.
- `RecordsAffected` (inherited from `RowUpdatedEventArgs`)

  The number of rows modified, inserted, or deleted by the execution of the `Command`.
- `Row` (inherited from `RowUpdatedEventArgs`)

  The `DataRow` sent for `Update`.
- `StatementType` (inherited from `RowUpdatedEventArgs`)

  The type of SQL statement executed.
- `Status` (inherited from `RowUpdatedEventArgs`)

  The `UpdateStatus` of the `Command`.
- `TableMapping` (inherited from `RowUpdatedEventArgs`)

  The `DataTableMapping` used during the `Update`.

Example

The following example shows how to use the `RowUpdating` and `RowUpdated` events.

```
// C#

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

class RowUpdatedSample
{
  // Event handler for RowUpdating event
  protected static void OnRowUpdating(object sender,
                                     OracleRowUpdatingEventArgs e)
  {
    Console.WriteLine("Row updating.....");
    Console.WriteLine("Event arguments:");
    Console.WriteLine("Command Text: " + e.Command.CommandText);
    Console.WriteLine("Command Type: " + e.StatementType);
    Console.WriteLine("Status: " + e.Status);
  }

  // Event handler for RowUpdated event
  protected static void OnRowUpdated(object sender,
                                    OracleRowUpdatedEventArgs e)
  {
    Console.WriteLine("Row updated.....");
    Console.WriteLine("Event arguments:");
    Console.WriteLine("Command Text: " + e.Command.CommandText);
    Console.WriteLine("Command Type: " + e.StatementType);
    Console.WriteLine("Status: " + e.Status);
  }

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

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

    // Create 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;

    // 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");
  }
}
```

6.6.6.2 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".

6.7 OracleDatabase Class

An `OracleDatabase` object represents an Oracle Database instance.

Class Inheritance

```
System.Object
```

```
  Oracle.DataAccess.Client.OracleDatabase
```

Declaration

```
// C#
public sealed class OracleDatabase : IDisposable
```

Requirements

Provider	ODP.NET, Unmanaged Driver
Assembly	`Oracle.DataAccess.dll`
Namespace	`Oracle.DataAccess.Client`
.NET Framework	3.5, 4.5, 4.6

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 OracleDatbase 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 OracleConnetion object
        con.Dispose();
      }
    }
  }
}
```

6.7.1 OracleDatabase Members

`OracleDatabase` members are listed in the following tables.

OracleDatabase Constructors

The `OracleDatabase` constructor is listed in Table 6-50.

Table 6-50 OracleDatabase Constructors

Constructor	Description
OracleDatabase Constructor	Instantiates a new instance of `OracleDatabase` class using the supplied connection string

OracleDatabase Properties

The `OracleDatabase` properties are listed in Table 6-51.

Table 6-51 OracleDatabase Properties

Property	Description
ServerVersion	Specifies the database version number of the Oracle Database instance to which the connection is made

OracleDatabase Public Methods

The `OracleDatabase` public methods are listed in Table 6-52.

Table 6-52 OracleDatabase Public Methods

Public Method	Description
Dispose	Releases any resources or memory allocated by the object.
ExecuteNonQuery	Executes the supplied non-`SELECT` statement against the database
Shutdown	Shuts down the database (Overloaded)
Startup	Starts up the database (Overloaded)

6.7.2 OracleDatabase Constructor

The `OracleDatabase` constructor instantiates a new instance of the `OracleDatabase` class using the supplied connection string.

Declaration

```
// C#
public OracleDatabase(String connetionString);
```

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.

6.7.3 OracleDatabase Properties

The `OracleDatabase` properties are listed in Table 6-53.

Table 6-53 OracleDatabase Properties

Property	Description
ServerVersion	Specifies the database version number of the Oracle Database instance to which the connection is made

6.7.3.1 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.

6.7.4 OracleDatabase Public Methods

The `OracleDatabase` public methods are listed in Table 6-54.

Table 6-54 OracleDatabase Public Methods

Public Method	Description
Dispose	Releases any resources or memory allocated by the object.
ExecuteNonQuery	Executes the supplied non-`SELECT` statement against the database
Shutdown	Shuts down the database (Overloaded)
Startup	Starts up the database (Overloaded)

6.7.4.1 Dispose

This method releases any resources or memory allocated by the object.

Declaration

```
// C#
public void Dispose();
```

Implements

```
IDisposable
```

6.7.4.2 ExecuteNonQuery

This method executes the supplied non-`SELECT` statement against the database.

Declaration

```
// 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.

6.7.4.3 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.

6.7.4.4 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.

6.7.4.5 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

```
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);
      }
    }
  }
}
```

6.7.4.6 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.

6.7.4.7 Startup()

This method starts up the database.

Declaration

```
// 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.

6.7.4.8 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");`

6.8 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".
- "Obtaining Data from an OracleDataReader Object".

Class Inheritance

```
System.Object

  System.MarshalByRefObject

    System.Data.Common.DataReader

      System.Data.Common.DbDataReader

        Oracle.DataAccess.Client.OracleDataReader
```

Declaration

```
// C#
public sealed class OracleDataReader : DbDataReader, IEnumerable,
   IDataReader, IDisposable, IDataRecord
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver
Assembly	`Oracle.DataAccess.dll`	`Oracle.ManagedDataAccess.dll`
Namespace	`Oracle.DataAccess.Client`	`Oracle.ManagedDataAccess.Client`
.NET Framework	3.5, 4.5, 4.6	4.5, 4.6

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.Write("Byte codes: ");
        for (int i = 0; i < len; i++)
          Console.Write(byteCodes[i].ToString("x"));

        Console.WriteLine();
      }

      Console.WriteLine();
    }

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

6.8.1 OracleDataReader Members

`OracleDataReader` members are listed in the following tables.

OracleDataReader Static Methods

The `OracleDataReader` static method is listed in Table 6-55.

Table 6-55 OracleDataReader Static Method

Method	Description
`Equals`	Inherited from `System.Object` (Overloaded)

OracleDataReader Properties

`OracleDataReader` properties are listed in Table 6-56.

Table 6-56 OracleDataReader Properties

Property	Description
Depth	Gets a value indicating the depth of nesting for the current row
FetchSize	Specifies the size of `OracleDataReader`'s internal cache
FieldCount	Gets the number of columns in the result set
HasRows	Indicates whether the `OracleDataReader` has one or more rows
HiddenFieldCount	Gets the number of fields in the `OracleDataReader` that are hidden
IsClosed	Indicates whether or not the data reader is closed
Item	Gets the value of the column (Overloaded)

Table 6-56 (Cont.) OracleDataReader Properties

Property	Description
InitialLOBFetchSize	Specifies the amount that the `OracleDataReader` initially fetches for LOB columns
InitialLONGFetchSize	Specifies the amount that the `OracleDataReader` initially fetches for `LONG` and `LONG RAW` columns
RecordsAffected	Gets the number of rows changed, inserted, or deleted by execution of the SQL statement
RowSize	Gets the amount of memory the internal cache of the `OracleDataReader` needs to store one row of data.
UseEdmMapping	Indicates whether or not the `OracleDataReader` utilizes the Entity Data Model mapping configuration when returning values
VisibleFieldCount	Gets the number of fields in the `OracleDataReader` that are not hidden

OracleDataReader Public Methods

`OracleDataReader` public methods are listed in Table 6-57.

Table 6-57 OracleDataReader Public Methods

Public Method	Description
Close	Closes the `OracleDataReader`
`CreateObjRef`	Inherited from `System.MarshalByRefObject`
Dispose	Releases any resources or memory allocated by the object
`Equals`	Inherited from `System.Object` (Overloaded)
GetBoolean	*Not Supported*
GetByte	Returns the byte value of the specified column
GetBytes	Populates the provided byte array with up to the maximum number of bytes, from the specified offset (in bytes) of the column
GetChar	*Not Supported*
GetChars	Populates the provided character array with up to the maximum number of characters, from the specified offset (in characters) of the column
GetData	*Not Supported*
GetDataTypeName	Returns the ODP.NET type name of the specified column
GetDateTime	Returns the `DateTime` value of the specified column
GetDecimal	Returns the `decimal` value of the specified `NUMBER` column
GetDouble	Returns the `double` value of the specified `NUMBER` column or `BINARY_DOUBLE` column
GetEnumerator	Returns an `IEnumerator` that can be used to iterate through the collection

Table 6-57 (Cont.) OracleDataReader Public Methods

Public Method	Description
GetFieldType	Returns the `Type` of the specified column
GetFloat	Returns the `float` value of the specified `NUMBER` column or `BINARY_FLOAT` column
GetGuid	*Not Supported*
`GetHashCode`	Inherited from `System.Object`
GetInt16	Returns the `Int16` value of the specified `NUMBER` column
GetInt32	Returns the `Int32` value of the specified `NUMBER` column
GetInt64	Returns the `Int64` value of the specified `NUMBER` column
`GetLifetimeService`	Inherited by `System.MarshalByRefObject`
GetName	Returns the name of the specified column
GetOracleBFile	Returns an `OracleBFile` object of the specified `BFILE` column
GetOracleBinary	Returns an `OracleBinary` structure of the specified column
GetOracleBlob	Returns an `OracleBlob` object of the specified `BLOB` column
GetOracleBlobForUpdate	Returns an updatable `OracleBlob` object of the specified `BLOB` column
GetOracleClob	Returns an `OracleClob` object of the specified `CLOB` column
GetOracleClobForUpdate	Returns an updatable `OracleClob` object of the specified `CLOB` column
GetOracleDate	Returns an `OracleDate` structure of the specified `DATE` column
GetOracleDecimal	Returns an `OracleDecimal` structure of the specified `NUMBER` column
GetOracleIntervalDS	Returns an `OracleIntervalDS` structure of the specified `INTERVAL DAY TO SECOND` column
GetOracleIntervalYM	Returns an `OracleIntervalYM` structure of the specified `INTERVAL YEAR TO MONTH` column
GetOracleRef	Returns an `OracleRef` object of the specified `REF` column
GetOracleString	Returns an `OracleString` structure of the specified column
GetOracleTimeStamp	Returns an `OracleTimeStamp` structure of the Oracle `TimeStamp` column
GetOracleTimeStampLTZ	Returns an `OracleTimeStampLTZ` structure of the specified Oracle `TimeStamp WITH LOCAL TIME ZONE` column
GetOracleTimeStampTZ	Returns an `OracleTimeStampTZ` structure of the specified Oracle `TimeStamp WITH TIME ZONE` column
GetOracleXmlType	Returns an `OracleXmlType` object of the specified `XMLType` column
GetOracleValue	Returns the specified column value as a ODP.NET type
GetOracleValues	Gets all the column values as ODP.NET types

Table 6-57 (Cont.) OracleDataReader Public Methods

Public Method	Description
GetOrdinal	Returns the `0`-based ordinal (or index) of the specified column name
GetProviderSpecificFieldType	Returns the provider-specific type of the specified column
GetProviderSpecificValue	Returns an object that represents the underlying provider-specific value of the specified ordinal
GetProviderSpecificValues	Returns an array of objects that represent the underlying provider-specific values
GetSchemaTable	Returns a `DataTable` that describes the column metadata of the `OracleDataReader`
GetString	Returns the string value of the specified column
GetTimeSpan	Returns the `TimeSpan` value of the specified `INTERVAL DAY TO SECOND` column
`GetType`	Inherited from `System.Object` class
GetValue	Returns the column value as a .NET type
GetValues	Gets all the column values as .NET types
GetXmlReader	Returns the value of an `XMLType` column as an instance of an .NET `XmlTextReader`
IsDBNull	Indicates whether or not the column value is null
NextResult	Advances the data reader to the next result set when reading the results
Read	Reads the next row in the result set
`ToString`	Inherited from `System.Object`

6.8.2 OracleDataReader Static Methods

The `OracleDataReader` static method is listed in Table 6-58.

Table 6-58 OracleDataReader Static Method

Method	Description
`Equals`	Inherited from `System.Object` (Overloaded)

6.8.3 OracleDataReader Properties

`OracleDataReader` properties are listed in Table 6-59.

Table 6-59 OracleDataReader Properties

Property	Description
Depth	Gets a value indicating the depth of nesting for the current row
FetchSize	Specifies the size of `OracleDataReader`'s internal cache
FieldCount	Gets the number of columns in the result set
HasRows	Indicates whether the `OracleDataReader` has one or more rows
HiddenFieldCount	Gets the number of fields in the `OracleDataReader` that are hidden
IsClosed	Indicates whether or not the data reader is closed
Item	Gets the value of the column (Overloaded)
InitialLOBFetchSize	Specifies the amount that the `OracleDataReader` initially fetches for LOB columns
InitialLONGFetchSize	Specifies the amount that the `OracleDataReader` initially fetches for `LONG` and `LONG RAW` columns
RecordsAffected	Gets the number of rows changed, inserted, or deleted by execution of the SQL statement
RowSize	Gets the amount of memory the internal cache of the `OracleDataReader` needs to store one row of data
UseEdmMapping	Indicates whether or not the `OracleDataReader` utilizes the Entity Data Model mapping configuration when returning values
VisibleFieldCount	Gets the number of fields in the `OracleDataReader` that are not hidden

6.8.3.1 Depth

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

Declaration

```
// 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.

6.8.3.2 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.

6.8.3.3 FieldCount

This property returns the number of columns in the result set.

Declaration

```
// 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.

6.8.3.4 HasRows

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

Declaration

```
// 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();
  }
}
```

6.8.3.5 HiddenFieldCount

This property gets the number of fields in the `OracleDataReader` that are hidden.

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.

6.8.3.6 IsClosed

This property indicates whether or not the data reader is closed.

Declaration

```
// 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.

6.8.3.7 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.

6.8.3.8 Item [index]

This property gets the .NET `Value` of the column specified by the column index.

Declaration

```
// 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.

6.8.3.9 Item [string]

This property gets the .NET `Value` of the column specified by the column name.

Declaration

```
// C#
public override object this[string columnName] {get;}
```

Parameters

- *columnName*

  The name of the column.

Property Value

The .NET `Value` of the specified column.

Implements

`IDataRecord`

Remarks

Default = Not Applicable

A case-sensitive search is made to locate the specified column by its name. If this fails, then a case-insensitive search is made.

In C#, this property is the indexer for this class.

6.8.3.10 InitialLOBFetchSize

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

Declaration

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

Property Value

The size of the chunk to retrieve.

Exceptions

`InvalidOperationException` - The reader is closed.

Remarks

The maximum value supported for `InitialLOBFetchSize` is 2 GB.

Default is the `OracleCommand.InitialLOBFetchSize`, from which this value is inherited.

6.8.3.11 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`.

6.8.3.12 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.

6.8.3.13 RowSize

This property gets the amount of memory the internal cache of the `OracleDataReader` needs 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

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 runtime, 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 `RowSize * `*`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.

ODP.NET now supports values up to 32K for `VARCHAR2`, `NVARCHAR2` or `RAW` type columns in its calculation of `RowSize` value

6.8.3.14 UseEdmMapping

This read-only property indicates whether or not the `OracleDataReader` utilizes the Entity Data Model mapping configuration when returning values.

Declaration

```
// C#
public bool UseEdmMapping {get;}
```

Property Value

A `boolean` that indicates whether the `OracleDataReader` uses the Entity Data Model mapping configuration for returning values.

Remarks

Default is `false`.

The value is inherited from the `OracleCommand` object.

6.8.3.15 VisibleFieldCount

This property gets the number of fields in the `OracleDataReader` that are not hidden.

Declaration

```
// 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

```
// C#

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;";
    DbProviderFactory factory =
            DbProviderFactories.GetFactory("Oracle.DataAccess.Client");

    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.Write("Visible field count: " + visFC);
        Console.Write("Hidden field count: " + hidFC);

        reader.Dispose();
        cmd.Dispose();
      }
      catch (Exception ex)
      {
        Console.WriteLine(ex.Message);
        Console.WriteLine(ex.StackTrace);
      }
    }
  }
}
```

6.8.4 OracleDataReader Public Methods

`OracleDataReader` public methods are listed in Table 6-60.

Table 6-60 OracleDataReader Public Methods

Public Method	Description
Close	Closes the `OracleDataReader`
`CreateObjRef`	Inherited from `System.MarshalByRefObject`
Dispose	Releases any resources or memory allocated by the object
`Equals`	Inherited from `System.Object` (Overloaded)
GetBoolean	*Not Supported*
GetByte	Returns the byte value of the specified column
GetBytes	Populates the provided byte array with up to the maximum number of bytes, from the specified offset (in bytes) of the column
GetChar	*Not Supported*
GetChars	Populates the provided character array with up to the maximum number of characters, from the specified offset (in characters) of the column
GetData	*Not Supported*
GetDataTypeName	Returns the ODP.NET type name of the specified column
GetDateTime	Returns the `DateTime` value of the specified column
GetDecimal	Returns the `decimal` value of the specified `NUMBER` column
GetDouble	Returns the `double` value of the specified `NUMBER` column or `BINARY_DOUBLE` column
GetEnumerator	Returns an `IEnumerator` that can be used to iterate through the collection
GetFieldType	Returns the `Type` of the specified column
GetFloat	Returns the `float` value of the specified `NUMBER` column or `BINARY_FLOAT` column
GetGuid	*Not Supported*
`GetHashCode`	Inherited from `System.Object`
GetInt16	Returns the `Int16` value of the specified `NUMBER` column
GetInt32	Returns the `Int32` value of the specified `NUMBER` column
GetInt64	Returns the `Int64` value of the specified `NUMBER` column
`GetLifetimeService`	Inherited by `System.MarshalByRefObject`
GetName	Returns the name of the specified column
GetOracleBFile	Returns an `OracleBFile` object of the specified `BFILE` column
GetOracleBinary	Returns an `OracleBinary` structure of the specified column

Table 6-60 (Cont.) OracleDataReader Public Methods

Public Method	Description
GetOracleBlob	Returns an `OracleBlob` object of the specified `BLOB` column
GetOracleBlobForUpdate	Returns an updatable `OracleBlob` object of the specified `BLOB` column *Not Available in ODP.NET, Managed Driver*
GetOracleClob	Returns an `OracleClob` object of the specified `CLOB` column
GetOracleClobForUpdate	Returns an updatable `OracleClob` object of the specified `CLOB` column *Not Available in ODP.NET, Managed Driver*
GetOracleDate	Returns an `OracleDate` structure of the specified `DATE` column
GetOracleDecimal	Returns an `OracleDecimal` structure of the specified `NUMBER` column
GetOracleIntervalDS	Returns an `OracleIntervalDS` structure of the specified `INTERVAL DAY TO SECOND` column
GetOracleIntervalYM	Returns an `OracleIntervalYM` structure of the specified `INTERVAL YEAR TO MONTH` column
GetOracleRef	Returns an `OracleRef` object of the specified `REF` column *Not Available in ODP.NET, Managed Driver*
GetOracleString	Returns an `OracleString` structure of the specified column
GetOracleTimeStamp	Returns an `OracleTimeStamp` structure of the Oracle `TimeStamp` column
GetOracleTimeStampLTZ	Returns an `OracleTimeStampLTZ` structure of the specified Oracle `TimeStamp WITH LOCAL TIME ZONE` column
GetOracleTimeStampTZ	Returns an `OracleTimeStampTZ` structure of the specified Oracle `TimeStamp WITH TIME ZONE` column
GetOracleXmlType	Returns an `OracleXmlType` object of the specified `XMLType` column *Not Available in ODP.NET, Managed Driver*
GetOracleValue	Returns the specified column value as a ODP.NET type
GetOracleValues	Gets all the column values as ODP.NET types
GetOrdinal	Returns the `0`-based ordinal (or index) of the specified column name
GetProviderSpecificFieldType	Returns the provider-specific type of the specified column
GetProviderSpecificValue	Returns an object that represents the underlying provider-specific value of the specified ordinal
GetProviderSpecificValues	Returns an array of objects that represent the underlying provider-specific values
GetSchemaTable	Returns a `DataTable` that describes the column metadata of the `OracleDataReader`
GetString	Returns the string value of the specified column
GetTimeSpan	Returns the `TimeSpan` value of the specified `INTERVAL DAY TO SECOND` column

Table 6-60 (Cont.) OracleDataReader Public Methods

Public Method	Description
`GetType`	Inherited from `System.Object` class
GetValue	Returns the column value as a .NET type
GetValues	Gets all the column values as .NET types
GetXmlReader	Returns the value of an `XMLType` column as an instance of an .NET `XmlTextReader`
IsDBNull	Indicates whether or not the column value is null
NextResult	Advances the data reader to the next result set when reading the results
Read	Reads the next row in the result set
`ToString`	Inherited from `System.Object`

6.8.4.1 Close

This method closes the `OracleDataReader`.

Declaration

```
// 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".

6.8.4.2 Dispose

This method releases any resources or memory allocated by the object.

Declaration

```
// C#
public void Dispose();
```

Implements

`IDisposable`

Remarks

The `Dispose` method also closes the `OracleDataReader`.

6.8.4.3 GetBoolean

This method is not supported.

Declaration

```
// C#
public override bool GetBoolean(int index);
```

Parameters

- *`index`*

  The zero-based column index.

Implements

`IDataRecord`

Exceptions

`NotSupportedException` - This property is not supported.

6.8.4.4 GetByte

This method returns the byte value of the specified column.

Declaration

```
// 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

`IsDBNull` should be called to check for `NULL` values before calling this method.

6.8.4.5 GetBytes

This method populates the provided byte array with up to the maximum number of bytes, from the specified offset (in bytes) of the column.

Declaration

```
// C#
public 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.

6.8.4.6 GetChar

This method is not supported.

Declaration

```
// C#
public override long GetChar(int index);
```

Parameters

- *`index`*

  The zero based column index.

Implements

`IDataRecord`

Exceptions

`NotSupportedException` - This property is not supported.

6.8.4.7 GetChars

This method populates the provided character array with up to the maximum number of characters, from the specified offset (in characters) of the column.

Declaration

```
// C#
public 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.

6.8.4.8 GetData

This method is not supported

6.8.4.9 GetDataTypeName

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

Declaration

```
// 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.

6.8.4.10 GetDateTime

This method returns the `DateTime` value of the specified column.

Declaration

```
// 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.

6.8.4.11 GetDecimal

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

Declaration

```
// C#
public override decimal GetDecimal(int index);
```

Parameters

- *`index`*

  The zero-based column index.

Return Value

The `decimal` value of the column.

Implements

`IDataRecord`

Exceptions

`InvalidOperationException` - The connection is closed, the reader is closed, `Read()` has not been called, or all rows have been read.

`IndexOutOfRangeException` - The column index is invalid.

`InvalidCastException` - The accessor method is invalid for this column type or the column value is `NULL`.

Remarks

`IsDBNull` should be called to check for `NULL` values before calling this method.

6.8.4.12 GetDouble

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

Declaration

```
// C#
public override double GetDouble(int index);
```

Parameters

- *`index`*

  The zero-based column index.

Return Value

The `double` value of the column.

Implements

`IDataRecord`

Exceptions

`InvalidOperationException` - The connection is closed, the reader is closed, `Read()` has not been called, or all rows have been read.

`IndexOutOfRangeException` - The column index is invalid.

`InvalidCastException` - The accessor method is invalid for this column type or the column value is `NULL`.

Remarks

`IsDBNull` should be called to check for `NULL` values before calling this method.

`GetDouble` now supports retrieval of data from `BINARY_DOUBLE` columns.

6.8.4.13 GetEnumerator

This method returns an `IEnumerator` that can be used to iterate through the collection (record set).

Declaration

```
// 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.

6.8.4.14 GetFieldType

This method returns the `type` of the specified column.

Declaration

```
// 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`.

6.8.4.15 GetFloat

This method returns the `float` value of the specified `NUMBER` column or `BINARY_FLOAT` column.

Declaration

```
// 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.

`GetFloat` now supports retrieval of data from `BINARY_FLOAT` columns.

6.8.4.16 GetGuid

This method is not supported.

Declaration

```
// C#
public override Guid GetGuid(int index);
```

Parameters

- *index*

  The zero-based column index.

Implements

`IDataRecord`

Exceptions

`NotSupportedException` - This property is not supported.

6.8.4.17 GetInt16

This method returns the `Int16` value of the specified `NUMBER` column.

> **Note:**
>
> `short` is equivalent to `Int16`.

Declaration

```
// 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.

6.8.4.18 GetInt32

This method returns the `Int32` value of the specified `NUMBER` column.

> **Note:**
>
> `int` is equivalent to `Int32`.

Declaration

```
// 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.

6.8.4.19 GetInt64

This method returns the `Int64` value of the specified `NUMBER` column.

Note:

`long` is equivalent to `Int64`.

Declaration

```
// 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.

6.8.4.20 GetName

This method returns the name of the specified column.

Declaration

```
// 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.

6.8.4.21 GetOracleBFile

This method returns an `OracleBFile` object of the specified `BFILE` column.

Declaration

```
// C#
public OracleBFile GetOracleBFile(int index);
```

Parameters

- *`index`*

  The zero-based column index.

Return Value

The `OracleBFile` value of the column.

Exceptions

`InvalidOperationException` - The connection is closed, the reader is closed, `Read()` has not been called, or all rows have been read.

`IndexOutOfRangeException` - The column index is invalid.

`InvalidCastException` - The accessor method is invalid for this column type or the column value is `NULL`.

Remarks

`IsDBNull` should be called to check for `NULL` values before calling this method.

6.8.4.22 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`

6.8.4.23 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.

6.8.4.24 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.

6.8.4.25 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();
      }
    }
```

6.8.4.26 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 `GetOracleBlob()` can have a different value than the `OracleBlob` obtained from `GetOracleBlobForUpdate()` since it is not obtained from the original snapshot.

`IsDBNull` should be called to check for `NULL` values before calling this method.

The returned `OracleBlob` object can be used to safely update the `BLOB` because the `BLOB` column has been locked after a call to this method.

Invoking this method internally executes a `SELECT..FOR UPDATE` statement which locks the row.

Different `WAIT` clauses are appended to the statement, depending on the *`wait`* value. If the *`wait`* value is:

- `0`

  "`NOWAIT`" is appended at the end of a `SELECT..FOR UPDATE` statement. The statement executes immediately whether the lock is acquired or not. If the lock is not acquired, an exception is thrown.

- *`n`*

  "`WAIT` *`n`*" is appended at the end of a `SELECT..FOR UPDATE` statement. The statement executes as soon as the lock is acquired. However, if the lock cannot be acquired by *`n`* seconds, this method call throws an exception.

  The `WAIT` *`n`*" feature is only available for Oracle9*i* or later. For any version lower than Oracle9*i*, *`n`* is implicitly treated as `-1` and nothing is appended at the end of a `SELECT..FOR UPDATE` statement.

- `-1`

  Nothing is appended at the end of the `SELECT..FOR UPDATE`. The statement execution waits indefinitely until a lock can be acquired.

Example

The `GetOracleBlobForUpdate` methods are comparable. See "Example" for a code example demonstrating usage.

6.8.4.27 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.

6.8.4.28 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.

6.8.4.29 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.

```
/* 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();
    }
  }
```

6.8.4.30 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 Oracle9*i* or later. For any version lower than Oracle9*i*, *`n`* is implicitly treated as `-1` and nothing is appended at the end of a `SELECT..FOR UPDATE` statement.

- `-1`

  Nothing is appended at the end of the `SELECT..FOR UPDATE`. The statement execution waits indefinitely until a lock can be acquired.

`IsDBNull` should be called to check for `NULL` values before calling this method.

Example

The GetOracleClobForUpdate methods are comparable. See "Example" for a code example demonstrating usage.

6.8.4.31 GetOracleDate

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

Declaration

```
// C#
public OracleDate GetOracleDate(int index);
```

Parameters

- *index*

  The zero-based column index.

Return Value

The `OracleDate` value of the column.

Exceptions

`InvalidOperationException` - The connection is closed, the reader is closed, `Read()` has not been called, or all rows have been read.

`IndexOutOfRangeException` - The column index is invalid.

`InvalidCastException` - The accessor method is invalid for this column type or the column value is `NULL`.

Remarks

`IsDBNull` should be called to check for `NULL` values before calling this method.

6.8.4.32 GetOracleDecimal

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

Declaration

```
// C#
public OracleDecimal GetOracleDecimal(int index);
```

Parameters

- *index*

  The zero-based column index.

Return Value

The `OracleDecimal` value of the column.

Exceptions

`InvalidOperationException` - The connection is closed, the reader is closed, `Read()` has not been called, or all rows have been read.

`IndexOutOfRangeException` - The column index is invalid.

`InvalidCastException` - The accessor method is invalid for this column type or the column value is `NULL`.

Remarks

`IsDBNull` should be called to check for `NULL` values before calling this method.

6.8.4.33 GetOracleIntervalDS

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

Declaration

```
// C#
public OracleIntervalDS GetOracleIntervalDS(int index);
```

Parameters

- *`index`*

  The zero-based column index.

Return Value

The `OracleIntervalDS` value of the column.

Exceptions

`InvalidOperationException` - The connection is closed, the reader is closed, `Read()` has not been called, or all rows have been read.

`IndexOutOfRangeException` - The column index is invalid.

`InvalidCastException` - The accessor method is invalid for this column type or the column value is `NULL`.

Remarks

`IsDBNull` should be called to check for `NULL` values before calling this method.

6.8.4.34 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.

6.8.4.35 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.

`InvalidCastException` - The accessor method is invalid for this column type.

6.8.4.36 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.

6.8.4.37 GetOracleTimeStamp

This method returns an `OracleTimeStamp` structure of the Oracle `TimeStamp` column.

Declaration

```
// C#
public OracleTimeStamp GetOracleTimeStamp(int index);
```

Parameters

- *`index`*

  The zero-based column index.

Return Value

The `OracleTimeStamp` value of the column.

Exceptions

`InvalidOperationException` - The connection is closed, the reader is closed, `Read()` has not been called, or all rows have been read.

`IndexOutOfRangeException` - The column index is invalid.

`InvalidCastException` - The accessor method is invalid for this column type or the column value is `NULL`.

Remarks

`GetOracleTimeStamp` is used with the Oracle Type `TimeStamp`.

`IsDBNull` should be called to check for `NULL` values before calling this method.

6.8.4.38 GetOracleTimeStampLTZ

This method returns an `OracleTimeStampLTZ` structure of the specified Oracle `TimeStamp WITH LOCAL TIME ZONE` column.

Declaration

```
// C#
public OracleTimeStampLTZ GetOracleTimeStampLTZ(int index);
```

Parameters

- *`index`*

  The zero-based column index.

Return Value

The `OracleTimeStampLTZ` value of the column.

Exceptions

`InvalidOperationException` - The connection is closed, the reader is closed, `Read()` has not been called, or all rows have been read.

`IndexOutOfRangeException` - The column index is invalid.

`InvalidCastException` - The accessor method is invalid for this column type or the column value is `NULL`.

Remarks

`GetOracleTimeStampLTZ` is used with the Oracle Type `TimeStamp` with Local Time Zone columns.

`IsDBNull` should be called to check for `NULL` values before calling this method.

6.8.4.39 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.

6.8.4.40 GetOracleXmlType

This method returns an `OracleXmlType` object of the specified `XMLType` column.

Declaration

```
// C#
public OracleXmlType GetOracleXmlType(int index);
```

Parameters

- *`index`*

  The zero-based column index.

Return Value

The `OracleXmlType` value of the column.

Exceptions

`InvalidCastException` - The accessor method is invalid for this column type or the column value is `NULL`.

Remarks

`IsDBNull` should be called to check for `NULL` values before calling this method.

6.8.4.41 GetOracleValue

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

Declaration

```
// C#
public object GetOracleValue(int index);
```

Parameters

- *`index`*

  The zero-based column index.

Return Value

The value of the column as an ODP.NET type.

Exceptions

`InvalidOperationException` - The connection is closed, the reader is closed, `Read()` has not been called, or all rows have been read.

`IndexOutOfRangeException` - The column index is invalid.

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.

6.8.4.42 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.

6.8.4.43 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.

6.8.4.44 GetProviderSpecificFieldType

This method returns the provider-specific type of the specified column.

Declaration

```
// 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

`GetProviderSpecficFieldType` 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`.

6.8.4.45 GetProviderSpecificValue

This method returns an object that represents the underlying provider-specific value of the specified ordinal.

Declaration

```
// 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.

6.8.4.46 GetProviderSpecificValues

This method returns an array of objects that represent the underlying provider-specific values.

Declaration

```
// 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.

6.8.4.47 GetSchemaTable

This method returns a `DataTable` that describes the column metadata of the `OracleDataReader`.

Declaration

```
// 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 value of `ColumnSize` can show value up to 32K depending on the definition of `VARCHAR2`, `NVARCHAR2`, or `RAW` type columns in the table definition.

The columns of the `SchemaTable` are in the order shown.

Table 6-61 OracleDataReader SchemaTable

Name	Name Type	Description
`ColumnName`	`System.String`	The name of the column.
`ColumnOrdinal`	`System.Int32`	The `0`-based ordinal of the column.

Table 6-61 (Cont.) OracleDataReader SchemaTable

Name	Name Type	Description
`ColumnSize`	`System.Int64`	The maximum possible length of a value in the column. `ColumnSize` value is determined as follows: • `CHAR` and `VARCHAR2` types: in bytes - if `IsByteSemantic` boolean value is `true` in characters - if `IsByteSemantic` boolean value is `false` • All other types: in bytes See "IsByteSemantic" for more information.
`NumericPrecision`	`System.Int16`	The maximum precision of the column, if the column is a numeric data type. This column has valid values for Oracle `NUMBER`, Oracle `INTERVAL YEAR TO MONTH`, and Oracle `INTERVAL DAY TO SECOND` columns. For all other columns, the value is `null`.
`NumericScale`	`System.Int16`	The scale of the column. This column has valid values for Oracle `NUMBER`, Oracle `INTERVAL DAY TO SECOND`, and the Oracle `TIMESTAMP` columns. For all other columns, the value is `null`.
`IsUnique`	`System.Boolean`	Indicates whether or not the column is unique. `true` if no two rows in the base table can have the same value in this column, where the base table is the table returned in `BaseTableName`. `IsUnique` is guaranteed to be `true` if one of the following applies in descending order of priority: • the column constitutes a base table primary key by itself and a `NOT NULL` constraint has been defined on the column • there is a unique constraint or a unique index that applies only to this column and a `NOT NULL` constraint has been defined on the column • the column is an explicitly selected `ROWID` `IsUnique` is false if the column can contain duplicate values in the base table. The default is `false`. The value of this property is the same for each occurrence of the base table column in the select list.

Table 6-61 (Cont.) OracleDataReader SchemaTable

Name	Name Type	Description
`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 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 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. • A base table composite primary key with the following condition: A `NULL` constraint must be defined on at least one, but not all, of the columns. • 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 Oracle9*i*.
`IsExpression`	`System.Boolean`	`true` if the column is an expression; otherwise `false`.
`IsHidden`	`System.Boolean`	`true` if the column is hidden; otherwise `false`.

Table 6-61 (Cont.) OracleDataReader SchemaTable

Name	Name Type	Description
`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.
`IsIdentity`	`System.Boolean`	`true` if the column is an identity column; otherwise `false`.
`IsAutoIncrement`	`System.Boolean`	`true` if the column assigns values to new rows in fixed increments; otherwise `false`. *Not Available in ODP.NET, Managed Driver*
`IdentityType`	`OracleIdentityType`	An `OracleIdentityType` enumeration value that specifies how the identity column values are generated; otherwise `DbNull.Value`, if the column is not an identity column. *Not Available in ODP.NET, Managed Driver*

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();
  }
}
```

6.8.4.48 GetString

This method returns the `string` value of the specified column.

Declaration

```
// 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.

6.8.4.49 GetTimeSpan

This method returns the `TimeSpan` value of the specified `INTERVAL DAY TO SECOND` column.

Declaration

```
// C#
public TimeSpan GetTimeSpan(int index);
```

Parameters

- *`index`*

  The zero-based column index.

Return Value

The `TimeSpan` value of the column.

Implements

`IDataRecord`

Exceptions

`InvalidOperationException` - The connection is closed, the reader is closed, `Read()` has not been called, or all rows have been read.

`IndexOutOfRangeException` - The column index is invalid.

`InvalidCastException` - The accessor method is invalid for this column type or the column value is `NULL`.

Remarks

`IsDBNull` should be called to check for `NULL` values before calling this method.

6.8.4.50 GetValue

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

Declaration

```
// 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 hexidecimal value is returned as a .NET string that represents the `REF` in the database.

If the UDT is `NULL`, `DBNull.Value` is returned

6.8.4.51 GetValues

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

Declaration

```
// 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.

6.8.4.52 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.

6.8.4.53 IsDBNull

This method indicates whether or not the column value is `NULL`.

Declaration

```
// 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".

6.8.4.54 NextResult

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

Declaration

```
// 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.

6.8.4.55 Read

This method reads the next row in the result set.

Declaration

```
// 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".

6.9 OracleDataSourceEnumerator Class

An `OracleDataSourceEnumerator` object allows applications to generically obtain a collection of data sources to connect to.

Class Inheritance

```
System.Object

  System.DbDataSourceEnumerator

    Oracle.DataAccess.Client.OracleDataSourceEnumerator
```

Declaration

```
// C#
public sealed class OracleDataSourceEnumerator : DbDataSourceEnumerator
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver
Assembly	`Oracle.DataAccess.dll`	`Oracle.ManagedDataAccess.dll`
Namespace	`Oracle.DataAccess.Client`	`Oracle.ManagedDataAccess.Client`
.NET Framework	3.5, 4.5, 4.6	4.5, 4.6

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)
    {
      DbDataSourceEnumerator 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]);
      Console.WriteLine(dt.Columns[2] + " : " + dt.Rows[0][2]);
      Console.WriteLine(dt.Columns[3] + " : " + dt.Rows[0][3]);
      Console.WriteLine(dt.Columns[4] + " : " + dt.Rows[0][4]);
    }
    else
      Console.Write("Data source enumeration is not supported by provider");
  }
}
```

6.9.1 OracleDataSourceEnumerator Members

`OracleDataSourceEnumerator` members are listed in the following tables.

OracleDataSourceEnumerator Constructor

`OracleDataSourceEnumerator` Public Methods are listed in Table 6-62.

Table 6-62 OracleDataSourceEnumerator Method

Method	Description
OracleDataSourceEnumerator Constructor	Instantiates a new instance of the OracleDataSourceEnumerator class

See Also:

- "Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"
- OracleDataSourceEnumerator Class

OracleDataSourceEnumerator Public Methods

`OracleDataSourceEnumerator` Public Methods are listed in Table 6-63.

Table 6-63 OracleDataSourceEnumerator Method

Method	Description
GetDataSources	Returns a `DataTable` object with information on all the TNS alias entries in the `tnsnames.ora` file and entries retrieved from the LDAP servers configured in `ldap.ora` if LDAP Naming is enabled

6.9.2 OracleDataSourceEnumerator Constructor

`OracleDataSourceEnumerator` constructor creates new instances of an `OracleDataSourceEnumerator` class.

Declaration

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

6.9.3 OracleDataSourceEnumerator Public Methods

The `OracleDataSourceEnumerator` static method is listed in Table 6-64.

Table 6-64 OracleDataSourceEnumerator Method

Method	Description
GetDataSources	Returns a `DataTable` object with information on all the TNS alias entries in the `tnsnames.ora` file

6.9.3.1 GetDataSources

This method returns a `DataTable` object with information on all the TNS alias entries in the `tnsnames.ora` file and entries retrieved from the LDAP servers configured in `ldap.ora` if LDAP naming is enabled.

Declaration

```
// 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 and each entry retrieved from the LDAP servers. If a `tnsnames.ora` file is not found and LDAP Naming is not configured, then the returned `DataTable` object will be empty.

This method in ODP.NET, Managed Driver can fetch all the data source aliases from an LDAP server, such as Oracle Internet Directory or Microsoft Active Directory. This method in ODP.NET, Unmanaged Driver does not support retrieving data source aliases from an LDAP server.

When Oracle Internet Directory (OID) is used for the TNS naming repository, there is a limit of 1000 TNS entries retrieved.

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`)

If the TNS and/or LDAP information changes for existing pooled connections, then calling `GetDataSources` will not return these changes unless the pools have been cleared.

6.10 OracleError Class

The `OracleError` class represents an error reported by Oracle.

Class Inheritance

```
System.Object

  Oracle.DataAccess.Client.OracleError
```

Declaration

```
// C#
public sealed class OracleError
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver
Assembly	`Oracle.DataAccess.dll`	`Oracle.ManagedDataAccess.dll`
Namespace	`Oracle.DataAccess.Client`	`Oracle.ManagedDataAccess.Client`
.NET Framework	3.5, 4.5, 4.6	4.5, 4.6

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);
      }
    }
  }
}
```

6.10.1 OracleError Members

`OracleError` members are listed in the following tables.

OracleError Static Methods

The `OracleError` static method is listed in Table 6-65.

Table 6-65 OracleError Static Method

Method	Description
`Equals`	Inherited from `System.Object` (Overloaded)

OracleError Properties

`OracleError` properties are listed in Table 6-66.

Table 6-66 OracleError Properties

Property	Description
ArrayBindIndex	Specifies the row number of errors that occurred during the Array Bind execution
DataSource	Specifies the Oracle service name (TNS name) that identifies the Oracle database
Message	Specifies the `message` describing the `error`
Number	Specifies the Oracle `error` number
Procedure	Specifies the stored procedure that causes the `error`
Source	Specifies the name of the data provider that generates the `error`

OracleError Methods

`OracleError` methods are listed in Table 6-67.

Table 6-67 OracleError Methods

Method	Description
`Equals`	Inherited from `System.Object` (Overloaded)
`GetHashCode`	Inherited from `System.Object`
`GetType`	Inherited from `System.Object`
ToString	Returns a string representation of the `OracleError`

6.10.2 OracleError Static Methods

The `OracleError` static method is listed in Table 6-68.

Table 6-68 OracleError Static Method

Method	Description
`Equals`	Inherited from `System.Object` (Overloaded)

6.10.3 OracleError Properties

`OracleError` properties are listed in Table 6-69.

Table 6-69 OracleError Properties

Property	Description
ArrayBindIndex	Specifies the row number of errors that occurred during the Array Bind execution

Table 6-69 (Cont.) OracleError Properties

Property	Description
DataSource	Specifies the Oracle service name (TNS name) that identifies the Oracle database
Message	Specifies the `message` describing the `error`
Number	Specifies the Oracle `error` number
Procedure	Specifies the stored procedure that causes the `error`
Source	Specifies the name of the data provider that generates the `error`

6.10.3.1 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.

6.10.3.2 DataSource

This property specifies the Oracle service name (TNS name) that identifies the Oracle database.

Declaration

```
// C#
public string DataSource {get;}
```

Property Value

A `string`.

6.10.3.3 Message

This property specifies the `message` describing the `error`.

Declaration

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

Property Value

A `string`.

6.10.3.4 Number

This property specifies the Oracle `error` number.

Declaration

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

Property Value

An `int`.

6.10.3.5 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.

6.10.3.6 Source

This property specifies the name of the data provider that generates the `error`.

Declaration

```
// C#
public string Source {get;}
```

Property Value

A `string`.

6.10.4 OracleError Methods

`OracleError` methods are listed in Table 6-70.

Table 6-70 OracleError Methods

Method	Description
`Equals`	Inherited from `System.Object` (Overloaded)
`GetHashCode`	Inherited from `System.Object`
`GetType`	Inherited from System.`Object`
ToString	Returns a string representation of the `OracleError`

6.10.4.1 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

6.11 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

    Oracle.DataAccess.Client.OracleErrorCollection
```

Declaration

```
// C#
public sealed class OracleErrorCollection : ArrayList
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver
Assembly	`Oracle.DataAccess.dll`	`Oracle.ManagedDataAccess.dll`
Namespace	`Oracle.DataAccess.Client`	`Oracle.ManagedDataAccess.Client`
.NET Framework	3.5, 4.5, 4.6	4.5, 4.6

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 `OracleError`s.

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);
      }
    }
  }
}
```

6.11.1 OracleErrorCollection Members

`OracleErrorCollection` members are listed in the following tables.

OracleErrorCollection Static Methods

`OracleErrorCollection` static methods are listed in Table 6-71.

Table 6-71 OracleErrorCollection Static Methods

Method	Description
`Equals`	Inherited from `System.Object` (Overloaded)

OracleErrorCollection Properties

`OracleErrorCollection` properties are listed in Table 6-72.

Table 6-72 OracleErrorCollection Properties

Property	Description
`Capacity`	Inherited from `System.Collections.ArrayList`
`Count`	Inherited from `System.Collections.ArrayList`
`IsReadOnly`	Inherited from `System.Collections.ArrayList`
`IsSynchronized`	Inherited from `System.Collections.ArrayList`
`Item`	Inherited from `System.Collections.ArrayList`

OracleErrorCollection Public Methods

`OracleErrorCollection` public methods are listed in Table 6-73.

Table 6-73 OracleErrorCollection Public Methods

Public Method	Description
`CopyTo`	Inherited from `System.Collections.ArrayList`
`Equals`	Inherited from `System.Object` (Overloaded)
`GetHashCode`	Inherited from `System.Object`
`GetType`	Inherited from `System.Object`
`ToString`	Inherited from `System.Object`

6.11.2 OracleErrorCollection Static Methods

The `OracleErrorCollection` static method is listed in Table 6-74.

Table 6-74 OracleErrorCollection Static Method

Method	Description
`Equals`	Inherited from `System.Object` (Overloaded)

6.11.3 OracleErrorCollection Properties

`OracleErrorCollection` properties are listed in Table 6-75.

Table 6-75 OracleErrorCollection Properties

Property	Description
`Capacity`	Inherited from `System.Collections.ArrayList`
`Count`	Inherited from `System.Collections.ArrayList`
`IsReadOnly`	Inherited from `System.Collections.ArrayList`
`IsSynchronized`	Inherited from `System.Collections.ArrayList`
`Item`	Inherited from `System.Collections.ArrayList`

6.11.4 OracleErrorCollection Public Methods

`OracleErrorCollection` public methods are listed in Table 6-76.

Table 6-76 OracleErrorCollection Public Methods

Public Method	Description
`CopyTo`	Inherited from `System.Collections.ArrayList`
`Equals`	Inherited from `System.Object` (Overloaded)
`GetHashCode`	Inherited from System.`Object`
`GetType`	Inherited from `System.Object`
`ToString`	Inherited from `System.Object`

6.12 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

        System.Data.Common.DbException

       Oracle.DataAccess.Client.OracleException
```

Declaration

```
// C#
public sealed class OracleException : SystemException
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver
Assembly	`Oracle.DataAccess.dll`	`Oracle.ManagedDataAccess.dll`
Namespace	`Oracle.DataAccess.Client`	`Oracle.ManagedDataAccess.Client`
.NET Framework	3.5, 4.5, 4.6	4.5, 4.6

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);
    }
    }
  }
```

See Also:

- "Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"
- OracleException Members
- OracleException Methods
- OracleException Static Methods
- OracleException Static Methods
- OracleException Properties
- OracleException Methods

6.12.1 OracleException Members

`OracleException` members are listed in the following tables.

OracleException Static Methods

The `OracleException` static method is listed in Table 6-77.

Table 6-77 OracleException Static Method

Method	Description
`Equals`	Inherited from `System.Object` (Overloaded)

OracleException Properties

`OracleException` properties are listed in Table 6-78.

Table 6-78 OracleException Properties

Property	Description
DataSource	Specifies the TNS name that contains the information for connecting to an Oracle instance
Errors	Specifies a collection of one or more `OracleError` objects that contain information about exceptions generated by the Oracle database
`HelpLink`	Inherited from `System.Exception`
`InnerException`	Inherited from `System.Exception`
IsRecoverable	Specifies whether the current operation producing this exception can succeed if retried
Message	Specifies the error messages that occur in the exception

Table 6-78 (Cont.) OracleException Properties

Property	Description
Number	Specifies the Oracle error number
OracleLogicalTransaction	Returns an `OracleLogicalTransaction` object for a recoverable error when using Transaction Guard
Procedure	Specifies the stored procedure that cause the exception
Source	Specifies the name of the data provider that generates the error
`StackTrace`	Inherited from `System.Exception`
`TargetSite`	Inherited from `System.Exception`

OracleException Methods

`OracleException` methods are listed in Table 6-79.

Table 6-79 OracleException Methods

Method	Description
`Equals`	Inherited from `System.Object` (Overloaded)
`GetBaseException`	Inherited from `System.Exception`
`GetHashCode`	Inherited from `System.Object`
GetObjectData	Sets the serializable `info` object with information about the exception
`GetType`	Inherited from `System.Object`
ToString	Returns the fully qualified name of this exception

6.12.2 OracleException Static Methods

The `OracleException` static method is listed in Table 6-80.

Table 6-80 OracleException Static Method

Method	Description
`Equals`	Inherited from `System.Object` (Overloaded)

6.12.3 OracleException Properties

`OracleException` properties are listed in Table 6-81.

Table 6-81 OracleException Properties

Property	Description
DataSource	Specifies the TNS name that contains the information for connecting to an Oracle instance
Errors	Specifies a collection of one or more `OracleError` objects that contain information about exceptions generated by the Oracle database
`HelpLink`	Inherited from `System.Exception`
`InnerException`	Inherited from `System.Exception`
IsRecoverable	Specifies whether the current operation producing this exception can succeed if retried
Message	Specifies the error messages that occur in the exception
Number	Specifies the Oracle error number
OracleLogicalTransaction	Returns an `OracleLogicalTransaction` object for a recoverable error when using Transaction Guard
Procedure	Specifies the stored procedure that cause the exception
Source	Specifies the name of the data provider that generates the error
`StackTrace`	Inherited from `System.Exception`
`TargetSite`	Inherited from `System.Exception`

6.12.3.1 DataSource

This property specifies the TNS name that contains the information for connecting to an Oracle instance.

Declaration

```
// C#
public string DataSource {get;}
```

Property Value

The TNS name containing the connect information.

6.12.3.2 Errors

This property specifies a collection of one or more `OracleError` objects that contain information about exceptions generated by the Oracle database.

Declaration

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

Property Value

An `OracleErrorCollection`.

Remarks

The `Errors` property contains at least one instance of `OracleError` objects.

6.12.3.3 IsRecoverable

This property specifies whether the current operation producing this exception can succeed if retried.

Declaration

```
// C#
public bool IsRecoverable {get;}
```

Property Value

A `bool`.

Remarks

When a database outage occurs, such as during a network failure, the session becomes unavailable and the client receives an error code. The client can have difficulty determining whether the in-flight operation committed or needs to be resubmitted. Oracle automatically determines whether an in-flight database operation can be recovered or not using the `IsRecoverable` property. If `IsRecoverable` returns true after an outage, then the application can retrieve the current operation status and complete the transaction. If `IsRecoverable` returns false, then the application can rollback the current operation and resubmit the transaction.

This property is often used in conjunction with Transaction Guard.

6.12.3.4 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.

6.12.3.5 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.

6.12.3.6 OracleLogicalTransaction

This property will returns an `OracleLogicalTransaction` object for a recoverable error when using Transaction Guard.

Declaration

```
// C#
public OracleLogicalTransaction OracleLogicalTransaction {get;}
```

Property Value

An `OracleLogicalTransaction`.

Remarks

`OracleLogicalTransaction` is non-null when both of the following conditions are met:

- Transaction Guard is enabled on the service
- The exception is a recoverable error

`OracleLogicalTransaction` can be used to determine the transaction outcome by looking at the two properties that it exposes: `Committed` and `UserCallCompleted`. If the outcome is not known, then `Committed` and `UserCallCompleted` will be set to null.

If the outcome of a recoverable error could not be determined by ODP.NET and the connection have not participated in a distributed transaction, then the `OracleLogicalTransactionId` property of the `OracleLogicalTransaction` object will be non-null and it can be used to determine the outcome by having the application explicitly call the `OracleLogicalTransaction.GetOutcome` static method, if the database/ service is up.

6.12.3.7 Procedure

This property specifies the stored procedure that caused the exception.

Declaration

```
// C#
public string Procedure {get;}
```

Property Value

The stored procedure name.

6.12.3.8 Source

Overrides `Exception`

This property specifies the name of the data provider that generates the error.

Declaration

```
// C#
public override string Source {get;}
```

Property Value

The name of the data provider.

6.12.4 OracleException Methods

`OracleException` methods are listed in Table 6-82.

Table 6-82   OracleException Methods

Method	Description
`Equals`	Inherited from `System.Object` (Overloaded)
`GetBaseException`	Inherited from `System.Exception`
`GetHashCode`	Inherited from System.`Object`
GetObjectData	Sets the serializable `info` object with information about the exception
`GetType`	Inherited from `System.Object`
ToString	Returns the fully qualified name of this exception

6.12.4.1 GetObjectData

Overrides `Exception`

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

Declaration

```
// C#
public override void GetObjectData(SerializationInfo info, StreamingContext
    context);
```

Parameters

- *`info`*

  A `SerializationInfo` object.

- *`context`*

  A `StreamingContext` object.

Remarks

The information includes `DataSource`, `Message`, `Number`, `Procedure`, `Source`, and `StackTrace`.

6.12.4.2 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());
    }
  }
}
```

6.13 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

    Oracle.DataAccess.Client.OracleInfoMessageEventArgs
```

Declaration

```
// C#
public sealed class OracleInfoMessageEventArgs
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver
Assembly	`Oracle.DataAccess.dll`	`Oracle.ManagedDataAccess.dll`
Namespace	`Oracle.DataAccess.Client`	`Oracle.ManagedDataAccess.Client`
.NET Framework	3.5, 4.5, 4.6	4.5, 4.6

Thread Safety

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

Example

```
// C#

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


class InfoMessageSample
{
  public static void WarningHandler(object src,
    OracleInfoMessageEventArgs args)
  {
      Console.WriteLine("Source object is: " + src.GetType().Name);
      Console.WriteLine("InfoMessageArgs.Message is " + args.Message);
      Console.WriteLine("InfoMessageArgs.Source is " + args.Source);
  }
  static void Main()
  {
    OracleConnection con = new OracleConnection("User Id=scott;" +
      "Password=tiger;Data Source=oracle;");
    con.Open();

    OracleCommand cmd = con.CreateCommand();

    //Register to the InfoMessageHandler
    cmd.Connection.InfoMessage +=
      new OracleInfoMessageEventHandler(WarningHandler);

    cmd.CommandText =
      "create or replace procedure SelectWithNoInto( " +
      "  empname in VARCHAR2) AS " +
      "BEGIN " +
      "  select * from emp where ename = empname; " +
      "END SelectWithNoInto;";

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

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

6.13.1 OracleInfoMessageEventArgs Members

`OracleInfoMessageEventArgs` members are listed in the following tables.

OracleInfoMessageEventArgs Static Methods

The `OracleInfoMessageEventArgs` static methods is listed in Table 6-83.

Table 6-83 OracleInfoMessageEventArgs Static Method

Method	Description
`Equals`	Inherited from `System.Object` (Overloaded)

OracleInfoMessageEventArgs Properties

The `OracleInfoMessageEventArgs` properties are listed in Table 6-84.

Table 6-84 OracleInfoMessageEventArgs Properties

Property	Description
Errors	Specifies the collection of errors generated by the data source
Message	Specifies the error text generated by the data source
Source	Specifies the name of the object that generated the error

OracleInfoMessageEventArgs Public Methods

The `OracleInfoMessageEventArgs` methods are listed in Table 6-85.

Table 6-85 OracleInfoMessageEventArgs Public Methods

Method	Description
`Equals`	Inherited from `System.Object` (Overloaded)
`GetHashCode`	Inherited from `System.Object`
`GetType`	Inherited from `System.Object`
ToString	Returns the string representation of the current instance

6.13.2 OracleInfoMessageEventArgs Static Methods

The `OracleInfoMessageEventArgs` static method is listed in Table 6-86.

Table 6-86 OracleInfoMessageEventArgs Static Method

Method	Description
`Equals`	Inherited from `System.Object` (Overloaded)

6.13.3 OracleInfoMessageEventArgs Properties

The `OracleInfoMessageEventArgs` properties are listed in Table 6-87.

Table 6-87 OracleInfoMessageEventArgs Properties

Property	Description
Errors	Specifies the collection of errors generated by the data source
Message	Specifies the error text generated by the data source
Source	Specifies the name of the object that generated the error

6.13.3.1 Errors

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

Declaration

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

Property Value

The collection of errors.

6.13.3.2 Message

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

Declaration

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

Property Value

The error text.

6.13.3.3 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.

6.13.4 OracleInfoMessageEventArgs Public Methods

The `OracleInfoMessageEventArgs` methods are listed in Table 6-88.

Table 6-88   OracleInfoMessageEventArgs Public Methods

Method	Description
`Equals`	Inherited from `System.Object` (Overloaded)
`GetHashCode`	Inherited from `System.Object`
`GetType`	Inherited from `System.Object`
ToString	Returns the string representation of the current instance

6.13.4.1 ToString

Overrides `Object`

This method returns the string representation of the current instance.

Declaration

```
// C#
public override string ToString();
```

Return Value

Returns the `OracleInfoMessageEventArgs` value in a string representation.

Remarks

If the current instance has a null value, the returned string is null.

6.14 OracleInfoMessageEventHandler Delegate

The `OracleInfoMessageEventHandler` represents the signature of the method that handles the `OracleConnection.InfoMessage` event.

Declaration

```
// C#
public delegate void OracleInfoMessageEventHandler(object sender,
    OracleInfoMessageEventArgs eventArgs);
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver
Assembly	`Oracle.DataAccess.dll`	`Oracle.ManagedDataAccess.dll`
Namespace	`Oracle.DataAccess.Client`	`Oracle.ManagedDataAccess.Client`
.NET Framework	3.5, 4.5, 4.6	4.5, 4.6

Parameters

- *`sender`*

  The source of the event.

- *`eventArgs`*

  The `OracleInfoMessageEventArgs` object that contains the event data.

6.15 OracleLogicalTransaction Class

The `OracleLogicalTransaction` class provides detailed information about the logical transaction status. Applications can conclusively determine the outcome of the running transaction during the last database outage, then act accordingly to commit, complete, or rollback the transaction.

Class Inheritance

```
System.Object

  System.MarshalByRefObject

    Oracle.DataAccess.Client.OracleLogicalTransaction
```

Declaration

```
// C#
public sealed class OracleLogicalTransaction
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver
Assembly	`Oracle.DataAccess.dll`	`Oracle.ManagedDataAccess.dll`
Namespace	`Oracle.DataAccess.Client`	`Oracle.ManagedDataAccess.Client`

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver
.NET Framework	3.5, 4.5, 4.6	4.5, 4.6

Thread Safety

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

6.15.1 OracleLogicalTransaction Members

`OracleLogicalTransaction` members are listed in the following tables.

OracleLogicalTransaction Public Read-Only Properties

`OracleLoigcalTransaction` public read-only properties are listed in Table 6-89

Table 6-89 OracleLogicalTransaction Public Read-Only Properties

Property	Description
Committed	Specifies if the transaction was committed or not
ConnectionString	Specifies a subset of the connection string used for the transaction running during the last database outage
LogicalTransactionId	The logical transaction id is used to determine the commit outcome of the last transaction open in a database session following an outage.
UserCallCompleted	Specifies if the transaction completed and that the information returned may be incomplete and/or not all expected work was completed

OracleLogicalTransaction Methods

`OracleLoigcalTransaction` methods are listed in Table 6-90

Table 6-90 OracleLogicalTransaction Methods

Property	Description
Dispose	This method releases any resources or memory allocated by the object
GetOutcome	This method retrieves the transaction outcome from the database server. The method will determine whether the transaction committed and completed or not.

6.15.2 OracleLogicalTransaction Public Read-Only Properties

`OracleLoigcalTransaction` public read-only properties are listed in Table 6-91

Table 6-91 OracleLogicalTransaction Public Read-Only Properties

Property	Description
Committed	Specifies if the transaction was committed or not
ConnectionString	Specifies a subset of the connection string used for the transaction running during the last database outage
LogicalTransactionId	The logical transaction id is used to determine the commit outcome of the last transaction open in a database session following an outage.
UserCallCompleted	Specifies if the transaction completed and that the information returned may be incomplete and/or not all expected work was completed

6.15.2.1 Committed

This property specifies if the transaction was committed or not.

Declaration

```
// C#
public bool? Committed {get;}
```

Property Value

`bool`.

Remarks

If `GetOutcome()` is not called, the this property holds a `null` value.

Once `GetOutcome()` is called, then this property will hold either `true` or `false`.

Table 6-92 describes the possible outcomes of the `Committed` and `UserCallCompleted` properties.

Table 6-92 Outcome of OracleLogicalTransaction Committed and UserCallCompleted Properties

Committed Value	UserCallCompleted Value	Outcome
`false`	`false`	The call did not execute the commit.
`true`	`true`	The call did execute the commit and there was no additional information to return and no more work to do if that call was a PL/SQL procedure.

Table 6-92 (Cont.) Outcome of OracleLogicalTransaction Committed and UserCallCompleted Properties

Committed Value	UserCallCompleted Value	Outcome
`true`	`false`	The transaction is committed, but the information returned may be incomplete and/or not all expected work was completed. Examples of incomplete information or incomplete work done include: the number of rows modified when using autocommit or commit on success, parameter and function results when calling PL/SQL procedures, or PL/SQL procedures with more work to do after the commit. In order to function correctly, .NET applications that use data returned from the commit must check the `UserCallCompleted` value.

6.15.2.2 ConnectionString

This property specifies a subset of the connection string used for the transaction running during the last database outage.

Declaration

```
// C#
public string ConnectionString {get;}
```

Property Value

The data source as a string.

Remarks

This connection string can be useful if the outcome is not known at the time the exception is thrown due to a service that is down. In such a scenario, use the connection string from this property along with the `LogicalTransactionId` to determine the outcome of the logical transaction by invoking the static `OracleConnection.GetOutcome()` method, once the database or service is back up.

The string returned by this property will contain only the following attributes: `User Id`, `Proxy user Id` (if not null/empty), `Data Source`, and `Pooling` (which will be set to `false`).

6.15.2.3 LogicalTransactionId

The logical transaction id is used to determine the commit outcome of the last transaction open in a database session following an outage.

Declaration

```
// C#
public byte LogicalTransactionId {get;}
```

Property Value

byte[]

Remarks

This logical transaction id can be useful if the outcome is not known at the time the exception is thrown due to a service that is down. In such a scenario, use the `byte[]` returned from this property (along with the `ConnectionString`) to determine the outcome of the logical transaction by invoking the static `OracleConnection.GetOutcome()` method, once the database or service is back up.

This property will return a non-null value *only* when the outcome is not known. For example when database or service is down, then the outcome is not known.

`LogicalTransactionId` property will return `null` if the connection has participated in a distributed transaction.

6.15.2.4 UserCallCompleted

This property specifies if the transaction completed and that the information returned may be incomplete and/or not all expected work was completed.

Declaration

```
// C#
public bool? UserCallCompleted {get;}
```

Property Value

`bool`

Remarks

If `GetOutcome()` is not called, the this property holds a `null` value.

Once `GetOutcome()` is called, then this property will hold either `true` or `false`.

Table 6-92 describes the possible outcomes of the `Committed` and `UserCallCompleted` properties.

6.15.3 OracleLogicalTransaction Methods

`OracleLoigcalTransaction` methods are listed in Table 6-93

Table 6-93 OracleLogicalTransaction Methods

Property	Description
Dispose	This method releases any resources or memory allocated by the object
GetOutcome	This method retrieves the transaction outcome from the database server. The method will determine whether the transaction committed and completed or not.

6.15.3.1 Dispose

This method releases any resources or memory allocated by the object

Declaration

```
// C#
public void Dispose();
```

Implements

`IDisposable`

Remarks

The `Dispose` method also closes the `OracleLogicalTransaction` object.

6.15.3.2 GetOutcome

The `GetOutcome` method retrieves the transaction outcome from the database server. The method will determine whether the transaction committed and completed or not.

Overload List:

- `GetOutcome(string` *`constring`*, `byte[]` *`ltxid`*, `out bool?` *`bCommitted`*, `out bool?` *`bUserCallCompleted`*)

  The application can use this static method to determine the outcome if the outcome was not known when the exception was raised.

  The application will need to obtain the connection string and `logical transaction id` from the `OracleException.OracleLogicalTransaction` object before calling this method.

  The supplied connection string will be used to establish a connection to the database to determine the outcome of the provided logical transaction id.

  ODP.NET implicitly calls `GetOutcome` under the following conditions:

  - Transaction Guard is enabled on the service
  - OracleException is raised
  - The exception is a recoverable error

  When all of the above is true, then the `OracleException.OracleLogicalTransaction` property will be non-null.

  If a connection is involved in a distributed transaction, then `GetOutcome` is *not* called implicitly and the `OracleException.OracleLogicalTransaction.LogicalTransactionId` property returns null.

Note:

Once one server round-trip is incurred for the `GetOutcome() invocation`, the PL/SQL `ForceOutcome` is never invoked again against the server for a given `OracleLogicalTransaction` object.

6.16 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`

      `Oracle.DataAccess.Client.OracleParameter`

Declaration

```
// C#
public sealed class OracleParameter : DbParameter, IDisposable, ICloneable
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver
Assembly	`Oracle.DataAccess.dll`	`Oracle.ManagedDataAccess.dll`
Namespace	`Oracle.DataAccess.Client`	`Oracle.ManagedDataAccess.Client`
.NET Framework	3.5, 4.5, 4.6	4.5, 4.6

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[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);
  }
}
```

6.16.1 OracleParameter Members

`OracleParameter` members are listed in the following tables.

OracleParameter Constructors

`OracleParameter` constructors are listed in Table 6-94.

Table 6-94 OracleParameter Constructors

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

OracleParameter Static Methods

`OracleParameter` static methods are listed in Table 6-95.

Table 6-95 OracleParameter Static Methods

Method	Description
`Equals`	Inherited from `System.Object` (Overloaded)

OracleParameter Properties

`OracleParameter` properties are listed in Table 6-96.

Table 6-96 OracleParameter Properties

Property	Description
ArrayBindSize	Specifies the input or output size of elements in `Value` property of a parameter before or after an Array Bind or PL/SQL Associative Array Bind execution
ArrayBindStatus	Specifies the input or output status of elements in `Value` property of a parameter before or after an Array Bind or PL/SQL Associative Array Bind execution
CollectionType	Specifies whether or not the `OracleParameter` represents a collection, and if so, specifies the collection type
DbType	Specifies the data type of the parameter using the `Data.DbType` enumeration type
Direction	Specifies whether the parameter is input-only, output-only, bi-directional, or a stored function return value parameter
IsNullable	Not supported
Offset	Specifies the offset to the `Value` property or offset to the elements in the `Value` property
OracleDbType	Specifies the Oracle data type
OracleDbTypeEx	Specifies the Oracle data type to bind the parameter as, but returns a .NET type as output
ParameterName	Specifies the name of the parameter
Precision	Specifies the maximum number of digits used to represent the `Value` property
Scale	Specifies the number of decimal places to which `Value` property is resolved
Size	Specifies the maximum size, in bytes or characters, of the data transmitted to or from the database. For PL/SQL Associative Array Bind, `Size` specifies the maximum number of elements in PL/SQL Associative Array
SourceColumn	Specifies the name of the `DataTable` Column of the `DataSet`
SourceColumnNullMapping	Specifies a value which indicates whether the source column is nullable
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`

OracleParameter Public Methods

`OracleParameter` public methods are listed in Table 6-97.

Table 6-97 OracleParameter Public Methods

Public Method	Description
Clone	Creates a shallow copy of an `OracleParameter` object
`CreateObjRef`	Inherited from `System.MarshalByRefObject`
Dispose	Releases allocated resources
`Equals`	Inherited from `System.Object` (Overloaded)
`GetHashCode`	Inherited from `System.Object`
`GetLifetimeService`	Inherited from `System.MarshalByRefObject`
`GetType`	Inherited from `System.Object`
`InitializeLifetimeService`	Inherited from `System.MarshalByRefObject`
ResetDbType	Resets the type associated with the parameter so that it can infer its type from the value passed in the parameter
ResetOracleDbType	Resets the type associated with the parameter so that it can infer its type from the value passed in the parameter
ToString	Returns the string representation of the current instance

6.16.2 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.

6.16.2.1 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`

6.16.2.2 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`

6.16.2.3 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`
- `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`

6.16.2.4 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

```
// C#
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`
- `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`

6.16.2.5 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

```
// 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:

- `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`

6.16.2.6 OracleParameter(string, OracleDbType, int)

This constructor instantiates a new instance of the `OracleParameter` class using the supplied parameter name, data type, and size.

Declaration

```
// 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`
- `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`

6.16.2.7 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:

- `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`

6.16.2.8 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.
- *`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.
- *`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 `OracleType`s.

Remarks

Unless explicitly set in the constructor, all the properties have the default values.

Default Values:

- `DbType` - `String`
- `ParameterDirection` - `Input`
- `isNullable` - `true`
- `offset` - `0`
- `OracleDbType` - `Varchar2`
- `ParameterAlias` - Empty string
- `ParameterName` - Empty string
- `Precision` - `0`
- `Size` - `0`
- `SourceColumn` - Empty string
- `SourceVersion` - `Current`
- `ArrayBindStatus` - `Success`
- `Value` - `null`

6.16.2.9 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`

6.16.3 OracleParameter Static Methods

The `OracleParameter` static method is listed in Table 6-98.

Table 6-98 OracleParameter Static Method

Method	Description
`Equals`	Inherited from `System.Object` (Overloaded)

6.16.4 OracleParameter Properties

`OracleParameter` properties are listed in Table 6-99.

Table 6-99 OracleParameter Properties

Property	Description
ArrayBindSize	Specifies the input or output size of elements in `Value` property of a parameter before or after an Array Bind or PL/SQL Associative Array Bind execution
ArrayBindStatus	Specifies the input or output status of elements in `Value` property of a parameter before or after an Array Bind or PL/SQL Associative Array Bind execution
CollectionType	Specifies whether or not the `OracleParameter` represents a collection, and if so, specifies the collection type
DbType	Specifies the data type of the parameter using the `Data.DbType` enumeration type
Direction	Specifies whether the parameter is input-only, output-only, bi-directional, or a stored function return value parameter
IsNullable	Not supported
Offset	Specifies the offset to the `Value` property or offset to the elements in the `Value` property
OracleDbType	Specifies the Oracle data type
OracleDbTypeEx	Specifies the Oracle data type to bind the parameter as, but returns a .NET type as output
ParameterName	Specifies the name of the parameter
Precision	Specifies the maximum number of digits used to represent the `Value` property
Scale	Specifies the number of decimal places to which `Value` property is resolved
Size	Specifies the maximum size, in bytes or characters, of the data transmitted to or from the database. For PL/SQL Associative Array Bind, `Size` specifies the maximum number of elements in PL/SQL Associative Array
SourceColumn	Specifies the name of the `DataTable` Column of the `DataSet`
SourceColumnNullMapping	Specifies a value which indicates whether the source column is nullable
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`

6.16.4.1 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)";

    cmd.ExecuteNonQuery();

    Console.WriteLine("Record for employee id {0} has been inserted.",
      maxno + 10);
    Console.WriteLine("Record for employee id {0} has been inserted.",
      maxno + 11);

    prm[0].Dispose();
    prm[1].Dispose();
    prm[2].Dispose();
    cmd.Dispose();

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

6.16.4.2 ArrayBindStatus

This property specifies the input or output status of each element in the `Value` property before or after an Array Bind or PL/SQL Associative Array execution.

Declaration

```
// C#
public OracleParameterStatus[] ArrayBindStatus { get; set; }
```

Property Value

An array of `OracleParameterStatus` enumerated values.

Exceptions

`ArgumentOutofRangeException` - The `Status` value specified is invalid.

Remarks

Default = `null`.

`ArrayBindStatus` is used for Array Bind and PL/SQL Associative Array execution only.

Before execution, `ArrayBindStatus` indicates the bind status of each element in the `Value` property. After execution, it contains the execution status of each element in the `Value` property.

6.16.4.3 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`.

6.16.4.4 DbType

This property specifies the data type of the parameter using the `Data.DbType` enumeration type.

Declaration

```
// C#
public override DbType DbType {get; set; }
```

Property Value

A `DbType` enumerated value.

Implements

`IDataParameter`

Exceptions

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

Remarks

Default = `DbType.String`

`DbType` is the 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`.

6.16.4.5 Direction

This property specifies whether the parameter is input-only, output-only, bi-directional, or a stored function return value parameter.

Declaration

```
// C#
public override ParameterDirection Direction { get; set; }
```

Property Value

A `ParameterDirection` enumerated value.

Implements

`IDataParameter`

Exceptions

`ArgumentOutOfRangeException` - The `ParameterDirection` value specified is invalid.

Remarks

Default = `ParameterDirection.Input`

Possible values: `Input`, `InputOutput`, `Output`, and `ReturnValue`.

6.16.4.6 IsNullable

This property is not supported.

Declaration

```
// C#
public override bool IsNullable { get; set; }
```

Implements

`IDataParameter`

Property Value

This property is not supported.

6.16.4.7 Offset

This property specifies the offset to the `Value` property.

Declaration

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

Property Value

An `int` that specifies the offset.

Exceptions

`ArgumentOutOfRangeException` - The `Offset` value specified is invalid.

Remarks

Default = `0`

For Array Bind and PL/SQL Associative Array Bind, `Offset` applies to every element in the `Value` property.

The `Offset` property is used for binary and string data types. The `Offset` property represents the number of bytes for binary types and the number of characters for strings. The count for strings does not include the terminating character if a `null` is referenced. The `Offset` property is used by parameters of the following types:

- `OracleDbType.BFile`
- `OracleDbType.Blob`
- `OracleDbType.LongRaw`
- `OracleDbType.Raw`
- `OracleDbType.Char`
- `OracleDbType.Clob`
- `OracleDbType.NClob`
- `OracleDbType.NChar`
- `OracleDbType.NVarchar2`
- `OracleDbType.Varchar2`

6.16.4.8 OracleDbType

This property specifies the Oracle data type.

Declaration

```
// C#
public OracleDbType OracleDbType { get; set; }
```

Property Value

An `OracleDbType` enumerated value.

Remarks

Default = `OracleDbType.Varchar2`

If the `OracleParameter` object is used for Array Bind or PL/SQL Associative Array Bind execution, `OracleDbType` is the 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`.

6.16.4.9 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.

6.16.4.10 ParameterName

This property specifies the name of the parameter.

Declaration

```
// C#
public override string ParameterName { get; set; }
```

Property Value

`String`

Implements

`IDataParameter`

Remarks

Default = `null`

Oracle supports `ParameterName` up to 30 characters.

6.16.4.11 Precision

This property specifies the maximum number of digits used to represent the `Value` property.

Declaration

```
// C#
Public byte Precision { get; set; }
```

Property Value

`byte`

Remarks

Default = `0`

The `Precision` property is used by parameters of type `OracleDbType.Decimal`.

Oracle supports `Precision` range from `0` to `38`.

For Array Bind and PL/SQL Associative Array Bind, `Precision` applies to each element in the `Value` property.

6.16.4.12 Scale

This property specifies the number of decimal places to which `Value` property is resolved.

Declaration

```
// C#
public byte Scale { get; set; }
```

Property Value

```
byte
```

Remarks

Default = `0`.

`Scale` is used by parameters of type `OracleDbType.Decimal`.

Oracle supports `Scale` between `-84` and `127`.

For Array Bind and PL/SQL Associative Array Bind, `Scale` applies to each element in the `Value` property.

6.16.4.13 Size

This property specifies the maximum size, in bytes or characters, of the data transmitted to or from the database.

Declaration

```
// C#
public override int Size { get; set;}
```

Property Value

```
int
```

Exceptions

`ArgumentOutOfRangeException` - The `Size` value specified is invalid.

`InvalidOperationException` - The `Size` = 0 when the `OracleParameter` object is used to bind a PL/SQL Associative Array.

Remarks

For PL/SQL Associative Array Bind, `Size` specifies the maximum number of elements in PL/SQL Associative Array.

If `Size` is not explicitly set, it is inferred from the actual size of the specified parameter value when binding only for input parameters. Output parameters must have their size defined explicitly.

The default value is `0`.

Before execution, this property specifies the maximum size to be bound in the `Value` property. After execution, it contains the size of the type in the `Value` property.

`Size` is used for parameters of the following types:

- `OracleDbType.Blob`
- `OracleDbType.Char`
- `OracleDbType.Clob`
- `OracleDbType.LongRaw`
- `OracleDbType.NChar`
- `OracleDbType.NClob`
- `OracleDbType.NVarchar2`
- `OracleDbType.Raw`
- `OracleDbType.Varchar2`

The value of `Size` is handled as follows:

- Fixed length data types: ignored
- 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.

6.16.4.14 SourceColumn

This property specifies the name of the `DataTable` Column of the `DataSet`.

Declaration

```
// C#
public override string SourceColumn { get; set; }
```

Property Value

A `string`.

Implements

`IDataParameter`

Remarks

Default = empty string

6.16.4.15 SourceColumnNullMapping

This property specifies a value which indicates whether the source column is nullable.

Declaration

```
// 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`.

6.16.4.16 SourceVersion

This property specifies the `DataRowVersion` value to use when loading the `Value` property of the parameter.

Declaration

```
// 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()`.

6.16.4.17 Status

This property indicates the status of the execution related to the data in the `Value` property.

Declaration

```
// C#
public OracleParameterStatus Status { get; set; }
```

Property Value

An `OracleParameterStatus` enumerated value.

Exceptions

`ArgumentOutOfRangeException` - The `Status` value specified is invalid.

Remarks

Default = `OracleParameterStatus.Success`

Before execution, this property indicates the bind status related to the `Value` property. After execution, it returns the status of the execution.

`Status` indicates 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.

6.16.4.18 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 OracleParamter.UdtTypeName property differs if the application binds a custom object that represents a subtype of the parameter type.

6.16.4.19 Value

This property specifies the value of the Parameter.

Declaration

```
// C#
public override object Value { get; set; }
```

Property Value

An object.

Implements

IDataParameter

Exceptions

ArgumentException - The Value property specified is invalid.

InvalidArgumentException- The Value property specified is invalid.

Remarks

Default = null

If the OracleParameter object is used for Array Bind or PL/SQL Associative Array, Value is an array of parameter values.

The Value property can be overwritten by OracleDataAdapter.Update().

The provider attempts to convert any type of value if it supports the IConvertible interface. Conversion errors occur if the specified type is not compatible with the value.

When sending a null parameter value to the database, the user must specify DBNull, not null. The null value in the system is an empty object that has no value. DBNull is used to represent null values. The user can also specify a null value by setting Status to OracleParameterStatus.NullValue. In this case, the provider sends a null value to the database.

If neither OracleDbType nor DbType are set, their values can be inferred by Value. Please see the following for related information:

- Tables in section "Inference of DbType and OracleDbType from Value"
- "ArrayBindCount "
- "ArrayBindSize "
- "ArrayBindStatus "
- "OracleDbType Enumeration"

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`.

6.16.5 OracleParameter Public Methods

`OracleParameter` public methods are listed in Table 6-100.

Table 6-100 OracleParameter Public Methods

Public Method	Description
Clone	Creates a shallow copy of an `OracleParameter` object

Table 6-100 (Cont.) OracleParameter Public Methods

Public Method	Description
`CreateObjRef`	Inherited from `System.MarshalByRefObject`
Dispose	Releases allocated resources
`Equals`	Inherited from `System.Object` (Overloaded)
`GetHashCode`	Inherited from `System.Object`
`GetLifetimeService`	Inherited from `System.MarshalByRefObject`
`GetType`	Inherited from `System.Object`
`InitializeLifetimeService`	Inherited from `System.MarshalByRefObject`
ResetDbType	Resets the type associated with the parameter so that it can infer its type from the value passed in the parameter
ResetOracleDbType	Resets the type associated with the parameter so that it can infer its type from the value passed in the parameter
ToString	Returns the string representation of the current instance

6.16.5.1 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();
  }
}
```

6.16.5.2 Dispose

This method releases resources allocated for an `OracleParameter` object.

Declaration

```
// C#
public void Dispose();
```

Implements

```
IDisposable
```

6.16.5.3 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

```
// 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.

6.16.5.4 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

```
// 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.

6.16.5.5 ToString

Overrides `Object`

This method returns the string representation of the current instance.

Declaration

```
// C#
public override string ToString();
```

Return Value

Returns the `OracleParameter` value in a string representation.

Remarks

If the current instance has a null value, the returned string is null.

6.17 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

      Oracle.DataAccess.Client.OracleParameterCollection
```

Declaration

```
// C#
public sealed class OracleParameterCollection : DbParameterCollection,
    IDataParameterCollection, IList, ICollection, IEnumerable
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver
Assembly	`Oracle.DataAccess.dll`	`Oracle.ManagedDataAccess.dll`
Namespace	`Oracle.DataAccess.Client`	`Oracle.ManagedDataAccess.Client`
.NET Framework	3.5, 4.5, 4.6	4.5, 4.6

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();
  }
}
```

6.17.1 OracleParameterCollection Members

`OracleParameterCollection` members are listed in the following tables.

OracleParameterCollection Static Methods

`OracleParameterCollection` static methods are listed in Table 6-101.

Table 6-101 OracleParameterCollection Static Methods

Method	Description
`Equals`	Inherited from `System.Object` (Overloaded)

OracleParameterCollection Properties

`OracleParameterCollection` properties are listed in Table 6-102.

Table 6-102 OracleParameterCollection Properties

Property	Description
Count	Specifies the number of `OracleParameters` in the collection
Item	Gets and sets the `OracleParameter` object (Overloaded)
IsFixedSize	Gets a value that indicates whether the `OracleParameterCollection` has a fixed size
IsReadOnly	Gets a value that indicates whether the `OracleParameterCollection` is read-only
IsSynchronized	Gets a value that indicates whether the `OracleParameterCollection` is synchronized.
SyncRoot	Gets an object that can be used to synchronize access to the `OracleParameterCollection`

OracleParameterCollection Public Methods

`OracleParameterCollection` public methods are listed in Table 6-103.

Table 6-103 OracleParameterCollection Public Methods

Public Method	Description
Add	Adds objects to the collection (Overloaded)
AddRange	Adds elements to the end of the `OracleParameterCollection`
Clear	Removes all the `OracleParameter` objects from the collection
Contains	Indicates whether or not objects exist in the collection (Overloaded)
CopyTo	Copies `OracleParameter` objects from the collection, starting with the supplied `index` to the supplied `array`
`CreateObjRef`	Inherited from `System.MarshalByRefObject`
`Equals`	Inherited from `System.Object` (Overloaded)
GetEnumerator	Returns an enumerator that iterates through the `OracleParameterCollection`
`GetHashCode`	Inherited from `System.Object`
`GetLifetimeService`	Inherited from `System.MarshalByRefObject`
`GetType`	Inherited from `System.Object`
`InitializeLifetimeService`	Inherited from `System.MarshalByRefObject`
IndexOf	Returns the `index` of the objects in the collection (Overloaded)
Insert	Inserts the supplied `OracleParameter` to the collection at the specified `index`
Remove	Removes objects from the collection
RemoveAt	Removes objects from the collection by location (Overloaded)
`ToString`	Inherited from `System.Object`

6.17.2 OracleParameterCollection Static Methods

The `OracleParameterCollection` static method is listed in Table 6-104.

Table 6-104 OracleParameterCollection Static Method

Method	Description
`Equals`	Inherited from `System.Object` (Overloaded)

6.17.3 OracleParameterCollection Properties

`OracleParameterCollection` properties are listed in Table 6-105.

Table 6-105 OracleParameterCollection Properties

Property	Description
Count	Specifies the number of `OracleParameters` in the collection
Item	Gets and sets the `OracleParameter` object (Overloaded)
IsFixedSize	Gets a value that indicates whether the `OracleParameterCollection` has a fixed size
IsReadOnly	Gets a value that indicates whether the `OracleParameterCollection` is read-only
IsSynchronized	Gets a value that indicates whether the `OracleParameterCollection` is synchronized.
SyncRoot	Gets an object that can be used to synchronize access to the `OracleParameterCollection`

6.17.3.1 Count

This property specifies the number of `OracleParameter` objects in the collection.

Declaration

```
// C#
public override int Count {get;}
```

Property Value

The number of `OracleParameter` objects.

Implements

`ICollection`

Remarks

Default = `0`

6.17.3.2 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`.

6.17.3.3 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.

6.17.3.4 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.

6.17.3.5 IsFixedSize

`IsFixedSize` gets a value that indicates whether the `OracleParameterCollection` has a fixed size.

Declaration

```
// C#
public override bool IsFixedSize { get; };
```

Property Value

Returns `true` if the `OracleParameterCollection` has a fixed size; otherwise `false`.

Implements

`IList`

6.17.3.6 IsReadOnly

`IsReadOnly` gets a value that indicates whether the `OracleParameterCollection` is read-only.

Declaration

```
// C#
public override bool IsReadOnly { get; };
```

Property Value

Returns `true` if the `OracleParameterCollection` is read only; otherwise `false`.

Implements

`IList`

6.17.3.7 IsSynchronized

`IsSynchronized` gets a value that indicates whether the `OracleParameterCollection` is synchronized.

Declaration

```
// C#
public override bool IsSynchronized { get; };
```

Property Value

Returns `true` if the `OracleParameterCollection` is synchronized; otherwise `false`.

Implements

`ICollection`

6.17.3.8 SyncRoot

`SyncRoot` gets an object that can be used to synchronize access to the `OracleParameterCollection`.

Declaration

```
// C#
public override Object SyncRoot { get; };
```

Property Value

An object that can be used to synchronize access to the `OracleParameterCollection`.

Implements

`ICollection`

6.17.4 OracleParameterCollection Public Methods

`OracleParameterCollection` public methods are listed in Table 6-106.

Table 6-106 OracleParameterCollection Public Methods

Public Method	Description
Add	Adds objects to the collection (Overloaded)
AddRange	Adds elements to the end of the `OracleParameterCollection`
Clear	Removes all the `OracleParameter` objects from the collection
Contains	Indicates whether or not objects exist in the collection (Overloaded)
CopyTo	Copies `OracleParameter` objects from the collection, starting with the supplied `index` to the supplied `array`
`CreateObjRef`	Inherited from `System.MarshalByRefObject`
`Equals`	Inherited from `System.Object` (Overloaded)
GetEnumerator	Returns an enumerator that iterates through the `OracleParameterCollection`
`GetHashCode`	Inherited from `System.Object`
`GetLifetimeService`	Inherited from `System.MarshalByRefObject`
`GetType`	Inherited from `System.Object`
`InitializeLifetimeService`	Inherited from `System.MarshalByRefObject`
IndexOf	Returns the `index` of the objects in the collection (Overloaded)
Insert	Inserts the supplied `OracleParameter` to the collection at the specified `index`
Remove	Removes objects from the collection
RemoveAt	Removes objects from the collection by location (Overloaded)
`ToString`	Inherited from `System.Object`

6.17.4.1 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 and Oracle.ManagedDataAccess.Client Namespaces"
- OracleParameterCollection Class
- OracleParameterCollection Members

6.17.4.2 Add(object)

This method adds the supplied object to the collection.

Declaration

```
// 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

`InvalidCastException` - The supplied *obj* cannot be cast to an `OracleParameter` object.

6.17.4.3 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.

6.17.4.4 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.

6.17.4.5 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.

6.17.4.6 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.

6.17.4.7 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);

    prm.Dispose();
    cmd.Dispose();
  }
}
```

6.17.4.8 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.

6.17.4.9 Add(string, OracleDbType, int)

This method adds an `OracleParameter` object to the collection using the supplied name, database type, and size.

Declaration

```
// C#
public OracleParameter Add(string name, OracleDbType dbType, int size);
```

Parameters

- *name*

  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();
  }
}
```

6.17.4.10 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`.

6.17.4.11 Add(string, OracleDbType, int, ParameterDirection, bool, byte, byte, string, DataRowVersion, object)

This method adds an `OracleParameter` object to the collection using the supplied name, database type, size, direction, null indicator, precision, scale, source column, source version, and parameter value.

Declaration

```
// C#
public OracleParameter Add(string name, OracleDbType dbType, int size,
    ParameterDirection dir, bool isNullable, byte precision,
    byte scale, string srcColumn, DataRowVersion version, object val);
```

Parameters

- *`name`*

  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.

6.17.4.12 AddRange

This method adds elements to the end of the `OracleParameterCollection`.

Declaration

```
// C#
public override void AddRange(Array paramArray );
```

Parameters

`paramArray`

An array of `OracleParameter` objects.

Exceptions

`ArgumentNullException` - The input parameter is null.

6.17.4.13 Clear

This method removes all the `OracleParameter` objects from the collection.

Declaration

```
// C#
public override void Clear();
```

Implements

`IList`

Example

```
// C#

using System;
using Oracle.DataAccess.Client;

class ClearSample
{
  static void Main()
  {
    OracleCommand cmd = new OracleCommand();
    // Add parameter to the OracleParameterCollection
    OracleParameter prm = cmd.Parameters.Add("MyParam", OracleDbType.Decimal);

    // Prints "cmd.Parameters.Count = 1"
    Console.WriteLine("cmd.Parameters.Count = " + cmd.Parameters.Count);

    // Clear all parameters in the OracleParameterCollection
    cmd.Parameters.Clear();

    // Prints "cmd.Parameters.Count = 0"
    Console.WriteLine("cmd.Parameters.Count = " + cmd.Parameters.Count);

    prm.Dispose();
    cmd.Dispose();
  }
}
```

6.17.4.14 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.

6.17.4.15 Contains(object)

This method indicates whether or not the supplied object exists in the collection.

Declaration

```
// 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

```
// C#

using System;
using Oracle.DataAccess.Client;

class ContainsSample
{
  static void Main()
  {
    OracleCommand cmd = new OracleCommand();

    // Add parameter to the OracleParameterCollection
    OracleParameter prm1 = cmd.Parameters.Add("MyParam", OracleDbType.Decimal);

    // Check if the OracleParameterCollection contains prm1
    bool bContains = cmd.Parameters.Contains(prm1);

    // Prints "bContains = True"
    Console.WriteLine("bContains = " + bContains);

    OracleParameter prm2 = new OracleParameter();

    // Check if the OracleParameterCollection contains prm2
    bContains = cmd.Parameters.Contains(prm2);

    // Prints "bContains = False"
    Console.WriteLine("bContains = " + bContains);

    prm1.Dispose();
    prm2.Dispose();
    cmd.Dispose();
  }
}
```

6.17.4.16 Contains(string)

This method indicates whether or not an `OracleParameter` object exists in the collection using the supplied string.

Declaration

```
// 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();
  }
}
```

6.17.4.17 CopyTo

This method copies `OracleParameter` objects from the collection, starting with the supplied `index` to the supplied `array`.

Declaration

```
// C#
public override void CopyTo(Array array, int index);
```

Parameters

- *`array`*

  The specified array.

- *`index`*

The array index.

Implements

`ICollection`

6.17.4.18 GetEnumerator

`GetEnumerator` returns an enumerator that iterates through the `OracleParameterCollection`.

Declaration

```
// C#
public override IEnumerator GetEnumerator();
```

Implements

`IEnumerable`

6.17.4.19 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.

6.17.4.20 IndexOf(object)

This method returns the index of the `OracleParameter` object in the collection.

Declaration

```
// 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.

6.17.4.21 IndexOf(String)

This method returns the `index` of the `OracleParameter` object with the specified name in the collection.

Declaration

```
// C#
public override int IndexOf(String name);
```

Parameters

- *`name`*

  The name of parameter.

Return Value

Returns the `index` of the supplied `OracleParameter` in the collection.

Implements

`IDataParameterCollection`

6.17.4.22 Insert

This method inserts the supplied `OracleParameter` object to the collection at the specified `index`.

Declaration

```
// 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.

6.17.4.23 Remove

This method removes the supplied `OracleParameter` from the collection.

Declaration

```
// 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();
  }
}
```

6.17.4.24 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.

6.17.4.25 RemoveAt(int)

This method removes from the collection the `OracleParameter` object located at the index specified by the supplied index.

Declaration

```
// C#
public override void RemoveAt(int index);
```

Parameters

- *`index`*

  The specified index from which the `OracleParameter` is to be removed.

Implements

`IList`

6.17.4.26 RemoveAt(String)

This method removes from the collection the `OracleParameter` object specified by the supplied name.

Declaration

```
// C#
public override void RemoveAt(String name);
```

Parameters

- *`name`*

  The name of the `OracleParameter` object to be removed from the collection.

Implements

`IDataParameterCollection`

6.18 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.Security.CodeAccessPermission
      System.Data.Common.DBDataPermission
        Oracle.DataAccess.Client.OraclePermission
```

Declaration

```
// C#
public class OraclePermission: DBDataPermission
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver
Assembly	`Oracle.DataAccess.dll`	`Oracle.ManagedDataAccess.dll`
Namespace	`Oracle.DataAccess.Client`	`Oracle.ManagedDataAccess.Client`
.NET Framework	3.5, 4.5, 4.6	4.5, 4.6

Thread Safety

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

6.18.1 OraclePermission Members

`OraclePermission` members are listed in the following tables.

OraclePermission Constructors

The `OraclePermission` constructor is listed in Table 6-107.

Table 6-107   OraclePermission Constructor

Constructor	Description
OraclePermission Constructor	Instantiates a new instance of the `OraclePermission` class.

OraclePermission Static Methods

The `OraclePermission` static methods are listed in Table 6-108.

Table 6-108 OraclePermission Static Methods

Static Method	Description
`Equals`	Inherited from `System.Object`
`ReferenceEquals`	Inherited from `System.Object`
`RevertAll`	Inherited from `CodeAccessPermission`
`RevertAssert`	Inherited from `CodeAccessPermission`
`RevertDeny`	Inherited from `CodeAccessPermission`
`RevertPermitOnly`	Inherited from `CodeAccessPermission`

OraclePermission Public Properties

The `OraclePermission` public methods are listed in Table 6-112.

Table 6-109 OraclePermission Public Properties

Public Properties	Description
`AllowBlankPassword`	Inherited from `DBDataPermission` `OraclePermission` does not support this property.

OraclePermission Public Methods

The `OraclePermission` public methods are listed in Table 6-110.

Table 6-110 OraclePermission Public Methods

Public Method	Description
Add	Adds a new connection string fragment and a list of restricted keywords to the `OraclePermission` object
`Assert`	Inherited from `CodeAccessPermission`
Copy	Returns a copy of the current permission object
`Demand`	Inherited from `CodeAccessPermission`
`Deny`	Inherited from `CodeAccessPermission`
`Equals`	Inherited from `CodeAccessPermission`
`FromXml`	Inherited from `DBDataPermission`
`GetHashCode`	Inherited from `CodeAccessPermission`
`GetType`	Inherited from `System.Object`
`Intersect`	Inherited from `DBDataPermission`
IsSubsetOf	Returns a boolean value that indicates whether or not the current permission is a subset of the target permission

Table 6-110 (Cont.) OraclePermission Public Methods

Public Method	Description
`IsUnrestricted`	Inherited from `DBDataPermission`
`PermitOnly`	Inherited from `CodeAccessPermission`
`ToString`	Inherited from `CodeAccessPermission`
`ToXml`	Inherited from `DBDataPermission`
`Union`	Inherited from `DBDataPermission`

6.18.2 OraclePermission Constructor

The `OraclePermission` constructor instantiates a new instance of the `OraclePermission` class.

Declaration

```
// C#
public OraclePermission (PermissionState state);
```

Parameters

- *`state`*

  The *`state`* parameter takes one of the following two values: `PermissionState.None` or `PermissionState.Unrestricted`.

Exceptions

`ArgumentException` - The `PermissionState` value is invalid.

6.18.3 OraclePermission Static Methods

The `OraclePermission` static methods are listed in Table 6-111.

Table 6-111 OraclePermission Static Methods

Static Method	Description
`Equals`	Inherited from `System.Object`
`ReferenceEquals`	Inherited from `System.Object`
`RevertAll`	Inherited from `CodeAccessPermission`
`RevertAssert`	Inherited from `CodeAccessPermission`
`RevertDeny`	Inherited from `CodeAccessPermission`
`RevertPermitOnly`	Inherited from `CodeAccessPermission`

6.18.4 OraclePermission Public Properties

The `OraclePermission` public methods are listed in Table 6-112.

Table 6-112 OraclePermission Public Properties

Public Properties	Description
`AllowBlankPassword`	Inherited from `DBDataPermission` `OraclePermission` ignores the value of this property. Any value set for this property, for an `OraclePermission` object, is ignored.

6.18.5 OraclePermission Public Methods

The `OraclePermission` public methods are listed in Table 6-113.

Table 6-113 OraclePermission Public Methods

Public Method	Description
Add	Adds a new connection string fragment and a list of restricted keywords to the `OraclePermission` object
`Assert`	Inherited from `CodeAccessPermission`
Copy	Returns a copy of the current permission object
`Demand`	Inherited from `CodeAccessPermission`
`Deny`	Inherited from `CodeAccessPermission`
`Equals`	Inherited from `CodeAccessPermission`
`FromXml`	Inherited from `DBDataPermission`
`GetHashCode`	Inherited from `CodeAccessPermission`
`GetType`	Inherited from `System.Object`
`Intersect`	Inherited from `DBDataPermission`
IsSubsetOf	Returns a boolean value that indicates whether or not the current permission is a subset of the target permission
`IsUnrestricted`	Inherited from `DBDataPermission`
`PermitOnly`	Inherited from `CodeAccessPermission`
`ToString`	Inherited from `CodeAccessPermission`
`ToXml`	Inherited from `DBDataPermission`
`Union`	Inherited from `DBDataPermission`

6.18.5.1 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);
```

Parameters

- *`connStr`*

  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 *`keyRestict`* 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.

```
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.

```
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 attributes` keywords are allowed in the connection string.

6.18.5.2 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.

6.18.5.3 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.

6.19 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.Security.Permissions.SecurityAttribute
```

```
System.Security.Permissions.CodeAccessSecurityAttribute
  System.Data.Common.DBDataPermissionAttribute
    Oracle.DataAccess.Client.OraclePermissionAttribute
```

Declaration

```
// C#
[Serializable, AttributeUsage(AttributeTargets.Method |
AttributeTargets.Constructor | AttributeTargets.Class | AttributeTargets.Struct |
AttributeTargets.Assembly, AllowMultiple = true, Inherited = false)]
public sealed class OraclePermissionAttribute: DBDataPermissionAttribute
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver
Assembly	`Oracle.DataAccess.dll`	`Oracle.ManagedDataAccess.dll`
Namespace	`Oracle.DataAccess.Client`	`Oracle.ManagedDataAccess.Client`
.NET Framework	3.5, 4.5, 4.6	4.5, 4.6

Thread Safety

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

6.19.1 OraclePermissionAttribute Members

`OraclePermissionAttribute` members are listed in the following tables.

OraclePermissionAttribute Constructor

The `OraclePermissionAttribute` constructor is listed in Table 6-114.

Table 6-114 OraclePermission Constructor

Constructor	Description
OraclePermissionAttribute Constructor	Instantiates a new instance of the `OraclePermissionAttribute` class.

OraclePermissionAttribute Static Methods

The `OraclePermissionAttribute` static methods are listed in Table 6-115.

Table 6-115 OraclePermissionAttribute Static Methods

Static Methods	Description
`GetCustomAttribute`	Inherited from `System.Attribute` (Overloaded)
`GetCustomAttributes`	Inherited from `System.Attribute`(Overloaded)
`IsDefined`	Inherited from `System.Attribute`(Overloaded)
`ReferenceEquals`	Inherited from `System.Object`

OraclePermissionAttribute Public Properties

The `OraclePermissionAttribute` public properties are listed in Table 6-116.

Table 6-116 OraclePermissionAttribute Public Properties

Public Properties	Description
`Action`	Inherited from `SecurityAttribute`
`AllowBlankPassword`	Inherited from `DBDataPermissionAttribute`. `OraclePermissionAttribute` ignores this property. Any value set for this property, for an `OraclePermissionAttribute` object, is ignored.
`ConnectionString`	Inherited from `DBDataPermissionAttribute`
`KeyRestrictionBehavior`	Inherited from `DBDataPermissionAttribute`
`KeyRestrictions`	Inherited from `DBDataPermissionAttribute`
`TypeId`	Inherited from `System.Attribute`
`Unrestricted`	Inherited from `SecurityAttribute`

OraclePermissionAttribute Public Methods

The `OraclePermissionAttribute` public methods are listed in Table 6-117.

Table 6-117 OraclePermissionAttribute Public Methods

Public Methods	Description
CreatePermission	Returns a new `OraclePermissionAttribute` object that is configured based on the attributes set
`Equals`	Inherited from `System.Attribute`
`GetHashCode`	Inherited from `System.Attribute`
`GetType`	Inherited from `System.Attribute`
`IsDefaultAttribute`	Inherited from `System.Attribute`
`Match`	Inherited from `System.Attribute`
`ShouldSerializeConnectionString`	Inherited from `DBDataPermissionAttribute`
`ShouldSerializeKeyRestrictions`	Inherited from `DBDataPermissionAttribute`
`ToString`	Inherited from `System.Object`

6.19.2 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.

6.19.3 OraclePermissionAttribute Static Methods

The `OraclePermissionAttribute` static methods are listed in Table 6-118.

Table 6-118 OraclePermissionAttribute Static Methods

Static Methods	Description
`GetCustomAttribute`	Inherited from `System.Attribute` (Overloaded)
`GetCustomAttributes`	Inherited from `System.Attribute`(Overloaded)
`IsDefined`	Inherited from `System.Attribute`(Overloaded)
`ReferenceEquals`	Inherited from `System.Object`

6.19.4 OraclePermissionAttribute Public Properties

The `OraclePermissionAttribute` public properties are listed in Table 6-119.

Table 6-119 OraclePermissionAttribute Public Properties

Public Properties	Description
`Action`	Inherited from `SecurityAttribute`
`AllowBlankPassword`	Inherited from `DBDataPermissionAttribute`. `OraclePermissionAttribute` ignores this property. Any value set for this property, for an `OraclePermissionAttribute` object, is ignored.
`ConnectionString`	Inherited from `DBDataPermissionAttribute`
`KeyRestrictionBehavior`	Inherited from `DBDataPermissionAttribute`
`KeyRestrictions`	Inherited from `DBDataPermissionAttribute`
`TypeId`	Inherited from `System.Attribute`
`Unrestricted`	Inherited from `SecurityAttribute`

6.19.5 OraclePermissionAttribute Public Methods

The `OraclePermissionAttribute` public methods are listed in Table 6-120.

Table 6-120 OraclePermissionAttribute Public Methods

Public Methods	Description
CreatePermission	Returns a new `OraclePermissionAttribute` object that is configured based on the attributes set
`Equals`	Inherited from `System.Attribute`
`GetHashCode`	Inherited from `System.Attribute`
`GetType`	Inherited from `System.Attribute`
`IsDefaultAttribute`	Inherited from `System.Attribute`
`Match`	Inherited from `System.Attribute`
`ShouldSerializeConnectionString`	Inherited from `DBDataPermissionAttribute`
`ShouldSerializeKeyRestrictions`	Inherited from `DBDataPermissionAttribute`
`ToString`	Inherited from `System.Object`

6.19.5.1 CreatePermission

This method returns a new `OraclePermissionAttribute` object that is configured based on the attributes set.

Declaration

```
// C#
public override IPermission CreatePermission();
```

Return Value

An `OraclePermission` object.

6.20 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
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver
Assembly	`Oracle.DataAccess.dll`	`Oracle.ManagedDataAccess.dll`
Namespace	`Oracle.DataAccess.Client`	`Oracle.ManagedDataAccess.Client`
.NET Framework	3.5, 4.5, 4.6	4.5, 4.6

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".

6.20.1 OracleRowUpdatedEventArgs Members

`OracleRowUpdatedEventArgs` members are listed in the following tables.

OracleRowUpdatedEventArgs Constructors

`OracleRowUpdatedEventArgs` constructors are listed in Table 6-121.

Table 6-121 OracleRowUpdatedEventArgs Constructors

Constructor	Description
OracleRowUpdatedEventArgs Constructor	Instantiates a new instance of `OracleRowUpdatedEventArgs` class

OracleRowUpdatedEventArgs Static Methods

The `OracleRowUpdatedEventArgs` static method is listed in Table 6-122.

Table 6-122 OracleRowUpdatedEventArgs Static Method

Method	Description
`Equals`	Inherited from `System.Object` (Overloaded)

OracleRowUpdatedEventArgs Properties

The `OracleRowUpdatedEventArgs` properties are listed in Table 6-123.

Table 6-123 OracleRowUpdatedEventArgs Properties

Property	Description
Command	Specifies the `OracleCommand` that is used when `OracleDataAdapter.Update()` is called

Table 6-123 (Cont.) OracleRowUpdatedEventArgs Properties

Property	Description
`Errors`	Inherited from `System.Data.Common.RowUpdatedEventArgs`
`RecordsAffected`	Inherited from `System.Data.Common.RowUpdatedEventArgs`
`Row`	Inherited from `System.Data.Common.RowUpdatedEventArgs`
`StatementType`	Inherited from `System.Data.Common.RowUpdatedEventArgs`
`Status`	Inherited from `System.Data.Common.RowUpdatedEventArgs`
`TableMapping`	Inherited from `System.Data.Common.RowUpdatedEventArgs`

OracleRowUpdatedEventArgs Public Methods

The `OracleRowUpdatedEventArgs` properties are listed in Table 6-124.

Table 6-124 OracleRowUpdatedEventArgs Public Methods

Public Method	Description
`Equals`	Inherited from `System.Object` (Overloaded)
`GetHashCode`	Inherited from `System.Object`
`GetType`	Inherited from `System.Object`
`ToString`	Inherited from `System.Object`

6.20.2 OracleRowUpdatedEventArgs Constructor

The `OracleRowUpdatedEventArgs` constructor creates a new `OracleRowUpdatedEventArgs` instance.

Declaration

```
// C#
public OracleRowUpdatedEventArgs(DataRow row,IDbCommand command,
    StatementType statementType, DataTableMapping tableMapping);
```

Parameters

- *`row`*

  The `DataRow` sent for `Update`.

- *`command`*

  The `IDbCommand` executed during the `Update`.

- *`statementType`*

The `StatementType` Enumeration value indicating the type of SQL statement executed.

- *`tableMapping`*

  The `DataTableMapping` used for the `Update`.

6.20.3 OracleRowUpdatedEventArgs Static Methods

The `OracleRowUpdatedEventArgs` static method is listed in Table 6-125.

Table 6-125 OracleRowUpdatedEventArgs Static Method

Method	Description
`Equals`	Inherited from `System.Object` (Overloaded)

6.20.4 OracleRowUpdatedEventArgs Properties

The `OracleRowUpdatedEventArgs` properties are listed in Table 6-126.

Table 6-126 OracleRowUpdatedEventArgs Properties

Property	Description
Command	Specifies the `OracleCommand` that is used when `OracleDataAdapter.Update()` is called
`Errors`	Inherited from `System.Data.Common.RowUpdatedEventArgs`
`RecordsAffected`	Inherited from `System.Data.Common.RowUpdatedEventArgs`
`Row`	Inherited from `System.Data.Common.RowUpdatedEventArgs`
`StatementType`	Inherited from `System.Data.Common.RowUpdatedEventArgs`
`Status`	Inherited from `System.Data.Common.RowUpdatedEventArgs`
`TableMapping`	Inherited from `System.Data.Common.RowUpdatedEventArgs`

6.20.4.1 Command

This property specifies the `OracleCommand` that is used when `OracleDataAdapter.Update()` is called.

Declaration

```
// C#
public new OracleCommand Command {get;}
```

Property Value

The `OracleCommand` executed when `Update` is called.

6.20.5 OracleRowUpdatedEventArgs Public Methods

The `OracleRowUpdatedEventArgs` properties are listed in Table 6-127.

Table 6-127 OracleRowUpdatedEventArgs Public Methods

Public Method	Description
`Equals`	Inherited from `System.Object` (Overloaded)
`GetHashCode`	Inherited from System.`Object`
`GetType`	Inherited from `System.Object`
`ToString`	Inherited from `System.Object`

6.21 OracleRowUpdatedEventHandler Delegate

The `OracleRowUpdatedEventHandler` delegate represents the signature of the method that handles the `OracleDataAdapter.RowUpdated` event.

Declaration

```
// C#
public delegate void OracleRowUpdatedEventHandler(object sender,
   OracleRowUpdatedEventArgs eventArgs);
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver
Assembly	`Oracle.DataAccess.dll`	`Oracle.ManagedDataAccess.dll`
Namespace	`Oracle.DataAccess.Client`	`Oracle.ManagedDataAccess.Client`
.NET Framework	3.5, 4.5, 4.6	4.5, 4.6

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.

6.22 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
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver
Assembly	`Oracle.DataAccess.dll`	`Oracle.ManagedDataAccess.dll`
Namespace	`Oracle.DataAccess.Client`	`Oracle.ManagedDataAccess.Client`
.NET Framework	3.5, 4.5, 4.6	4.5, 4.6

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".

6.22.1 OracleRowUpdatingEventArgs Members

`OracleRowUpdatingEventArgs` members are listed in the following tables.

OracleRowUpdatingEventArgs Constructors

`OracleRowUpdatingEventArgs` constructors are listed in Table 6-128.

Table 6-128 OracleRowUpdatingEventArgs Constructors

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

OracleRowUpdatingEventArgs Static Methods

The `OracleRowUpdatingEventArgs` static methods are listed in Table 6-129.

Table 6-129 OracleRowUpdatingEventArgs Static Methods

Method	Description
`Equals`	Inherited from `System.Object` (Overloaded)

OracleRowUpdatingEventArgs Properties

The `OracleRowUpdatingEventArgs` properties are listed in Table 6-130.

Table 6-130 OracleRowUpdatingEventArgs Properties

Property	Description
Command	Specifies the `OracleCommand` that is used when the `OracleDataAdapter.Update()` is called
`Errors`	Inherited from `System.Data.Common.RowUpdatingEventArgs`
`Row`	Inherited from `System.Data.Common.RowUpdatingEventArgs`
`StatementType`	Inherited from `System.Data.Common.RowUpdatingEventArgs`
`Status`	Inherited from `System.Data.Common.RowUpdatingEventArgs`
`TableMapping`	Inherited from `System.Data.Common.RowUpdatingEventArgs`

OracleRowUpdatingEventArgs Public Methods

The `OracleRowUpdatingEventArgs` public methods are listed in Table 6-131.

Table 6-131 OracleRowUpdatingEventArgs Public Methods

Public Method	Description
`Equals`	Inherited from `System.Object` (Overloaded)
`GetHashCode`	Inherited from `System.Object`
`GetType`	Inherited from `System.Object`
`ToString`	Inherited from `System.Object`

6.22.2 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`.

6.22.3 OracleRowUpdatingEventArgs Static Methods

The `OracleRowUpdatingEventArgs` static method is listed in Table 6-132.

Table 6-132 OracleRowUpdatingEventArgs Static Method

Method	Description
`Equals`	Inherited from `System.Object` (Overloaded)

6.22.4 OracleRowUpdatingEventArgs Properties

The `OracleRowUpdatingEventArgs` properties are listed in Table 6-133.

Table 6-133 OracleRowUpdatingEventArgs Properties

Property	Description
Command	Specifies the `OracleCommand` that is used when the `OracleDataAdapter.Update()` is called
`Errors`	Inherited from `System.Data.Common.RowUpdatingEventArgs`
`Row`	Inherited from `System.Data.Common.RowUpdatingEventArgs`
`StatementType`	Inherited from `System.Data.Common.RowUpdatingEventArgs`
`Status`	Inherited from `System.Data.Common.RowUpdatingEventArgs`
`TableMapping`	Inherited from `System.Data.Common.RowUpdatingEventArgs`

6.22.4.1 Command

This property specifies the `OracleCommand` that is used when the `OracleDataAdapter.Update()` is called.

Declaration

```
// C#
public new OracleCommand Command {get; set;}
```

Property Value

The `OracleCommand` executed when `Update` is called.

6.22.5 OracleRowUpdatingEventArgs Public Methods

The `OracleRowUpdatingEventArgs` public methods are listed in Table 6-134.

Table 6-134 OracleRowUpdatingEventArgs Public Methods

Public Method	Description
`Equals`	Inherited from `System.Object` (Overloaded)
`GetHashCode`	Inherited from `System.Object`
`GetType`	Inherited from `System.Object`
`ToString`	Inherited from `System.Object`

See Also:

- "Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"
- OracleRowUpdatingEventArgs Class
- OracleRowUpdatingEventArgs Members

6.23 OracleRowUpdatingEventHandler Delegate

The `OracleRowUpdatingEventHandler` delegate represents the signature of the method that handles the `OracleDataAdapter.RowUpdating` event.

Declaration

```
// C#
public delegate void OracleRowUpdatingEventHandler (object sender,
    OracleRowUpdatingEventArgs eventArgs);
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver
Assembly	`Oracle.DataAccess.dll`	`Oracle.ManagedDataAccess.dll`
Namespace	`Oracle.DataAccess.Client`	`Oracle.ManagedDataAccess.Client`
.NET Framework	3.5, 4.5, 4.6	4.5, 4.6

Parameters

- *`sender`*

  The source of the event.

- *`eventArgs`*

  The `OracleRowUpdatingEventArgs` object that contains the event data.

Remarks

Event callbacks can be registered through this event delegate for applications that wish to be notified after a row is updated.

In the .NET framework, the convention of an event delegate requires two parameters: the object that raises the event and the event data.

See Also:

- "Oracle.DataAccess.Client and Oracle.ManagedDataAccess.Client Namespaces"
- "RowUpdating"

6.24 OracleShardingKey Class

An `OracleShardingKey` object can represent either a sharding key or a super sharding key.

Class Inheritance

```
System.Object

   Oracle.DataAccess.Client.OracleShardingKey
```

Declaration

```
// C#
public class OracleShardingKey : IDisposable
```

Requirements

Provider	ODP.NET, Unmanaged Driver
Assembly	`Oracle.DataAccess.dll`
Namespace	`Oracle.DataAccess.Client`
.NET Framework	3.5, 4.5, 4.6

Example

```
// C#

using System;
using Oracle.DataAccess.Client;

class Sharding
{
  static void Main()
  {
    OracleConnection con = new OracleConnection("user id=hr;password=hr;Data 
Source=orcl;");
    //Setting a shard key
    OracleShardingKey shardingKey = new OracleShardingKey(OracleDbType.Int32, 123);
    //Setting a second shard key value for a composite key
    shardingKey.SetShardingKey(OracleDbType.Varchar2, "gold");
    //Creating and setting the super shard key
    OracleShardingKey superShardingKey = new OracleShardingKey();
    superShardingKey.SetShardingKey(OracleDbType.Int32, 1000);

    //Setting super sharding key and sharding key on the connection
    con.SetShardingKey(shardingKey, superShardingKey);
    con.Open();

    //perform SQL query
  }
}
```

6.24.1 OracleShardingKey Members

`OracleShardingKey` members are listed in the following tables.

OracleShardingKey Constructors

`OracleShardingKey` constructors are listed in Table 6-135.

Table 6-135 OracleShardingKey Constructors

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

OracleShardingKey Instance Methods

`OracleShardingKey` instance methods are listed in Table 6-136.

Table 6-136 OracleShardingKey Instance Methods

Method	Description
SetShardingKey(OracleDbType, object)	Enables applications to set a key within the `OracleShardingKey` object
Dispose	Enables applications to explicitly dispose the `OracleShardingKey` object

6.24.2 OracleShardingKey Constructors

`OracleShardingKey` constructors instantiate new instances of the `OracleShardingKey` class.

Overload List:

- OracleShardingKey()

  This constructor instantiates a new instance of `OracleShardingKey` class.
- OracleShardingKey(OracleDbType, object)

  This constructor instantiates a new instance of the `OracleShardingKey` class using the supplied data type and key.

6.24.2.1 OracleShardingKey()

This constructor enables applications to construct the `OracleShardingKey` object.

Declaration

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

Exceptions

None

Remarks

This constructs an `OracleShardingKey` without any keys set.

6.24.2.2 OracleShardingKey(OracleDbType, object)

This constructor enables applications to construct the `OracleShardingKey` object with the supplied key.

Declaration

```
// C#
public OracleShardingKey(OracleDbType type, object key);
```

Exceptions

`InvalidArgumentException` – The supplied argument is invalid

Remarks

This constructs an `OracleShardingKey` with the supplied key set.

Acceptable `OracleDbType` enumeration values are `Byte`, `Decimal`, `Double`, `Int16`, `In32`, `Int64`, `Single`, `Varchar2`, `String`, `Date`, `TimeStamp`, and `Raw`.

6.24.3 OracleShardingKey Instance Methods

`OracleShardingKey` instance methods are listed in Table 6-137.

Table 6-137 OracleShardingKey Instance Methods

Instance Method	Description
SetShardingKey(OracleDbType, object)	Enables applications to set a key within the `OracleShardingKey` object
Dispose	Enables applications to explicitly dispose the `OracleShardingKey` object

6.24.3.1 SetShardingKey(OracleDbType, object)

This instance method enables applications to set a key within the `OracleShardingKey` object.

Declaration

```
// C#
public void SetShardingKey(OracleDbType type, object key);
```

Exceptions

`InvalidArgumentException` – The supplied argument is invalid

Remarks

This method sets a key within the `OracleShardingKey` object.

Acceptable `OracleDbType` enumeration values are `Byte`, `Decimal`, `Double`, `Int16`, `In32`, `Int64`, `Single`, `Varchar2`, `String`, `Date`, `TimeStamp`, and `Raw`.

This can be called multiple times to construct a composite key.

6.24.3.2 Dispose

This instance method enables applications to explicitly dispose the `OracleShardingKey` object.

Declaration

```
// C#
public void Dispose();
```

Exceptions

None

Remarks

This method disposes the `OracleShardingKey` object.

6.25 OracleTransaction Class

An `OracleTransaction` object represents a local transaction.

Class Inheritance

```
System.Object

  System.MarshalByRefObject

    System.Data.Common.DbTransaction

      Oracle.DataAccess.Client.OracleTransaction
```

Declaration

```
// C#
public sealed class OracleTransaction : DbTransaction

// C#
public sealed class OracleTransaction : MarshalByRefObject,
   IDbTransaction, IDisposable
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver
Assembly	`Oracle.DataAccess.dll`	`Oracle.ManagedDataAccess.dll`
Namespace	`Oracle.DataAccess.Client`	`Oracle.ManagedDataAccess.Client`
.NET Framework	3.5, 4.5, 4.6	4.5, 4.6

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();
  }
}
```

Not supported in a .NET stored procedure

6.25.1 OracleTransaction Members

`OracleTransaction` members are listed in the following tables.

OracleTransaction Static Methods

The `OracleTransaction` static method is listed in Table 6-138.

Table 6-138 OracleTransaction Static Method

Method	Description
`Equals`	Inherited from `System.Object` (Overloaded)

OracleTransaction Properties

`OracleTransaction` properties are listed in Table 6-139.

Table 6-139 OracleTransaction Properties

Property	Description
IsolationLevel	Specifies the isolation level for the transaction
Connection	Specifies the connection that is associated with the transaction

OracleTransaction Public Methods

`OracleTransaction` public methods are listed in Table 6-140.

Table 6-140   OracleTransaction Public Methods

Public Method	Description
Commit	Commits the database transaction
`CreateObjRef`	Inherited from `System.MarshalByRefObject`
Dispose	Frees the resources used by the `OracleTransaction` object
`Equals`	Inherited from `System.Object` (Overloaded)
`GetHashCode`	Inherited from `System.Object`
`GetLifetimeService`	Inherited from `System.MarshalByRefObject`
`GetType`	Inherited from `System.Object`
`InitializeLifetimeService`	Inherited from `System.MarshalByRefObject`
Rollback	Rolls back a database transaction (Overloaded)
Save	Creates a savepoint within the current transaction
`ToString`	Inherited from `System.Object`

6.25.2 OracleTransaction Static Methods

The `OracleTransaction` static method is listed in Table 6-141.

Table 6-141   OracleTransaction Static Method

Method	Description
`Equals`	Inherited from `System.Object` (Overloaded)

6.25.3 OracleTransaction Properties

`OracleTransaction` properties are listed in Table 6-142.

Table 6-142   OracleTransaction Properties

Property	Description
IsolationLevel	Specifies the isolation level for the transaction
Connection	Specifies the connection that is associated with the transaction

6.25.3.1 IsolationLevel

This property specifies the isolation level for the transaction.

Declaration

```
// C#
public override IsolationLevel IsolationLevel {get;}
```

Property Value

`IsolationLevel`

Implements

`IDbTransaction`

Exceptions

`InvalidOperationException` - The transaction has already completed.

Remarks

Default = `IsolationLevel.ReadCommitted`

6.25.3.2 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.

6.25.4 OracleTransaction Public Methods

`OracleTransaction` public methods are listed in Table 6-143.

Table 6-143 OracleTransaction Public Methods

Public Method	Description
Commit	Commits the database transaction
`CreateObjRef`	Inherited from `System.MarshalByRefObject`
Dispose	Frees the resources used by the `OracleTransaction` object
`Equals`	Inherited from `System.Object` (Overloaded)
`GetHashCode`	Inherited from `System.Object`
`GetLifetimeService`	Inherited from `System.MarshalByRefObject`

Table 6-143 (Cont.) OracleTransaction Public Methods

Public Method	Description
`GetType`	Inherited from `System.Object`
`InitializeLifetimeService`	Inherited from `System.MarshalByRefObject`
Rollback	Rolls back a database transaction (Overloaded)
Save	Creates a savepoint within the current transaction
`ToString`	Inherited from `System.Object`

6.25.4.1 Commit

This method commits the database transaction.

Declaration

```
// 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

```
// 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();
  }
}
```

6.25.4.2 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.

6.25.4.3 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.

6.25.4.4 Rollback()

This method rolls back a database transaction.

Declaration

```
// 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 rollsback the transaction, the number of rows remains the same

    string constr = "User Id=scott;Password=tiger;Data Source=oracle";
    OracleConnection con = new OracleConnection(constr);
    con.Open();

    OracleCommand cmd = con.CreateCommand();

    // Check the number of rows in MyTable before transaction
    cmd.CommandText = "SELECT COUNT(*) FROM MyTable";
    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();
  }
}
```

6.25.4.5 Rollback(string)

This method rolls back a database transaction to a savepoint within the current transaction.

Declaration

```
// 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.

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.

6.25.4.6 Save

This method creates a savepoint within the current transaction.

Declaration

```
// C#
public void Save(string savepointName);
```

Parameters

- *`savepointName`*

  The name of the savepoint being created in the current transaction.

Exceptions

`InvalidOperationException` - The transaction has already been completed.

Remarks

After creating a savepoint, the transaction does not enter a completed state and can be used for further operations.

The *savepointName* specified is created in the database in a case-insensitive manner. Calling the `Rollback` method rolls back to *savepointName*. This allows portions of a transaction to be rolled back, instead of the entire transaction.

Example

```
// Database Setup, 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();
    }
  }
```

6.26 OracleConnectionType Enumeration

`OracleConnectionType` enumerated 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 6-144 lists all the `OracleConnectionType` enumeration values with a description of each enumerated value.

Table 6-144 OracleConnectionType Enumeration Values

Member Name	Description
`Undefined`	No connection is associated with the `OracleConnection` object.
`Oracle`	The `OracleConnection` object is associated with an Oracle database.
`TimesTen`	The `OracleConnection` object is associated with a TimesTen database.

Requirements

Provider	ODP.NET, Unmanaged Driver
Assembly	`Oracle.DataAccess.dll`
Namespace	`Oracle.DataAccess.Client`
.NET Framework	3.5, 4.5, 4.6

6.27 OracleCollectionType Enumeration

`OracleCollectionType` enumerated values specify whether or not the `OracleParameter` object represents a collection, and if so, specifies the collection type.

Table 6-145 lists all the `OracleCollectionType` enumeration values with a description of each enumerated value.

Table 6-145 OracleCollectionType Enumeration Values

Member Name	Description
`None`	Is not a collection type
`PLSQLAssociativeArray`	Indicates that the collection type is a PL/SQL Associative Array (or PL/SQL Index-By Table)

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver
Assembly	`Oracle.DataAccess.dll`	`Oracle.ManagedDataAccess.dll`
Namespace	`Oracle.DataAccess.Client`	`Oracle.ManagedDataAccess.Client`
.NET Framework	3.5, 4.5, 4.6	4.5, 4.6

6.28 OracleDBShutdownMode Enumeration

`OracleDBShutdownMode` enumerated values specify the database shutdown options.

Table 6-147 lists all the `OracleDBShutdownMode` enumeration values with a description of each enumerated value.

Table 6-146 OracleDBShutdownMode Enumeration Values

Member Name	Description
`Default`	Refuses new connections and waits for existing connections to end.
`Transactional`	Refuses new connections and does not allow any new transactions. Waits for active transactions to commit.
`TransactionalLocal`	Refuses new connections and does not allow any new transactions. Waits for only local transactions to commit.
`Immediate`	Does not wait for current calls to complete or users to disconnect from the database. All uncommitted transactions are terminated and rolled back.
`Final`	Shuts down the database. Used in the second call for shutdown after the database has been closed and dismounted.
`Abort`	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.

Requirements

Provider	ODP.NET, Unmanaged Driver
Assembly	`Oracle.DataAccess.dll`

Provider	ODP.NET, Unmanaged Driver
Namespace	`Oracle.DataAccess.Client`
.NET Framework	3.5, 4.5, 4.6

6.29 OracleDBStartupMode Enumeration

`OracleDBStartupMode` enumerated values specify the database startup options.

Table 6-147 lists all the `OracleDBStartupMode` enumeration values with a description of each enumerated value.

Table 6-147 OracleDBStartupMode Enumeration Values

Member Name	Description
`NoRestriction`	Starts the database and allows access to all users.
`Restrict`	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.
`Force`	Shuts down a running instance in abort mode and starts a new instance.

Requirements

Provider	ODP.NET, Unmanaged Driver
Assembly	`Oracle.DataAccess.dll`
Namespace	`Oracle.DataAccess.Client`
.NET Framework	3.5, 4.5, 4.6

6.30 OracleDbType Enumeration

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

Table 6-148 lists all the `OracleDbType` enumeration values with a description of each enumerated value.

Table 6-148 OracleDbType Enumeration Values

Member Name	Description
`Array`	Oracle Collection (`VArray` or Nested Table) *Not Available in ODP.NET, Managed Driver*
`BFile`	Oracle `BFILE` type
`BinaryFloat`	Oracle `BINARY_FLOAT` type

Table 6-148 (Cont.) OracleDbType Enumeration Values

Member Name	Description
`BinaryDouble`	Oracle `BINARY_DOUBLE` type
`Blob`	Oracle `BLOB` type
`Boolean`	Oracle `BOOLEAN` type
`Byte`	`byte` type
`Char`	Oracle `CHAR` type
`Clob`	Oracle `CLOB` type
`Date`	Oracle `DATE` type
`Decimal`	Oracle `NUMBER` type
`Double`	8-byte `FLOAT` type
`Int16`	2-byte `INTEGER` type
`Int32`	4-byte `INTEGER` type
`Int64`	8-byte `INTEGER` type
`IntervalDS`	Oracle `INTERVAL DAY TO SECOND` type
`IntervalYM`	Oracle `INTERVAL YEAR TO MONTH` type
`Long`	Oracle `LONG` type
`LongRaw`	Oracle `LONG RAW` type
`NChar`	Oracle `NCHAR` type
`NClob`	Oracle `NCLOB` type
`NVarchar2`	Oracle `NVARCHAR2` type
`Object`	Oracle `Object` *Not Available in ODP.NET, Managed Driver*
`Raw`	Oracle `RAW` type
`Ref`	Oracle `REF` *Not Available in ODP.NET, Managed Driver*
`RefCursor`	Oracle `REF CURSOR` type
`Single`	4-byte `FLOAT` type, supports 6 precisions
`TimeStamp`	Oracle `TIMESTAMP` type
`TimeStampLTZ`	Oracle `TIMESTAMP WITH LOCAL TIME ZONE` type
`TimeStampTZ`	Oracle `TIMESTAMP WITH TIME ZONE` type
`Varchar2`	Oracle `VARCHAR2` type
`XmlType`	Oracle `XMLType` type

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver
Assembly	`Oracle.DataAccess.dll`	`Oracle.ManagedDataAccess.dll`
Namespace	`Oracle.DataAccess.Client`	`Oracle.ManagedDataAccess.Client`
.NET Framework	3.5, 4.5, 4.6	4.5, 4.6

6.31 OracleIdentityType Enumeration

The `OracleIdentityType` enumeration specifies how Oracle identity column values are generated.

Table 6-149 lists all the `OracleIdentityType` enumeration values with a description of each enumerated value.

Table 6-149 OracleIdentityType Members

Member Name	Description
`GeneratedAlways`	Indicates that unique values are generated for every insertion. No updates or inserts are allowed for this identity column.
`GeneratedByDefault`	Indicates that the values are generated only if no explicit value is provided for the identity column. Null values are not allowed for this identity column.
`GeneratedByDefaultOnNull`	Indicates that the values are generated only if no explicit value is provided or a `NULL` is inserted for the identity column.

Requirements

Provider	ODP.NET, Unmanaged Driver
Assembly	`Oracle.DataAccess.dll`
Namespace	`Oracle.DataAccess.Client`
.NET Framework	3.5, 4.5, 4.6

6.32 OracleParameterStatus Enumeration

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 6-150 lists all the `OracleParameterStatus` enumeration values with a description of each enumerated value.

Table 6-150 OracleParameterStatus Members

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

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver
Assembly	`Oracle.DataAccess.dll`	`Oracle.ManagedDataAccess.dll`
Namespace	`Oracle.DataAccess.Client`	`Oracle.ManagedDataAccess.Client`
.NET Framework	3.5, 4.5, 4.6	4.5, 4.6

7

Oracle Data Provider for .NET XML-Related Classes

This chapter describes ODP.NET XML-related classes and enumerations.

This chapter contains these topics:

All offsets are `0`-based for `OracleXmlStream` object parameters.

7.1 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 7-1 lists all the `OracleXmlCommandType` enumeration values with a description of each enumerated value.

Table 7-1 OracleXmlCommandType Members

Member Name	Description
`None`	No XML operation is desired
`Query`	The command text is a SQL query and the result of the query is an XML document. The SQL query needs to be a select statement
`Insert`	The command text is an XML document containing rows to insert.
`Update`	The command text is an XML document containing rows to update.
`Delete`	The command text is an XML document containing rows to delete.

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver
Assembly	`Oracle.DataAccess.dll`	`Oracle.ManagedDataAccess.dll`
Namespace	`Oracle.DataAccess.Client`	`Oracle.ManagedDataAccess.Client`
.NET Framework	3.5, 4.5, 4.6	4.5, 4.6

7.2 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
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver
Assembly	`Oracle.DataAccess.dll`	`Oracle.ManagedDataAccess.dll`
Namespace	`Oracle.DataAccess.Client`	`Oracle.ManagedDataAccess.Client`
.NET Framework	3.5, 4.5, 4.6	4.5, 4.6

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:template>\n" +
  "  <xsl:template match=\"ROWSET\">\n" +
  "      <xsl:apply-templates select=\"ROW\"/>\n" +
  "  </xsl:template>\n" +
  "  <xsl:template match=\"ROW\">\n" +
  "    <EMPLOYEE>\n" +
  "    <EMPLOYEE_ID>\n" +
  "      <xsl:apply-templates select=\"EMPNO\"/>\n" +
  "    </EMPLOYEE_ID>\n" +
  "    <EMPLOYEE_NAME>\n" +
  "      <xsl:apply-templates select=\"ENAME\"/>\n" +
  "    </EMPLOYEE_NAME>\n" +
  "    <HIRE_DATE>\n" +
  "      <xsl:apply-templates select=\"HIREDATE\"/>\n" +
  "    </HIRE_DATE>\n" +
  "    <JOB_TITLE>\n" +
  "      <xsl:apply-templates select=\"JOB\"/>\n" +
  "    </JOB_TITLE>\n" +
  "    </EMPLOYEE>\n" +
  "  </xsl:template>\n" +
  "</xsl:stylesheet>\n";

// 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");
Console.WriteLine("XML Document from OracleCommand.ExecuteStream():");
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)");
Console.WriteLine("XML Document from OracleCommand.ExecuteToStream():");
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();
      }
    }
```

7.2.1 OracleXmlQueryProperties Members

`OracleXmlQueryProperties` members are listed in the following tables.

OracleXmlQueryProperties Constructors

The `OracleXmlQueryProperties` constructors are listed in Table 7-2.

Table 7-2 OracleXmlQueryProperties Constructors

Constructor	Description
OracleXmlQueryProperties Constructor	Instantiates a new instance of the `OracleXmlQueryProperties` class

OracleXmlQueryProperties Properties

The `OracleXmlQueryProperties` properties are listed in Table 7-3.

Table 7-3 OracleXmlQueryProperties Properties

Name	Description
MaxRows	Specifies the maximum number of rows from the result set of the query that can be represented in the result XML document
RootTag	Specifies the root element of the result XML document
RowTag	Specifies the value of the XML element which identifies a row of data from the result set in an XML document
Xslt	Specifies the XSL document used for XML transformation using XSLT
XsltParams	Specifies parameters for the XSL document

OracleXmlQueryProperties Public Methods

The `OracleXmlQueryProperties` public methods are listed in Table 7-4.

Table 7-4 OracleXmlQueryProperties Public Methods

Name	Description
Clone	Creates a copy of an `OracleXmlQueryProperties` object

7.2.2 OracleXmlQueryProperties Constructor

The `OracleXmlQueryProperties` constructor instantiates a new instance of the `OracleXmlQueryProperties` class.

Declaration

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

7.2.3 OracleXmlQueryProperties Properties

The `OracleXmlQueryProperties` properties are listed in Table 7-5.

Table 7-5 OracleXmlQueryProperties Properties

Name	Description
MaxRows	Specifies the maximum number of rows from the result set of the query that can be represented in the result XML document
RootTag	Specifies the root element of the result XML document
RowTag	Specifies the value of the XML element which identifies a row of data from the result set in an XML document
Xslt	Specifies the XSL document used for XML transformation using XSLT
XsltParams	Specifies parameters for the XSL document

7.2.3.1 MaxRows

This property specifies the maximum number of rows from the result set of the query that can be represented in the result XML document.

Declaration

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

Property Value

The maximum number of rows.

Exceptions

`ArgumentException` - The new value for `MaxRows` is not valid.

Remarks

Default value is `-1`.

Possible values are:

- `-1` (all rows).
- A number greater than or equal to `0`.

7.2.3.2 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.

7.2.3.3 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.

7.2.3.4 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 value is `null`.

The XSL document is used for XML transformation of the XML document generated from the result set of the query.

7.2.3.5 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.

7.2.4 OracleXmlQueryProperties Public Methods

The `OracleXmlQueryProperties` public methods are listed in Table 7-6.

Table 7-6 OracleXmlQueryProperties Public Methods

Name	Description
Clone	Creates a copy of an `OracleXmlQueryProperties` object

7.2.4.1 Clone

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

Declaration

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

Return Value

An `OracleXmlQueryProperties` object

Implements

`ICloneable`

7.3 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
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver
Assembly	`Oracle.DataAccess.dll`	`Oracle.ManagedDataAccess.dll`
Namespace	`Oracle.DataAccess.Client`	`Oracle.ManagedDataAccess.Client`
.NET Framework	3.5, 4.5, 4.6	4.5, 4.6

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.
        cmd.CommandText = "<?xml version=\"1.0\"?>\n" +
          "<ROWSET>\n" +
          " <MYROW num = \"1\">\n" +
          " <EMPLOYEE_ID>1234</EMPLOYEE_ID>\n" +
          " </MYROW>\n" +
          " <MYROW num = \"2\">\n" +
          " <EMPLOYEE_ID>1235</EMPLOYEE_ID>\n" +
          " </MYROW>\n" +
          "</ROWSET>\n";

        // Set the XML save properties.
        KeyColumnsList = new string[1];
        KeyColumnsList[0] = "EMPLOYEE_ID";
        cmd.XmlSaveProperties.KeyColumnsList = KeyColumnsList;
        cmd.XmlSaveProperties.UpdateColumnsList = null;

        // Do the deletes.
        rows = cmd.ExecuteNonQuery();
        Console.WriteLine("rows: " + rows);

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

7.3.1 OracleXmlSaveProperties Members

`OracleXmlSaveProperties` members are listed in the following tables.

OracleXmlSaveProperties Constructor

`OracleXmlSaveProperties` constructors are listed in Table 7-7

Table 7-7 OracleXmlSaveProperties Constructor

Constructor	Description
OracleXmlSaveProperties Constructor	Instantiates a new instance of the `OracleXmlSaveProperties` class

OracleXmlSaveProperties Properties

The `OracleXmlSaveProperties` properties are listed in Table 7-8.

Table 7-8 OracleXmlSaveProperties Properties

Name	Description
KeyColumnsList	Specifies the list of columns used as a key to locate existing rows for update or delete using an XML document
RowTag	Specifies the value for the XML element that identifies a row of data in an XML document
Table	Specifies the name of the table or view to which changes are saved
UpdateColumnsList	Specifies the list of columns to update or insert
Xslt	Specifies the XSL document used for XML transformation using XSLT
XsltParams	Specifies the parameters for the XSLT document specified in the Xslt property

OracleXmlSaveProperties Public Methods

The `OracleXmlSaveProperties` public methods are listed in Table 7-9.

Table 7-9 OracleXmlSaveProperties Public Methods

Name	Description
Clone	Creates a copy of an `OracleXmlSaveProperties` object

7.3.2 OracleXmlSaveProperties Constructor

The `OracleXmlSaveProperties` constructor instantiates a new instance of `OracleXmlSaveProperties` class.

Declaration

```
// C#
public OracleXmlSaveProperties;
```

7.3.3 OracleXmlSaveProperties Properties

The `OracleXmlSaveProperties` properties are listed in Table 7-10.

Table 7-10 OracleXmlSaveProperties Properties

Name	Description
KeyColumnsList	Specifies the list of columns used as a key to locate existing rows for update or delete using an XML document
RowTag	Specifies the value for the XML element that identifies a row of data in an XML document
Table	Specifies the name of the table or view to which changes are saved
UpdateColumnsList	Specifies the list of columns to update or insert
Xslt	Specifies the XSL document used for XML transformation using XSLT
XsltParams	Specifies the parameters for the XSLT document specified in the Xslt property

7.3.3.1 KeyColumnsList

This property specifies the list of columns used as a key to locate existing rows for update or delete using an XML document.

Declaration

```
// C#
public string[] KeyColumnsList {get; set;}
```

Property Value

The list of columns.

Remarks

Default value is null.

The first null value (if any) terminates the list.

`KeyColumnsList` usage with `XMLCommandType` property values:

- `Insert` - `KeyColumnsList` is ignored and can be null.
- `Update` - `KeyColumnsList` must be specified; it identifies the columns to use to find the rows to be updated.
- `Delete` - If `KeyColumnsList` is null, all the column values in each row element in the XML document are used to locate the rows to delete. Otherwise, `KeyColumnsList` specifies the columns used to identify the rows to delete.

7.3.3.2 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.

7.3.3.3 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.

7.3.3.4 UpdateColumnsList

This property specifies the list of columns to update or insert.

Declaration

```
// C#
public string[] UpdateColumnsList {get; set;}
```

Property Value

A list of columns.

Remarks

Default value is null.

The first null value (if any) terminates the list.

`UpdateColumnList` usage with `XMLCommandType` property values:

- `Insert` - `UpdateColumnList` indicates which columns are assigned values when a new row is created. If `UpdateColumnList` is null, then all columns are assigned values. If a column is on the `UpdateColumnList`, but no value is specified for the row in the XML file, then `NULL` is used. If a column is not on the `UpdateColumnList`, then the default value for that column is used.
- `Update` - `UpdateColumnList` specifies columns to modify for each row of data in the XML document. If `UpdateColumnList` is null, all the values in each XML element in the XML document are used to modify the columns.
- `Delete` - `UpdateColumnsList` is ignored and can be null.

7.3.3.5 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.

7.3.3.6 XsltParams

This property specifies the parameters for the XSLT document specified in the `Xslt` property.

Declaration

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

Property Value

The parameters for the XSLT document.

Remarks

Default is `null`.

This property is a string delimited by semicolons in "`name=value`" pairs of the form "`param1=value1; param2=value2; …`".

7.3.4 OracleXmlSaveProperties Public Methods

The `OracleXmlSaveProperties` public methods are listed in Table 7-11.

Table 7-11 OracleXmlSaveProperties Public Methods

Name	Description
Clone	Creates a copy of an `OracleXmlSaveProperties` object

7.3.4.1 Clone

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

Declaration

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

Return Value

An `OracleXmlSaveProperties` object

Implements

`ICloneable`

7.4 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
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver
Assembly	`Oracle.DataAccess.dll`	`Oracle.ManagedDataAccess.dll`
Namespace	`Oracle.DataAccess.Types`	`Oracle.ManagedDataAccess.Client`
.NET Framework	3.5, 4.5, 4.6	4.5, 4.6

Thread Safety

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

7.4.1 OracleXmlStream Members

`OracleXmlStream` members are listed in the following tables.

OracleXmlStream Constructors

The `OracleXmlStream` constructors are listed in Table 7-12.

Table 7-12 OracleXmlStream Constructors

Constructor	Description
OracleXmlStream Constructor	Creates an instance of an `OracleXmlStream` object which provides a `Stream` representation of the XML data stored in an `OracleXmlType`

OracleXmlStream Static Methods

The `OracleXmlStream` static methods are listed in Table 7-13.

Table 7-13 OracleXmlStream Static Methods

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

OracleXmlStream Instance Properties

The `OracleXmlStream` instance properties are listed in Table 7-14.

Table 7-14 OracleXmlStream Instance Properties

Properties	Description
CanRead	Indicates whether or not the XML stream can be read
CanSeek	Indicates whether or not forward and backward seek operation can be performed
CanWrite	*Not Supported*
Connection	Indicates the `OracleConnection` that is used to retrieve the XML data
Length	Indicates the number of bytes in the XML stream
Position	Gets or sets the byte position within the stream
Value	Returns the XML data, starting from the first character in the stream as a string

OracleXmlStream Instance Methods

The `OracleXmlStream` instance methods are listed in Table 7-15.

Table 7-15 OracleXmlStream Instance Methods

Methods	Description
`BeginRead`	Inherited from `System.IO.Stream`
`BeginWrite`	Inherited from `System.IO.Stream`
Clone	Creates a copy of an `OracleXmlStream` object
Close	Closes the current stream and releases any resources associated with it
Dispose	Releases resources allocated by this object
`EndRead`	Inherited from `System.IO.Stream`
`EndWrite`	Inherited from `System.IO.Stream`
`Equals`	Inherited from `System.Object`
Flush	*Not Supported*
`GetHashCode`	Inherited from `System.Object`
`GetLifetimeService`	Inherited from `System.MarshalByRefObject`
`GetType`	Inherited from `System.Object`
`InitializeLifetimeService`	Inherited from `System.MarshalByRefObject`
Read	Reads a specified amount from the current stream instance and populates the array buffer (Overloaded)
`ReadByte`	Inherited from `System.IO.Stream`
Seek	Sets the position within the current stream and returns the new position within the current stream
SetLength	*Not Supported*
`ToString`	Inherited from `System.Object`
Write	*Not Supported*
`WriteByte`	*Not Supported*
WriteLine	*Not Supported*

7.4.2 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.

7.4.3 OracleXmlStream Static Methods

The `OracleXmlStream` static methods are listed in Table 7-16.

Table 7-16 OracleXmlStream Static Methods

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

7.4.4 OracleXmlStream Instance Properties

The `OracleXmlStream` instance properties are listed in Table 7-17.

Table 7-17 OracleXmlStream Instance Properties

Properties	Description
CanRead	Indicates whether or not the XML stream can be read
CanSeek	Indicates whether or not forward and backward seek operation can be performed
`CanWrite`	*Not Supported*
Connection	Indicates the `OracleConnection` that is used to retrieve the XML data
Length	Indicates the number of bytes in the XML stream
Position	Gets or sets the byte position within the stream
Value	Returns the XML data, starting from the first character in the stream as a string

7.4.4.1 CanRead

Overrides `Stream`

This property indicates whether or not the XML stream can be read.

Declaration

```
// C#
public override bool CanRead{get;}
```

Property Value

If the XML stream is can be read, returns `true`; otherwise, returns `false`.

7.4.4.2 CanSeek

Overrides `Stream`

This property indicates whether or not forward and backward seek operation can be performed.

Declaration

```
// C#
public override bool CanSeek{get;}
```

Property Value

If forward and backward seek operations can be performed, this property returns `true`. Otherwise, returns `false`.

7.4.4.3 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.

7.4.4.4 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.

7.4.4.5 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.

7.4.4.6 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.

7.4.5 OracleXmlStream Instance Methods

The `OracleXmlStream` instance methods are listed in Table 7-18.

Table 7-18 OracleXmlStream Instance Methods

Methods	Description
`BeginRead`	Inherited from `System.IO.Stream`
`BeginWrite`	Inherited from `System.IO.Stream`
Clone	Creates a copy of an `OracleXmlStream` object
Close	Closes the current stream and releases any resources associated with it
Dispose	Releases resources allocated by this object
`EndRead`	Inherited from `System.IO.Stream`
`EndWrite`	Inherited from `System.IO.Stream`
`Equals`	Inherited from `System.Object`
`Flush`	*Not Supported*
`GetHashCode`	Inherited from `System.Object`
`GetLifetimeService`	Inherited from `System.MarshalByRefObject`
`GetType`	Inherited from `System.Object`
`InitializeLifetimeService`	Inherited from `System.MarshalByRefObject`
Read	Reads a specified amount from the current XML stream instance and populates the array buffer (Overloaded)
`ReadByte`	Inherited from `System.IO.Stream`
Seek	Sets the position within the current stream and returns the new position within the current stream
`SetLength`	*Not Supported*
`ToString`	Inherited from `System.Object`
`Write`	*Not Supported*
`WriteByte`	*Not Supported*
`WriteLine`	*Not Supported*

7.4.5.1 Clone

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

Declaration

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

Return Value

An `OracleXmlStream` object.

Implements

`ICloneable`

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

Remarks

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

7.4.5.2 Close

Overrides `Stream`

This method closes the current stream and releases any resources associated with it.

Declaration

```
// C#
public override void Close();
```

7.4.5.3 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.

7.4.5.4 Flush

This method is not supported.

7.4.5.5 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.

7.4.5.6 Read(byte[ ], int, int)

Overrides `Stream`

This method reads a specified amount of unicode bytes from the current instance, advances the position within the stream, and populates the byte array buffer.

Declaration

```
// C#
public override int Read(byte[] buffer, int offset, int count);
```

Parameters

- *`buffer`*

  The byte array buffer that is populated.
- *`offset`*

  The zero-based offset (in bytes) at which the buffer is populated.
- *`count`*

  The maximum amount of bytes to be read.

Return Value

The number of unicode bytes read into the given `byte[]` buffer or `0` if the end of the stream has been reached.

Remarks

This method reads a maximum of *`count`* bytes from the current stream and stores them in buffer beginning at *`offset`*. The current position within the stream is advanced by the number of bytes read. However, if an exception occurs, the current position within the stream remains unchanged.

The XML data is read starting from the position specified by the `Position` property.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

7.4.5.7 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.

7.4.5.8 Seek

Overrides `Stream`.

This method sets the position within the current stream and returns the new position within the current stream.

Declaration

```
// C#
public long Seek(long offset, SeekOrigin origin);
```

Parameters

- *`offset`*

  A byte offset relative to origin.

  – If *`offset`* is negative, the new position precedes the position specified by *`origin`* by the number of bytes specified by *`offset`*.

  – If offset is zero, the new position is the position specified by *`origin`*.

  – If *`offset`* is positive, the new position follows the position specified by *`origin`* by the number of bytes specified by *`offset`*.

- *`origin`*

  A value of type `SeekOrigin` indicating the reference point used to obtain the new position.

Return Value

The new `Position` within the current stream.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object

Remarks

Use the `CanSeek` property to determine whether or not the current instance supports seeking. Seeking to any location beyond the length of the stream is supported.

7.4.5.9 SetLength

This method is not supported.

7.4.5.10 Write

This method is not supported.

7.4.5.11 WriteLine

This method is not supported.

7.5 OracleXmlType Class

An `OracleXmlType` object represents an Oracle `XMLType` instance.

Class Inheritance

`System.Object`

```
  System.OracleXmlType
```

Declaration

```
// C#
public sealed class OracleXmlType : IDisposable, INullable
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver
Assembly	`Oracle.DataAccess.dll`	`Oracle.ManagedDataAccess.dll`
Namespace	`Oracle.DataAccess.Types`	`Oracle.ManagedDataAccess.Client`
.NET Framework	3.5, 4.5, 4.6	4.5, 4.6

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.

7.5.1 OracleXmlType Members

`OracleXmlType` members are listed in the following tables.

OracleXmlType Constructors

The `OracleXmlType` constructors are listed in Table 7-19.

Table 7-19 OracleXmlType Constructors

Constructor	Description
OracleXmlType Constructors	Creates an instance of the `OracleXmlType` class (Overloaded)

OracleXmlType Static Methods

The `OracleXmlType` static methods are listed in Table 7-20.

Table 7-20 OracleXmlType Static Methods

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

OracleXmlType Static Fields

The `OracleXmlType` static field is listed in Table 7-21.

Table 7-21 OracleXmlType Static Field

Static Field	Description
Null	Represents a null value that can be assigned to an `OracleXmlType` instance

OracleXmlType Instance Properties

The `OracleXmlType` instance properties are listed in Table 7-22.

Table 7-22 OracleXmlType Instance Properties

Properties	Description
Connection	Indicates the `OracleConnection` that is used to retrieve and store XML data in the `OracleXmlType`
IsEmpty	Indicates whether or not the `OracleXmlType` is empty
IsFragment	Indicates whether the XML data is a collection of XML elements or a well-formed XML document
IsNull	Indicates whether or not the `OracleXmlType` is null
IsSchemaBased	Indicates whether or not the XML data represented by the `OracleXmlType` is based on an XML schema
RootElement	Represents the name of the top-level element of the schema-based XML data contained in the `OracleXmlType`
Schema	Represents the XML schema of the XML data contained in the `OracleXmlType`
SchemaUrl	Represents in the database for the XML schema of the XML data contained in the `OracleXmlType`.
Value	Returns the XML data starting from the first character in the current instance as a string

OracleXmlType Instance Methods

The `OracleXmlType` instance methods are listed in Table 7-23.

Table 7-23 OracleXmlType Instance Methods

Methods	Description
Clone	Creates a copy of the `OracleXmlType` instance
Dispose	Releases the resources allocated by this `OracleXmlType` object
`Equals`	Inherited from `System.Object`
Extract	Extracts a subset from the XML data using the given XPath expression (Overloaded)
`GetHashCode`	Inherited from `System.Object`
GetStream	Returns an instance of `OracleXmlStream` which provides a read-only stream of the XML data stored in this `OracleXmlType` instance

Table 7-23 (Cont.) OracleXmlType Instance Methods

Methods	Description
`GetType`	Inherited from `System.Object`
GetXmlDocument	Returns a `XmlDocument` object containing the XML data stored in this `OracleXmlType` instance
GetXmlReader	Returns a `XmlTextReader` object that can be used to manipulate XML data directly using the .NET Framework classes and methods
IsExists	Checks for the existence of a particular set of nodes identified by the given XPath expression in the XMLdata (Overloaded)
`ToString`	Inherited from `System.Object`
Transform	Transforms the `OracleXmlType` into another `OracleXmlType` instance using the given XSL document (Overloaded)
Update	Updates the XML node or fragment identified by the given XPath expression in the current `OracleXmlType` instance (Overloaded)
Validate	Validates whether or not the XML data in the `OracleXmlType` object conforms to the given XML schema.

7.5.2 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.

7.5.2.1 OracleXmlType(OracleClob)

This constructor creates an instance of the `OracleXmlType` class using the XML data contained in an `OracleClob` object.

Declaration

```
// C#
public OracleXmlType(OracleClob oraClob);
```

Parameters

- *`oraClob`*

  An `OracleClob` object.

Exceptions

`ArgumentNullException` - The `OracleClob` object is null.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

Remarks

The `CLOB` data depends on a valid connection object and the new `OracleXMLType` uses the `OracleConnection` in the `OracleClob` object to store data for the current instance.

7.5.2.2 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.

7.5.2.3 OracleXmlType(OracleConnection, XmlReader)

This constructor creates an instance of the `OracleXmlType` class using the contents of the .NET `XmlReader` object.

Declaration

```
// C#
public OracleXmlType(OracleConnection con, XmlReader reader);
```

Parameters

- *`con`*

  An `OracleConnection` object.
- *`reader`*

  An `XmlReader` object.

Exceptions

`ArgumentNullException` - The `OracleConnection` object is null.

`ArgumentException` - The *`reader`* argument contains no data.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

Remarks

The new `OracleXMLType` uses the given `OracleConnection` object to store data for the current instance.

7.5.2.4 OracleXmlType(OracleConnection, XmlDocument)

This constructor creates an instance of the `OracleXmlType` object using the contents of the XML DOM document in the .NET `XmlDocument` object.

Declaration

```
// C#
public OracleXmlType(OracleConnection con, XmlDocument domDoc);
```

Parameters

- *`con`*

  An `OracleConnection` object.
- *`domDoc`*

  An XML document.

Exceptions

`ArgumentNullException` - The `OracleConnection` object is null.

`ArgumentException` - The *`domDoc`* argument contains no data.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

Remarks

The new `OracleXMLType` uses the given `OracleConnection` object to store data for the current instance.

7.5.3 OracleXmlType Static Methods

The `OracleXmlType` static methods are listed in Table 7-24.

Table 7-24 OracleXmlType Static Methods

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

7.5.4 OracleXmlType Static Fields

The `OracleXmlType` static field is listed in Table 7-25.

Table 7-25 OracleXmlType Static Field

Static Field	Description
Null	Represents a null value that can be assigned to an `OracleXmlType` instance

7.5.4.1 Null

This static field represents a null value that can be assigned to an `OracleXmlType` instance.

Declaration

```
// C#
public static readonly OracleXmlType Null;
```

7.5.5 OracleXmlType Instance Properties

The `OracleXmlType` instance properties are listed in Table 7-26.

Table 7-26 OracleXmlType Instance Properties

Properties	Description
Connection	Indicates the `OracleConnection` that is used to retrieve and store XML data in the `OracleXmlType`
IsEmpty	Indicates whether or not the `OracleXmlType` is empty

Table 7-26 (Cont.) OracleXmlType Instance Properties

Properties	Description
IsFragment	Indicates whether the XML data is a collection of XML elements or a well-formed XML document
IsNull	Indicates whether or not the `OracleXmlType` is null
IsSchemaBased	Indicates whether or not the XML data represented by the `OracleXmlType` is based on an XML schema
Null	Represents a null value that can be assigned to an `OracleXmlType` instance
RootElement	Represents the name of the top-level element of the schema-based XML data contained in the `OracleXmlType`
Schema	Represents the XML schema of the XML data contained in the `OracleXmlType`
SchemaUrl	Represents URL in the database for the XML schema of the XML data contained in the `OracleXmlType`
Value	Returns the XML data starting from the first character in the current instance as a string

7.5.5.1 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`.

7.5.5.2 IsEmpty

This property indicates whether or not the `OracleXmlType` is empty.

Declaration

```
// C#
public bool IsEmpty {get;}
```

Property Value

Returns `true` if the `OracleXmlType` represents an empty XML document. Returns `false` otherwise.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

7.5.5.3 IsFragment

This property indicates whether the XML data is a collection of XML elements or a well-formed XML document.

Declaration

```
// C#
public bool IsFragment {get;}
```

Property Value

Returns `true` if the XML data contained in the `OracleXmlType` object is a collection of XML elements with no root element. Returns `false` otherwise.

Exceptions

`ObjectDisposedException` - The object is already disposed.

7.5.5.4 IsNull

This property indicates whether or not the `OracleXmlType` is null.

Declaration

```
// C#
public bool IsNull {get;}
```

Property Value

Returns `true` if the `OracleXmlType` represents a null value. 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.

7.5.5.5 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.

7.5.5.6 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.

7.5.5.7 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.

7.5.5.8 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.

7.5.5.9 Value

This property returns the XML data starting from the first character in the current instance as a `string`.

Declaration

```
// C#
public string Value{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.

7.5.6 OracleXmlType Instance Methods

The `OracleXmlType` instance methods are listed in Table 7-27.

Table 7-27 OracleXmlType Instance Methods

Methods	Description
Clone	Creates a copy of the `OracleXmlType` instance
Dispose	Releases the resources allocated by this `OracleXmlType` object
`Equals`	Inherited from `System.Object`
Extract	Extracts a subset from the XML data using the given XPath expression (Overloaded)
`GetHashCode`	Inherited from `System.Object`
GetStream	Returns an instance of `OracleXmlStream` which provides a read-only stream of the XML data stored in this `OracleXmlType` instance
`GetType`	Inherited from `System.Object`
GetXmlDocument	Returns a `XmlDocument` object containing the XML data stored in this `OracleXmlType` instance
GetXmlReader	Returns a `XmlTextReader` object that can be used to manipulate XML data directly using the .NET Framework classes and methods
IsExists	Checks for the existence of a particular set of nodes identified by the given XPath expression in the XMLdata (Overloaded)
`ToString`	Inherited from `System.Object`
Transform	Transforms the `OracleXmlType` into another `OracleXmlType` instance using the given XSL document (Overloaded)
Update	Updates the XML node or fragment identified by the given XPath expression in the current `OracleXmlType` instance (Overloaded)
Validate	Validates whether or not the XML data in the `OracleXmlType` object conforms to the given XML schema.

7.5.6.1 Clone

This method creates a copy of this `OracleXmlType` instance.

Declaration

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

Implements

`ICloneable`

Return Value

An `OracleXmlType` object.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

7.5.6.2 Dispose

This method releases the resources allocated by this object.

Declaration

```
// C#
public void Dispose();
```

Implements

`IDisposable`

7.5.6.3 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.

7.5.6.4 Extract(string, string)

This method extracts a subset from the XML data represented by the `OracleXmlType` object using the given XPath expression and a string parameter for namespace resolution.

Declaration

```
// C#
public OracleXmlType Extract(string xpathExpr, string nsMap);
```

Parameters

- *`xpathExpr`*

  The XPath expression.
- *`nsMap`*

  The string parameter used for namespace resolution of the XPath expression. *`nsMap`* has zero or more namespaces separated by spaces. *`nsMap`* can be null. For example:

  ```
  xmlns:nsi"=http://www.company1.com" xmlns:nsz="http://www.company2.com"
  ```

Return Value

An `OracleXmlType` object.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`ArgumentNullException` - The *`xpathExpr`* is null or zero-length.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

7.5.6.5 Extract(string, XmlNameSpaceManager)

This public method extracts a subset from the XML data represented by the `OracleXmlType` object, using the given XPath expression and a .NET `XmlNameSpaceManager` object for namespace resolution.

Declaration

```
// C#
public OracleXmlType Extract(string xpathExpr, XmlNameSpaceManager nsMgr);
```

Parameters

- *`xpathExpr`*

  The XPath expression.

- *`nsMgr`*

  The .NET `XmlNameSpaceManager` object used for namespace resolution of the XPath expression. *`nsMgr`* can be null.

Return Value

An `OracleXmlType`.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`ArgumentNullException` - The *`xpathExpr`* is null or zero-length.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

Remarks

The default namespace is ignored if its value is an empty string.

7.5.6.6 GetStream

This public method returns an instance of `OracleXmlStream` which provides a read-only stream of the XML data stored in this `OracleXmlType` instance.

Declaration

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

Return Value

A `Stream` object.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

7.5.6.7 GetXmlDocument

This public method returns a `XmlDocument` object containing the XML data stored in this `OracleXmlType` instance.

Declaration

```
// C#
public XmlDocument GetXmlDocument();
```

Return Value

An `XmlDocument` object.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

Remarks

The XML data in the `XmlDocument` object is a copy of the XML data in the `OracleXmlType` instance and modifying it does not automatically modify the XML data in the `OracleXmlType` instance. The `XmlDocument` instance returned has the `PreserveWhitespace` property set to `true`.

7.5.6.8 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.

7.5.6.9 IsExists

`IsExists` checks for the existence of a particular set of nodes identified by the XPath expression in the XML data.

Overload List:

- IsExists(string, string)

  This method checks for the existence of a particular set of nodes identified by the XPath expression in the XML data represented by the current `OracleXmlType` instance using a string parameter for namespace resolution.
- IsExists(string, XmlNameSpaceManager)

  This method checks for the existence of a particular set of nodes identified by the XPath expression in the XML document represented by the current `OracleXmlType` instance using a .NET `XmlNameSpaceManager` object for namespace resolution.

7.5.6.10 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.

7.5.6.11 IsExists(string, XmlNameSpaceManager)

This method checks the existence of a particular set of nodes identified by the XPath expression in the XML document represented by the current `OracleXmlType` instance using a .NET `XmlNameSpaceManager` object for namespace resolution.

Declaration

```
// C#
public bool IsExists(string xpathExpr, XmlNameSpaceManager nsMgr);
```

Parameters

- *`xpathExpr`*

  The XPath expression.
- *`nsMgr`*

  The .NET `XmlNameSpaceManager` object used for namespace resolution of the XPath expression. *`nsMgr`* can be null.

Return Value

Returns `true` if the required set of nodes exists; otherwise, returns `false`.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`ArgumentNullException` - The `xpathExpr` is null or zero-length.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

Remarks

The default namespace is ignored if its value is an empty string.

7.5.6.12 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.

7.5.6.13 Transform(OracleXmlType, string)

This method transforms the current `OracleXmlType` instance into another `OracleXmlType` instance using the given XSL document and a string of XSLT parameters.

Declaration

```
// C#
public OracleXmlType Transform(OracleXmlType xsldoc, string paramMap);
```

Parameters

- *`xsldoc`*

  The XSL document as an `OracleXmlType` object.
- *`paramMap`*

  A string which provides the parameters for the XSL document.

  For this release, *`paramMap`* is ignored.

Return Value

An `OracleXmlType` object containing the transformed XML document.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`ArgumentNullException` - The *`xsldoc`* parameter is null.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

7.5.6.14 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.

7.5.6.15 Update

This method updates the XML node or fragment identified by the given XPath expression in the current `OracleXmlType` instance.

Overload List:

- Update(string, string, string)

  This method updates the XML nodes identified by the given XPath expression with the given string value and a string parameter for namespace resolution.
- Update(string, XmlNameSpaceManager, string)

  This method updates the XML nodes identified by the given XPath expression with the given string value and a .NET `XmlNameSpaceManager` object for namespace resolution.
- Update(string, string, OracleXmlType)

  This method updates the XML nodes identified by the given XPath expression with the XML data stored in the given `OracleXmlType` value and a string parameter for namespace resolution.
- Update(string, XmlNameSpaceManager, OracleXmlType)

  This method updates the XML nodes identified by the given XPath expression with the XML data stored in the given `OracleXmlType` value and a .NET `XmlNameSpaceManager` object for namespace resolution.

7.5.6.16 Update(string, string, string)

This method updates the XML nodes identified by the given XPath expression with the given string value and a string parameter for namespace resolution.

Declaration

```
// C#
public void Update(string xpathExpr, string nsMap, string value);
```

Parameters

- *xpathExpr*

  The XPath expression that identifies the nodes to update.
- *nsMap*

  The string parameter used for namespace resolution of the XPath expression. *nsMap* has zero or more namespaces separated by spaces. *nsMap* can be null. For example:

  ```
  xmlns:nsi"=http://www.company1.com" xmlns:nsz="http://www.company2.com"
  ```
- *value*

  The new value as a `string`.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`ArgumentNullException` - The *xpathExpr* is null or zero-length.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

Remarks

The default namespace is ignored if its value is an empty string.

7.5.6.17 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.

7.5.6.18 Update(string, string, OracleXmlType)

This method updates the XML nodes identified by the given XPath expression with the XML data stored in the given `OracleXmlType` value and a string parameter for namespace resolution.

Declaration

```
// C#
public void Update(string xpathExpr, string nsMap, OracleXmlType value);
```

Parameters

- *`xpathExpr`*

  The XPath expression that identifies the nodes to update.
- *`nsMap`*

  The string parameter used for namespace resolution of the XPath expression. *`nsMap`* has zero or more namespaces separated by spaces. *`nsMap`* can be null.
- *`value`*

  The new value as an `OracleXmlType` object.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`ArgumentNullException` - The *`xpathExpr`* is null or zero-length.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

Remarks

The default namespace is ignored if its value is an empty string.

7.5.6.19 Update(string, XmlNameSpaceManager, OracleXmlType)

This method updates the XML nodes identified by the given XPath expression with the XML data stored in the given `OracleXmlType` value and a .NET `XmlNameSpaceManager` object for namespace resolution.

Declaration

```
// C#
public void Update(string xpathExpr, XmlNameSpaceManager nsMgr, OracleXmlType
value);
```

Parameters

- *xpathExpr*

  The XPath expression that identifies the nodes to update.
- *nsMgr*

  The .NET `XmlNameSpaceManager` object used for namespace resolution of the XPath expression. *nsMgr* can be null.
- *value*

  The new value as an `OracleXmlType` object.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`ArgumentNullException` - The *xpathExpr* is null or zero-length.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

Remarks

The default namespace is ignored if its value is an empty string.

7.5.6.20 Validate

This methods validates whether or not the XML data in the `OracleXmlType` object conforms to the given XML schema.

Declaration

```
// C#
public bool Validate(String schemaUrl);
```

Parameters

- `schemaUrl`

  A string representing the URL in the database of the XML schema.

Return Value

Returns true if the XML data conforms to the XML schema; otherwise, returns false.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

`ArgumentNullException` - The *`schemaUrl`* argument is null or an empty string.

8

Oracle Data Provider for .NET HA Event Classes

This chapter describes the following ODP.NET HA event class and enumerations:

- OracleHAEventArgs Class
- OracleHAEventHandler Delegate
- OracleHAEventSource Enumeration
- OracleHAEventStatus Enumeration

8.1 OracleHAEventArgs Class

The `OracleHAEventArgs` class provides event data for the `OracleConnection.HAEvent` event.

Class Inheritance

```
  System.Object

    System.EventArgs

      Oracle.DataAccess.Client.OracleHAEventArgs
```

Declaration

```
// C#
public sealed class OracleHAEventArgs
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver
Assembly	`Oracle.DataAccess.dll`	`Oracle.ManagedDataAccess.dll`
Namespace	`Oracle.DataAccess.Client`	`Oracle.ManagedDataAccess.Client`
.NET Framework	3.5, 4.5, 4.6	4.5, 4.6

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.

8.1.1 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

Name	Description
DatabaseDomainName	Specifies the domain name of the database affected by the `HAevent`
DatabaseName	Specifies the database affected by the `HAevent`
HostName	Specifies the host that triggered the event
InstanceName	Specifies the instance that triggered the event
Reason	Specifies the reason for which the HA event was sent by the server
ServiceName	Specifies the service that triggered the event
Source	Specifies the source that triggered the event
Status	Specifies the status of the source that triggered the event
Time	Specifies the time when the event was triggered on the server

8.1.2 OracleHAEventArgs Properties

The `OracleHAEventArgs` properties are listed in Table 8-2.

Table 8-2 OracleHAEventArgs Properties

Name	Description
DatabaseDomainName	Specifies the domain name of the database affected by the `HAevent`
DatabaseName	Specifies the database affected by the HAevent
HostName	Specifies the host that triggered the event
InstanceName	Specifies the instance that triggered the event
Reason	Specifies the reason for which the HA event was sent by the server
ServiceName	Specifies the service that triggered the event
Source	Specifies the source that triggered the event
Status	Specifies the status of the source that triggered the event
Time	Specifies the time when the event was triggered on the server

8.1.2.1 DatabaseDomainName

This property specifies the domain name of the database that is affected by the HA event.

Declaration

```
// C#
public string DatabaseDomainName {get;}
```

Property Value

The domain name of the database that is affected by the HA Event.

8.1.2.2 DatabaseName

This property specifies the database that is affected by the HA event.

Declaration

```
// C#
public string DatabaseName {get;}
```

Property Value

This property specifies the database name that is affected by the HA event.

8.1.2.3 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.

8.1.2.4 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.

8.1.2.5 Reason

This property specifies reason for which the HA event was sent by the server.

Declaration

```
// C#
public string Reason {get;}
```

Property Value

The reason the HA Event was triggered. Possible values include `Data_Guard_Failover`, `Failure`, `Dependency`, `User`, `Autostart`, and `Restart`.

The value `User` is indicative of a planned outage. All other values are indicative of an unplanned outage.

8.1.2.6 ServiceName

This property specifies the service that triggered the HA event.

Declaration

```
// C#
public string ServiceName {get;}
```

Property Value

The service that is affected by the HA Event.

8.1.2.7 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.

8.1.2.8 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.

8.1.2.9 Time

This property specifies the time when the HA event was triggered on the server.

Declaration

```
// C#
public DateTime Time {get;}
```

Property Value

The time that the HA Event was triggered.

8.2 OracleHAEventHandler Delegate

The `OracleHAEventHandler` delegate represents the signature of the method that handles the `OracleConnection.HAEvent` event.

Declaration

```
// C#
public delegate void OracleHAEventHandler(OracleHAEventArgs eventArgs);
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver
Assembly	`Oracle.DataAccess.dll`	`Oracle.ManagedDataAccess.dll`
Namespace	`Oracle.DataAccess.Client`	`Oracle.ManagedDataAccess.Client`
.NET Framework	3.5, 4.5, 4.6	4.5, 4.6

Parameters

- *`sender`*

  The source of the event.

- *`EventArgs`*

  The `OracleHAEventArgs` object that contains the event data.

8.3 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

Member Name	Description
`Service`	The source of the HA Event is a service.

Table 8-3 (Cont.) OracleHAEventSource Enumeration Member Values

Member Name	Description
`ServiceMember`	The source of the HA Event is a service member.
`Database`	The source of the HA Event is a database.
`Host`	The source of the HA Event is a host.
`Instance`	The source of the HA Event is an instance.
`Node`	The source of the HA Event is a node.

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver
Assembly	`Oracle.DataAccess.dll`	`Oracle.ManagedDataAccess.dll`
Namespace	`Oracle.DataAccess.Client`	`Oracle.ManagedDataAccess.Client`
.NET Framework	3.5, 4.5, 4.6	4.5, 4.6

8.4 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

Member Name	Description
`Up`	The source of the HA Event is up.
`Down`	The source of the HA Event is down.

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver
Assembly	`Oracle.DataAccess.dll`	`Oracle.ManagedDataAccess.dll`
Namespace	`Oracle.DataAccess.Client`	`Oracle.ManagedDataAccess.Client`
.NET Framework	3.5, 4.5, 4.6	4.5, 4.6

9

Continuous Query Notification Classes

This chapter describes Oracle Data Provider for .NET Continuous Query Notification Classes, Event Delegates, and Enumerations.

See Also:

"Continuous Query Notification Support "

This chapter contains these topics:

9.1 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

  Oracle.DataAccess.Client.OracleDependency
```

Declaration

```
// C#
public sealed class OracleDependency
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver
Assembly	`Oracle.DataAccess.dll`	`Oracle.ManagedDataAccess.dll`
Namespace	`Oracle.DataAccess.Client`	`Oracle.ManagedDataAccess.Client`

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver
.NET Framework	3.5, 4.5, 4.6	4.5, 4.6

Not supported in a .NET stored procedure

Thread Safety

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

9.1.1 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

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

OracleDependency Static Fields

The `OracleDependency` static field is listed in Table 9-2.

Table 9-2 OracleDependency Static Field

Static Field	Description
Port	Indicates the port number that the notification listener listens on, for database notifications

OracleDependency Static Methods

`OracleDependency` static methods are listed in Table 9-3.

Table 9-3 OracleDependency Static Methods

Static Methods	Description
`Equals`	Inherited from `System.Object`
GetOracleDependency	Returns an `OracleDependency` instance based on the specified unique identifier

OracleDependency Properties

`OracleDependency` properties are listed in Table 9-4.

Table 9-4 OracleDependency Properties

Properties	Description
DataSource	Indicates the data source associated with the `OracleDependency` instance
HasChanges	Indicates whether or not there is any change in the database associated with this dependency
Id	Represents the unique identifier for the `OracleDependency` instance
IsEnabled	Specifies whether or not the dependency is enabled between the application and the database
QueryBasedNotification	Specifies whether the change notification registration is object-based or query-based
RegisteredQueryIDs	Provides a list of `CHANGE_NOTIFICATION_QUERY_IDs`
RegisteredResources	Indicates the database resources that are registered in the notification registration
RowidInfo	Specifies whether or not `ROWID` information is part of change notification events fired whenever data changes on the database
UserName	Indicates the database user name associated with the `OracleDependency` instance

OracleDependency Methods

`OracleDependency` methods are listed in Table 9-5.

Table 9-5 OracleDependency Methods

Methods	Description
AddCommandDependency	Binds the `OracleDependency` instance to the specified `OracleCommand` instance
`Equals`	Inherited from `System.Object`
`GetHashCode`	Inherited from `System.Object`
`GetType`	Inherited from `System.Object`
RemoveRegistration	Removes the specified dependency between the application and the database
`ToString`	Inherited from `System.Object`

OracleDependency Events

The `OracleDependency` event is listed in Table 9-6.

Table 9-6 OracleDependency Events

Event	Description
OnChange	An event that is sent when a database notification associated with the dependency is received from the database

9.1.2 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.

9.1.2.1 OracleDependency ( )

This constructor creates an instance of the `OracleDependency` class.

Declaration

```
// 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.

9.1.2.2 OracleDependency(OracleCommand)

This constructor creates an instance of the `OracleDependency` class and binds it to an `OracleCommand` instance.

Declaration

```
// C#
public OracleDependency (OracleCommand cmd)
```

Parameters

- *`cmd`*

  The command that the `OracleDependecy` 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.

9.1.2.3 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

```
// 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.

9.1.3 OracleDependency Static Fields

The `OracleDependency` static field is listed in Table 9-7.

Table 9-7 OracleDependency Static Field

Static Field	Description
Port	Indicates the port number that the notification listener listens on, for database notifications

9.1.3.1 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.

9.1.4 OracleDependency Static Methods

`OracleDependency` static methods are listed in Table 9-8.

Table 9-8 OracleDependency Static Methods

Static Methods	Description
`Equals`	Inherited from `System.Object`
GetOracleDependency	Returns an `OracleDependency` instance based on the specified unique identifier

9.1.4.1 GetOracleDependency

This static method returns an `OracleDependency` instance based on the specified unique identifier.

Declaration

```
// 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.

9.1.5 OracleDependency Properties

`OracleDependency` properties are listed in Table 9-9.

Table 9-9 OracleDependency Properties

Properties	Description
DataSource	Indicates the data source associated with the `OracleDependency` instance
HasChanges	Indicates whether or not there is any change in the database associated with this dependency
Id	Represents the unique identifier for the `OracleDependency` instance
IsEnabled	Specifies whether or not the dependency is enabled between the application and the database
QueryBasedNotification	Specifies whether the change notification registration is object-based or query-based
RegisteredQueryIDs	Provides a list of `CHANGE_NOTIFICATION_QUERY_IDs`

Table 9-9 (Cont.) OracleDependency Properties

Properties	Description
RegisteredResources	Indicates the database resources that are registered in the notification registration
RowidInfo	Specifies whether or not `ROWID` information is part of change notification events fired whenever data changes on the database
UserName	Indicates the database user name associated with the `OracleDependency` instance

9.1.5.1 DataSource

This property indicates the data source associated with the `OracleDependency` instance.

Declaration

```
// 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.

9.1.5.2 HasChanges

This property indicates whether or not there is any change in the database associated with this dependency.

Declaration

```
// 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.

9.1.5.3 Id

This property represents the unique identifier for the `OracleDependency` instance.

Declaration

```
// 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.

9.1.5.4 IsEnabled

This property specifies whether or not the dependency is enabled between the application and the database.

Declaration

```
// 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.

9.1.5.5 QueryBasedNotification

This instance property specifies whether the change notification registration is object-based or query-based.

Declaration

```
// 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 11*g* 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.

9.1.5.6 RegisteredQueryIDs

This instance property provides a list of `CHANGE_NOTIFICATION_QUERY_ID`s.

Declaration

```
// C#
public ArrayList RegisteredQueryIDs {get;}
```

Property Value

This property is an `ArrayList` of `CHANGE_NOTIFICATION_QUERY_ID`s.

Exceptions

`ObjectDisposedException` - The object is already disposed.

Remarks

This property provides a list of `CHANGE_NOTIFICATION_QUERY_ID`s 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*.

9.1.5.7 RegisteredResources

This property indicates the database resources that are registered in the notification registration.

Declaration

```
// C#
public ArrayList RegisteredResources{get;}
```

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.

9.1.5.8 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`.

9.1.5.9 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.

9.1.6 OracleDependency Methods

`OracleDependency` methods are listed in Table 9-10.

Table 9-10 OracleDependency Methods

Methods	Description
AddCommandDependency	Binds the `OracleDependency` instance to the specified `OracleCommand` instance
`Equals`	Inherited from `System.Object`
`GetHashCode`	Inherited from `System.Object`
`GetType`	Inherited from `System.Object`
RemoveRegistration	Removes the specified dependency between the application and the database
`ToString`	Inherited from `System.Object`

9.1.6.1 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:

- `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.

9.1.6.2 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.

9.1.7 OracleDependency Events

The `OracleDependency` event is listed in Table 9-11.

Table 9-11 OracleDependency Event

Event	Description
OnChange	An event that is sent when a database notification associated with the dependency is received from the database

9.1.7.1 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.

9.2 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`

```
  Oracle.DataAccess.Client.OracleNotificationRequest
```

Declaration

```
// C#
public sealed class OracleNotificationRequest
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver
Assembly	`Oracle.DataAccess.dll`	`Oracle.ManagedDataAccess.dll`
Namespace	`Oracle.DataAccess.Client`	`Oracle.ManagedDataAccess.Client`
.NET Framework	3.5, 4.5, 4.6	4.5, 4.6

Not supported in a .NET stored procedure

Thread Safety

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

9.2.1 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

Static Method	Description
`Equals`	Inherited from `System.Object`

OracleNotificationRequest Properties

`OracleNotificationRequest` properties are listed in Table 9-13.

Table 9-13 OracleNotificationRequest Properties

Properties	Description
IsNotifiedOnce	Indicates whether or not the registration is to be removed upon notification
IsPersistent	Indicates whether or not the notification message should be queued persistently in the database before delivery
Timeout	Specifies the time that the registration remains alive
GroupingNotificationEnabled	Specifies whether grouping notification is enabled or not
GroupingType	Specifies the type of grouping notification
GroupingInterval	Specifies the interval between grouping notifications, in seconds

OracleNotificationRequest Methods

`OracleNotificationRequest` methods are listed in Table 9-14.

Table 9-14 OracleNotificationRequest Methods

Methods	Description
`Equals`	Inherited from `System.Object`
`GetHashCode`	Inherited from `System.Object`
`GetType`	Inherited from `System.Object`
`ToString`	Inherited from `System.Object`

9.2.2 OracleNotificationRequest Static Methods

The `OracleNotificationRequest` static method is listed in Table 9-15.

Table 9-15 OracleNotificationRequest Static Method

Static Method	Description
`Equals`	Inherited from `System.Object`

9.2.3 OracleNotificationRequest Properties

The `OracleNotificationRequest` properties are listed in Table 9-16.

Table 9-16 OracleNotificationRequest Properties

Properties	Description
IsNotifiedOnce	Indicates whether or not the registration is to be removed upon notification
IsPersistent	Indicates whether or not the notification message should be queued persistently in the database before delivery
Timeout	Specifies the time that the registration remains alive
GroupingNotificationEnabled	Specifies whether grouping notification is enabled or not
GroupingType	Specifies the type of grouping notification
GroupingInterval	Specifies the interval between grouping notifications, in seconds

9.2.3.1 IsNotifiedOnce

This property indicates whether or not the registration is to be removed upon notification.

Declaration

```
// 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.

9.2.3.2 IsPersistent

This property indicates whether or not the notification message should be queued persistently in the database until delivery.

Declaration

```
// 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.

9.2.3.3 Timeout

This property specifies the time, in seconds, that the registration remains alive.

Declaration

```
// 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.

9.2.3.4 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.

9.2.3.5 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.

9.2.3.6 GroupingInterval

This property specifies the interval of grouping notification in seconds. The group notifications are delivered at intervals specified by this property.

Declaration

```
// 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.

9.2.4 OracleNotificationRequest Methods

`OracleNotificationRequest` methods are listed in Table 9-17.

Table 9-17 OracleNotificationRequest Methods

Methods	Description
`Equals`	Inherited from `System.Object`
`GetHashCode`	Inherited from `System.Object`
`GetType`	Inherited from `System.Object`
`ToString`	Inherited from `System.Object`

9.3 OracleNotificationEventArgs Class

The `OracleNotificationEventArgs` class provides event data for a notification.

Class Inheritance

```
System.Object

  System.EventArgs

    Oracle.DataAccess.Client.OracleNotificationEventArgs
```

Declaration

```
// C#
public sealed class OracleNotificationEventArgs
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver
Assembly	`Oracle.DataAccess.dll`	`Oracle.ManagedDataAccess.dll`
Namespace	`Oracle.DataAccess.Client`	`Oracle.ManagedDataAccess.Client`
.NET Framework	3.5, 4.5, 4.6	4.5, 4.6

Not supported in a .NET stored procedure

Thread Safety

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

9.3.1 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

Static Field	Description
`Empty`	Inherited from `System.EventArgs`

OracleNotificationEventArgs Static Methods

The `OracleNotificationEventArgs` static method is listed in Table 9-19.

Table 9-19 OracleNotificationEventArgs Static Method

Static Method	Description
`Equals`	Inherited from `System.Object`

OracleNotificationEventArgs Properties

`OracleNotificationEventArgs` properties are listed in Table 9-20.

Table 9-20 OracleNotificationEventArgs Properties

Properties	Description
Details	Contains detailed information about the current notification

Table 9-20 (Cont.) OracleNotificationEventArgs Properties

Properties	Description
Info	Indicates the database events for the notification
ResourceNames	Indicates the database resources related to the current notification
Source	Returns the database event source for the notification
Type	Returns the database event type for the notification

OracleNotificationEventArgs Methods

`OracleNotificationEventArgs` methods are listed in Table 9-21.

Table 9-21 OracleNotificationEventArgs Methods

Methods	Description
`Equals`	Inherited from `System.Object`
`GetHashCode`	Inherited from `System.Object`
`GetType`	Inherited from `System.Object`
`ToString`	Inherited from `System.Object`

9.3.2 OracleNotificationEventArgs Static Fields

The `OracleNotificationEventArgs` static field is listed in Table 9-22.

Table 9-22 OracleNotificationEventArgs Static Field

Static Field	Description
`Empty`	Inherited from `System.EventArgs`

9.3.3 OracleNotificationEventArgs Static Methods

The `OracleNotificationEventArgs` static method is listed in Table 9-23.

Table 9-23 OracleNotificationEventArgs Static Method

Static Method	Description
`Equals`	Inherited from `System.Object`

9.3.4 OracleNotificationEventArgs Properties

`OracleNotificationEventArgs` properties are listed in Table 9-24.

Table 9-24 OracleNotificationEventArgs Properties

Properties	Description
Details	Contains detailed information about the current notification
Info	Indicates the database events for the notification
ResourceNames	Indicates the database resources related to the current notification
Source	Returns the database event source for the notification
Type	Returns the database event type for the notification

9.3.4.1 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 9-25 DataTable Object Column Data

Name	Type	Description
`ResourceName`	`System.String`	The resource name of the invalidated object in the format `<Schema_name>.<object_name>`
`Info`	`OracleNotificationInfo`	The information about the database event that occurs on a resource
`Rowid`	`System.String`	The rowid for the invalidated table row
`QueryId`	`Int32`	The `CHANGE_NOTIFICATION_QUERY_ID`

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_ID`s 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`.

9.3.4.2 Info

This property indicates the database events for the notification.

Declaration

```
// 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.Alter`ed or `OracleNotificationInfo.Insert.`

To obtain more detailed information from the invalidation message, use the `Details` and the `ResourceNames` properties.

9.3.4.3 ResourceNames

This property indicates the database resources related to the current notification.

Declaration

```
// 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.

9.3.4.4 Source

This property returns the database event source for the notification.

Declaration

```
// 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 either `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`

9.3.4.5 Type

This property returns the database event type for the notification.

Declaration

```
// 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.

9.3.5 OracleNotificationEventArgs Methods

`OracleNotificationEventArgs` methods are listed in Table 9-26.

Table 9-26   OracleNotificationEventArgs Methods

Methods	Description
`Equals`	Inherited from `System.Object`
`GetHashCode`	Inherited from `System.Object`
`GetType`	Inherited from `System.Object`
`ToString`	Inherited from `System.Object`

9.4 OnChangeEventHandler Delegate

The `OnChangeEventHandler` delegate represents the signature of the method that handles the notification.

Declaration

```
// C#
public delegate void OnChangeEventHandler(object sender,
   OracleNotificationEventArgs args);
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver
Assembly	`Oracle.DataAccess.dll`	`Oracle.ManagedDataAccess.dll`
Namespace	`Oracle.DataAccess.Client`	`Oracle.ManagedDataAccess.Client`
.NET Framework	3.5, 4.5, 4.6	4.5, 4.6

Not supported in a .NET stored procedure

Parameters

- *`sender`*

  The source of the event.

- *`args`*

  The `OracleNotificationEventArgs` instance that contains the event data.

9.5 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

Member Name	Description
`Default`	`ROWID` information is included only if `OracleCommand.AddRowid` property is set to true or if `ROWID` column is explicitly included in the query.
`Include`	`ROWID` information is included regardless of whether `ROWID` is included in the select-list of the query or not.
`Exclude`	`ROWID` information is not included regardless of whether `ROWID` is included in the select-list of the query or not.

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver
Assembly	`Oracle.DataAccess.dll`	`Oracle.ManagedDataAccess.dll`
Namespace	`Oracle.DataAccess.Client`	`Oracle.ManagedDataAccess.Client`
.NET Framework	3.5, 4.5, 4.6	4.5, 4.6

9.6 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

Member Name	Description
`Change`	A change occurs in the database.

Table 9-28 (Cont.) OracleNotificationType Members

Member Name	Description
`Subscribe`	A change occurs in the subscription.
`Query`	A query-based change occurs in the database.

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver
Assembly	`Oracle.DataAccess.dll`	`Oracle.ManagedDataAccess.dll`
Namespace	`Oracle.DataAccess.Client`	`Oracle.ManagedDataAccess.Client`
.NET Framework	3.5, 4.5, 4.6	4.5, 4.6

9.7 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

Member Name	Description
`Data`	The data in a table has changed.
`Database`	A database event such as a database startup or shutdown occurs.
`Object`	A database object is altered or dropped.
`Subscription`	The subscription is changed.

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver
Assembly	`Oracle.DataAccess.dll`	`Oracle.ManagedDataAccess.dll`
Namespace	`Oracle.DataAccess.Client`	`Oracle.ManagedDataAccess.Client`
.NET Framework	3.5, 4.5, 4.6	4.5, 4.6

9.8 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 9-30 OracleNotificationInfo Members

Member Name	Description
`Insert`	A row is inserted.
`Delete`	A row is deleted.
`Update`	A row is updated.
`Startup`	A database starts.
`Shutdown`	A database shuts down.
`Shutdown_any`	A database instance in a Real Application Cluster (Oracle RAC) environment shuts down.
`Alter`	An object is altered.
`Drop`	An object or database is dropped.
`End`	A registration is removed.
`Error`	A notification error occurs.

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver
Assembly	`Oracle.DataAccess.dll`	`Oracle.ManagedDataAccess.dll`
Namespace	`Oracle.DataAccess.Client`	`Oracle.ManagedDataAccess.Client`
.NET Framework	3.5, 4.5, 4.6	4.5, 4.6

10 Oracle Data Provider for .NET Globalization Classes

This chapter describes the ODP.NET globalization classes.

This chapter contains these topics:

- OracleGlobalization Class

10.1 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

  Oracle.DataAccess.Client.OracleGlobalization
```

Declaration

```
public sealed class OracleGlobalization : ICloneable, IDisposable
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver
Assembly	`Oracle.DataAccess.dll`	`Oracle.ManagedDataAccess.dll`
Namespace	`Oracle.DataAccess.Client`	`Oracle.ManagedDataAccess.Client`
.NET Framework	3.5, 4.5, 4.6	4.5, 4.6

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();
  }
}
```

10.1.1 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

Name	Description
GetClientInfo	Returns an `OracleGlobalization` object that represents the Oracle globalization settings of the local computer (Overloaded) *Not Available in ODP.NET, Managed Driver*
GetThreadInfo	Returns or refreshes an `OracleGlobalization` instance that represents Oracle globalization settings of the current thread (Overloaded) *Not Available in ODP.NET, Managed Driver*
SetThreadInfo	Sets Oracle globalization parameters to the current thread *Not Available in ODP.NET, Managed Driver*

OracleGlobalization Properties

The `OracleGlobalization` properties are listed in Table 10-2.

Table 10-2 OracleGlobalization Properties

Name	Description
Calendar	Specifies the calendar system
ClientCharacterSet	Specifies a client character set *Not Available in ODP.NET, Managed Driver*
Comparison	Specifies a method of comparison for `WHERE` clauses and comparison in PL/SQL blocks
Currency	Specifies the string to use as a local currency symbol for the L number format element
DateFormat	Specifies the date format for Oracle `Date` type as a string
DateLanguage	Specifies the language used to spell day and month names and date abbreviations
DualCurrency	Specifies the dual currency symbol, such as *Euro*, for the U number format element
ISOCurrency	Specifies the string to use as an international currency symbol for the C number format element
Language	Specifies the default language of the database
LengthSemantics	Enables creation of `CHAR` and `VARCHAR2` columns using either byte or character (default) length semantics
NCharConversionException	Determines whether or not data loss during an implicit or explicit character type conversion reports an error
NumericCharacters	Specifies the characters used for the decimal character and the group separator character for numeric values in strings
Sort	Specifies the collating sequence for `ORDER` by clause
Territory	Specifies the name of the territory
TimeStampFormat	Specifies the string format for `TimeStamp` types
TimeStampTZFormat	Specifies the string format for `TimeStampTZ` types
TimeZone	Specifies the time zone region name

OracleGlobalization Public Methods

`OracleGlobalization` public methods are listed in Table 10-3.

Table 10-3 OracleGlobalization Public Methods

Public Method	Description
Clone	Creates a copy of an `OracleGlobalization` object
Dispose	Releases any resources or memory allocated by the object

10.1.2 OracleGlobalization Static Methods

The `OracleGlobalization` static methods are listed in Table 10-4.

Table 10-4 OracleGlobalization Static Methods

Name	Description
GetClientInfo	Returns an `OracleGlobalization` object that represents the Oracle globalization settings of the local computer (Overloaded) *Not Available in ODP.NET, Managed Driver*
GetThreadInfo	Returns or refreshes an `OracleGlobalization` instance that represents Oracle globalization settings of the current thread (Overloaded) *Not Available in ODP.NET, Managed Driver*
SetThreadInfo	Sets Oracle globalization parameters to the current thread *Not Available in ODP.NET, Managed Driver*

10.1.2.1 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.

10.1.2.2 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();
  }
}
```

10.1.2.3 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();
}
}
```

10.1.2.4 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.

10.1.2.5 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();
  }
}
```

10.1.2.6 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();
  }
}
```

10.1.2.7 SetThreadInfo

This method sets Oracle globalization parameters to the current thread.

Declaration

```
// C#
public static void SetThreadInfo(OracleGlobalization oraGlob);
```

Parameters

- *oraGlob*

  An `OracleGlobalization` object.

Remarks

Any .NET string conversions to and from ODP.NET Types, as well as ODP.NET Type constructors, use the globalization property values where applicable. For example, when constructing an `OracleDate` structure from a .NET string, that string is expected to be in the format specified by the `OracleGlobalization.DateFormat` property of the thread.

Example

```
// C#

using System;
using Oracle.DataAccess.Client;

class SetThreadInfoSample
{
  static void Main()
  {
    // Get thread's globalization info
    OracleGlobalization glob1 = OracleGlobalization.GetThreadInfo();

    // Prints "glob1.Language = AMERICAN"
    Console.WriteLine("glob1.Language = " + glob1.Language);

    // Set language on thread's globalization info
    glob1.Language = "FRENCH";
    OracleGlobalization.SetThreadInfo(glob1);
    OracleGlobalization glob2 = OracleGlobalization.GetThreadInfo();

    // Prints "glob2.Language = FRENCH"
    Console.WriteLine("glob2.Language = " + glob2.Language);

    glob1.Dispose();
    glob2.Dispose();
  }
}
```

10.1.3 OracleGlobalization Properties

The `OracleGlobalization` properties are listed in Table 10-5.

Table 10-5 OracleGlobalization Properties

Name	Description
Calendar	Specifies the calendar system
ClientCharacterSet	Specifies a client character set *Not Available in ODP.NET, Managed Driver*
Comparison	Specifies a method of comparison for `WHERE` clauses and comparison in PL/SQL blocks

Table 10-5 (Cont.) OracleGlobalization Properties

Name	Description
Currency	Specifies the string to use as a local currency symbol for the L number format element
DateFormat	Specifies the date format for Oracle `Date` type as a string
DateLanguage	Specifies the language used to spell day and month names and date abbreviations
DualCurrency	Specifies the dual currency symbol, such as *Euro*, for the U number format element
ISOCurrency	Specifies the string to use as an international currency symbol for the C number format element
Language	Specifies the default language of the database
LengthSemantics	Enables creation of `CHAR` and `VARCHAR2` columns using either byte or character (default) length semantics
NCharConversionException	Determines whether or not data loss during an implicit or explicit character type conversion reports an error
NumericCharacters	Specifies the characters used for the decimal character and the group separator character for numeric values in strings
Sort	Specifies the collating sequence for `ORDER` by clause
Territory	Specifies the name of the territory
TimeStampFormat	Specifies the string format for `TimeStamp` types
TimeStampTZFormat	Specifies the string format for `TimeStampTZ` types
TimeZone	Specifies the time zone region name

10.1.3.1 Calendar

This property specifies the calendar system.

Declaration

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

Property Value

A string representing the `Calendar`.

Exceptions

`ObjectDisposedException` - The object is already disposed.

Remarks

The default value is the `NLS_CALENDAR` setting of the local computer. This value is the same regardless of whether or not the `OracleGlobalization` object represents the settings of the client, thread, or session.

10.1.3.2 ClientCharacterSet

This property specifies a client character set.

Declaration

```
// C#
public string ClientCharacterSet {get;}
```

Property Value

A string that the provides the name of the character set of the local computer.

Remarks

The default value is the character set of the local computer.

10.1.3.3 Comparison

This property represents a method of comparison for `WHERE` clauses and comparison in PL/SQL blocks.

Declaration

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

Property Value

A string that provides the name of the method of comparison.

Exceptions

`ObjectDisposedException` - The object is already disposed.

Remarks

The default value is the `NLS_COMP` setting of the local computer.

10.1.3.4 Currency

This property specifies the string to use as a local currency symbol for the L number format element.

Declaration

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

Property Value

The string to use as a local currency symbol for the L number format element.

Exceptions

`ObjectDisposedException` - The object is already disposed.

Remarks

The default value is the `NLS_CURRENCY` setting of the local computer.

10.1.3.5 DateFormat

This property specifies the date format for Oracle `Date` type as a string.

Declaration

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

Property Value

The date format for Oracle `Date` type as a string

Exceptions

`ObjectDisposedException` - The object is already disposed.

Remarks

The default value is the `NLS_DATE_FORMAT` setting of the local computer.

10.1.3.6 DateLanguage

This property specifies the language used to spell names of days and months, and date abbreviations (for example: a.m., p.m., AD, BC).

Declaration

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

Property Value

A string specifying the language.

Exceptions

`ObjectDisposedException` - The object is already disposed.

Remarks

The default value is the `NLS_DATE_LANGUAGE` setting of the local computer.

10.1.3.7 DualCurrency

This property specifies the dual currency symbol, such as *Euro*, for the U number format element.

Declaration

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

Property Value

A string that provides the dual currency symbol.

Exceptions

`ObjectDisposedException` - The object is already disposed.

Remarks

The default value is the `NLS_DUAL_CURRENCY` setting of the local computer.

10.1.3.8 ISOCurrency

This property specifies the string to use as an international currency symbol for the C number format element.

Declaration

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

Property Value

The string used as an international currency symbol.

Exceptions

`ObjectDisposedException` - The object is already disposed.

Remarks

The default value is the `NLS_ISO_CURRENCY` setting of the local computer.

10.1.3.9 Language

This property specifies the default language of the database.

Declaration

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

Property Value

The default language of the database.

Exceptions

`ObjectDisposedException` - The object is already disposed.

Remarks

The default value is the NLS_LANGUAGE setting of the local computer.

Language is used for messages, day and month names, and sorting algorithms. It also determines NLS_DATE_LANGUAGE and NLS_SORT parameter values.

10.1.3.10 LengthSemantics

This property indicates whether or not CHAR and VARCHAR2 columns use byte or character (default) length semantics.

Declaration

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

Property Value

A string that indicates either byte or character length semantics.

Exceptions

ObjectDisposedException - The object is already disposed.

Remarks

The default value is the NLS_LENGTH_SEMANTICS setting of the local computer.

10.1.3.11 NCharConversionException

This property determines whether or not data loss during an implicit or explicit character type conversion reports an error.

Declaration

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

Property Value

A string that indicates whether or not a character type conversion causes an error message.

Exceptions

ObjectDisposedException - The object is already disposed.

Remarks

The default value of NLS_NCHAR_CONV_EXCP is False, unless it is overridden by a setting in the INIT.ORA file.

10.1.3.12 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.

10.1.3.13 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.

10.1.3.14 Territory

This property specifies the name of the territory.

Declaration

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

Property Value

A string that provides the name of the territory.

Exceptions

`ObjectDisposedException` - The object is already disposed.

Remarks

The default value is the `NLS_TERRITORY` setting of the local computer.

Changing this property changes other globalization properties.

10.1.3.15 TimeStampFormat

This property specifies the string format for `TimeStamp` types.

Declaration

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

Property Value

The string format for `TimeStamp` types.

Exceptions

`ObjectDisposedException` - The object is already disposed.

Remarks

The default value is the `NLS_TIMESTAMP_FORMAT` setting of the local computer.

10.1.3.16 TimeStampTZFormat

This property specifies the string format for `TimeStampTZ` types.

Declaration

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

Property Value

The string format for `TimeStampTZ` types.

Exceptions

`ObjectDisposedException` - The object is already disposed.

Remarks

The default value is the `NLS_TIMESTAMP_TZ_FORMAT` setting of the local computer.

10.1.3.17 TimeZone

This property specifies the time zone region name or hour offset.

Declaration

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

Property Value

The string represents the time zone region name or the time zone offset.

Exceptions

`ObjectDisposedException` - The object is already disposed.

Remarks

The default value is the time zone region name of the local computer

`TimeZone` is only used when the thread constructs one of the `TimeStamp` structures. `TimeZone` has no effect on the session.

`TimeZone` can be either an hour offset, for example, 7:00, or a valid time zone region name that is provided in `V$TIMEZONE_NAMES`, such as US/Pacific. Time zone abbreviations are not supported.

Note:

PST is a time zone region name as well as a time zone abbreviation; therefore it is accepted by `OracleGlobalization`.

This property returns an empty string if the `OracleGlobalization` object is obtained using `GetSessionInfo()` or `GetSessionInfo(OracleGlobalization)`. Initially, by default, the time zone of the session is identical to the time zone of the thread. Therefore, given that the session time zone is not changed by invoking `ALTER SESSION` calls, the session time zone can be fetched from the client's globalization settings.

10.1.4 OracleGlobalization Public Methods

`OracleGlobalization` public methods are listed in Table 10-6.

Table 10-6 OracleGlobalization Public Methods

Public Method	Description
Clone	Creates a copy of an `OracleGlobalization` object
Dispose	Releases any resources or memory allocated by the object

10.1.4.1 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.

10.1.4.2 Dispose

This method releases any resources or memory allocated by the object.

Declaration

```
// C#
public void Dispose();
```

Implements

```
IDisposable
```

Remarks

The `Dispose` method also closes the `OracleGlobalization` object.

11 Oracle Data Provider for .NET Failover Classes

This chapter describes the ODP.NET failover classes and enumerations.

This chapter contains these topics:

11.1 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

    Oracle.DataAccess.Client.OracleFailoverEventArgs
```

Declaration

```
// C#
public sealed class OracleFailoverEventArgs
```

Requirements

Provider	ODP.NET, Unmanaged Driver
Assembly	`Oracle.DataAccess.dll`
Namespace	`Oracle.DataAccess.Client`
.NET Framework	3.5, 4.5, 4.6

Not supported in a .NET stored procedure

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;
      }
    }
```

11.1.1 OracleFailoverEventArgs Members

`OracleFailoverEventArgs` members are listed in the following tables.

OracleFailoverEventArgs Static Methods

The `OracleFailoverEventArgs` static methods are listed in Table 11-1.

Table 11-1 OracleFailoverEventArgs Static Methods

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

OracleFailoverEventArgs Properties

The `OracleFailoverEventArgs` properties are listed in Table 11-2.

Table 11-2 OracleFailoverEventArgs Properties

Name	Description
FailoverType	Specifies the type of failover the client has requested
FailoverEvent	Indicates the state of the failover

OracleFailoverEventArgs Public Methods

The `OracleFailoverEventArgs` public methods are listed in Table 11-3.

Table 11-3 OracleFailoverEventArgs Public Methods

Name	Description
`Equals`	Inherited from `System.Object` (Overloaded)
`GetHashCode`	Inherited from `System.Object`
`GetType`	Inherited from `System.Object`
`ToString`	Inherited from `System.Object`

11.1.2 OracleFailoverEventArgs Static Methods

The `OracleFailoverEventArgs` static methods are listed in Table 11-1.

Table 11-4 OracleFailoverEventArgs Static Methods

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

11.1.3 OracleFailoverEventArgs Properties

The `OracleFailoverEventArgs` properties are listed in Table 11-5.

Table 11-5 OracleFailoverEventArgs Properties

Name	Description
FailoverType	Specifies the type of failover the client has requested
FailoverEvent	Indicates the state of the failover

11.1.3.1 FailoverType

This property indicates the state of the failover.

Declaration

```
// C#
public FailoverType FailoverType {get;}
```

Property Value

A `FailoverType` enumeration value.

11.1.3.2 FailoverEvent

This property indicates the state of the failover.

Declaration

```
// C#
public FailoverEvent FailoverEvent {get;}
```

Property Value

A `FailoverEvent` enumerated value.

11.1.4 OracleFailoverEventArgs Public Methods

The `OracleFailoverEventArgs` public methods are listed in Table 11-6.

Table 11-6 OracleFailoverEventArgs Public Methods

Name	Description
`Equals`	Inherited from `System.Object` (Overloaded)
`GetHashCode`	Inherited from `System.Object`
`GetType`	Inherited from `System.Object`
`ToString`	Inherited from `System.Object`

11.2 OracleFailoverEventHandler Delegate

The `OracleFailoverEventHandler` represents the signature of the method that handles the `OracleConnection.Failover` event.

Declaration

```
// C#
public delegate FailoverReturnCode OracleFailoverEventHandler(object sender,
    OracleFailoverEventArgs eventArgs);
```

Requirements

Provider	ODP.NET, Unmanaged Driver
Assembly	`Oracle.DataAccess.dll`
Namespace	`Oracle.DataAccess.Client`
.NET Framework	3.5, 4.5, 4.6

Not supported in a .NET stored procedure

Parameter

- *`sender`*

  The source of the event.

- *`eventArgs`*

  The `OracleFailoverEventArgs` object that contains the event data.

Return Type

An `int`.

Remarks

To receive failover notifications, a callback function can be registered as follows:

```
ConObj.Failover += new OracleFailoverEventHandler(OnFailover);
```

The definition of the callback function `OnFailover` can be as follows:

```
public FailoverReturnCode OnFailover(object sender, OracleFailoverEventArgs
eventArgs)
```

11.3 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

Member Names	Description
`FailoverEvent.Begin`	Indicates that failover has detected a lost connection and that failover is starting.
`FailoverEvent.End`	Indicates successful completion of failover.
`FailoverEvent.Abort`	Indicates that failover was unsuccessful, and there is no option of retrying.
`FailoverEvent.Error`	Indicates that failover was unsuccessful, and it gives the application the opportunity to handle the error and retry failover. The application can retry failover by returning `FailoverReturnCode.Retry` for the event notification.
`FailoverEvent.Reauth`	Indicates that a user handle has been reauthenticated. This applies to the situation where a client has multiple user sessions on a single server connection. During the initial failover, only the active user session is failed over. Other sessions are failed over when the application tries to use them. This is the value passed to the callback during these subsequent failovers.

No significant database operation should occur immediately after a `FailoverEvent.Begin` event. SQL and major database operations should wait until the `FailoverEvent.End` event. `FailoverEvent.Begin` is primarily used to reject failover or to trace it. `FailoverEvent.Begin` can also be used for non-database application operations, such as informing the end user a failover is in progress and to wait until it completes before proceeding. Transactions can be used in the `FailoverEvent.End` callback phase, such as to file fault tickets or audit. These transactions must be committed before the callback completes.

Requirements

Provider	ODP.NET, Unmanaged Driver
Assembly	`Oracle.DataAccess.dll`
Namespace	`Oracle.DataAccess.Client`
.NET Framework	3.5, 4.5, 4.6

11.4 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

Member Names	Description
`FailoverReturnCode.Retry`	Requests ODP.NET to retry failover in case `FailoverEvent.Error` is passed to the application
`FailoverReturnCode.Success`	Requests ODP.NET to proceed so that the application receive more notifications, if any

Requirements

Provider	ODP.NET, Unmanaged Driver
Assembly	`Oracle.DataAccess.dll`
Namespace	`Oracle.DataAccess.Client`
.NET Framework	3.5, 4.5, 4.6

11.5 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 11-9 FailoverType Enumeration Values

Member Names	Description
`FailoverType.Session`	Indicates that the user has requested only session failover.
`FailoverType.Select`	Indicates that the user has requested select and session failover.

Requirements

Provider	ODP.NET, Unmanaged Driver
Assembly	`Oracle.DataAccess.dll`
Namespace	`Oracle.DataAccess.Client`
.NET Framework	3.5, 4.5, 4.6

12 Oracle Database Advanced Queuing Classes

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

12.1 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
```

Requirements

Provider	ODP.NET, Unmanaged Driver
Assembly	`Oracle.DataAccess.dll`

Provider	ODP.NET, Unmanaged Driver
Namespace	`Oracle.DataAccess.Client`
.NET Framework	3.5, 4.5, 4.6

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.

12.1.1 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

Constructor	Description
OracleAQAgent Constructors	Instantiates a new instance of the `OracleAQAgent` class (Overloaded).

OracleAQAgent Properties

`OracleAQAgent` properties are listed in Table 12-2.

Table 12-2 OracleAQAgent Properties

Property	Description
Address	Specifies the address of the agent.
Name	Specifies the name of the agent.

12.1.2 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.

12.1.2.1 OracleAQAgent (string)

This constructor instantiates the `OracleAQAgent` class using the specified name.

Declaration

```
// C#
public OracleAQAgent(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.

12.1.2.2 OracleAQAgent (string, string)

This constructor instantiates the `OracleAQAgent` class using the specified name and address.

Declaration

```
// 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`.

12.1.3 OracleAQAgent Properties

`OracleAQAgent` properties are listed in Table 12-3.

Table 12-3 OracleAQAgent Properties

Property	Description
Address	Specifies the address of the agent.
Name	Specifies the name of the agent.

12.1.3.1 Address

This instance property specifies the address of the agent.

Declaration

```
// 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.

12.1.3.2 Name

This instance property specifies the name of the agent.

Declaration

```
// 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.

12.2 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
```

Requirements

Provider	ODP.NET, Unmanaged Driver
Assembly	`Oracle.DataAccess.dll`
Namespace	`Oracle.DataAccess.Client`
.NET Framework	3.5, 4.5, 4.6

Thread Safety

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

12.2.1 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

Constructor	Description
OracleAQDequeueOptions Constructor	Instantiates a new instance of the `OracleAQDequeueOptions` class

OracleAQDequeueOptions Properties

`OracleAQDequeueOptions` properties are listed in Table 12-5.

Table 12-5 OracleAQDequeueOptions Properties

Property	Description
ConsumerName	Specifies the consumer name for which to dequeue the message
Correlation	Specifies the correlation identifier of the message to be dequeued
DeliveryMode	Specifies the expected delivery mode of the message being dequeued
DequeueMode	Specifies the locking behavior associated with the dequeue operation
MessageId	Specifies the message identifier of the message to be dequeued
NavigationMode	Specifies the position of the message that will be retrieved
ProviderSpecificType	Specifies whether the payload of a dequeued message is provided as an ODP.NET specific type or a .NET type
Visibility	Specifies whether or not the new message is dequeued as part of the current transaction
Wait	Specifies the wait time, in seconds, for a message that matches the search criteria

OracleAQDequeueOptions Public Methods

`OracleAQDequeueOptions` public methods are listed in Table 12-6.

Table 12-6 OracleAQDequeueOptions Public Methods

Public Method	Description
Clone	Creates a copy of an `OracleAQDequeueOptions` object.

12.2.2 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();
```

12.2.3 OracleAQDequeueOptions Properties

`OracleAQDequeueOptions` properties are listed in Table 12-7.

Table 12-7 OracleAQDequeueOptions Properties

Property	Description
ConsumerName	Specifies the consumer name for which to dequeue the message
Correlation	Specifies the correlation identifier of the message to be dequeued
DeliveryMode	Specifies the expected delivery mode of the message being dequeued
DequeueMode	Specifies the locking behavior associated with the dequeue operation
MessageId	Specifies the message identifier of the message to be dequeued
NavigationMode	Specifies the position of the message that will be retrieved
ProviderSpecificType	Specifies whether the payload of a dequeued message is provided as an ODP.NET specific type or a .NET type
Visibility	Specifies whether or not the new message is dequeued as part of the current transaction
Wait	Specifies the wait time, in seconds, for a message that matches the search criteria

12.2.3.1 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`.

12.2.3.2 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`.

12.2.3.3 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`
- `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.

12.2.3.4 DequeueMode

This instance property specifies the locking behavior associated with the dequeue operation.

Declaration

```
// C#
public OracleAQDequeueMode DequeueMode {get;set;}
```

Property Value

An `OracleAQDequeueMode` enumerated value.

Exceptions

`ArgumentOutOfRangeException` - The specified `DequeueMode` value is invalid.

Remarks

The default value is `OracleAQDequeueMode.Remove`.

12.2.3.5 MessageId

This instance property specifies the message identifier of the message to be dequeued.

Declaration

```
// 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.

12.2.3.6 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`.

12.2.3.7 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.

12.2.3.8 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`.

12.2.3.9 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.

12.2.4 OracleAQDequeueOptions Public Methods

The `OracleAQDequeueOptions` public method is listed in Table 12-8.

Table 12-8 OracleAQDequeueOptions Public Methods

Public Method	Description
Clone	Creates a copy of an `OracleAQDequeueOptions` object

12.2.4.1 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.

12.3 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
```

Requirements

Provider	ODP.NET, Unmanaged Driver
Assembly	`Oracle.DataAccess.dll`
Namespace	`Oracle.DataAccess.Client`
.NET Framework	3.5, 4.5, 4.6

Thread Safety

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

12.3.1 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

Constructor	Description
OracleAQEnqueueOptions Constructor	Instantiates a new instance of the `OracleAQEnqueueOptions` class.

OracleAQEnqueueOptions Properties

`OracleAQEnqueueOptions` properties are listed in Table 12-10.

Table 12-10 OracleAQEnqueueOptions Properties

Property	Description
DeliveryMode	Specifies the delivery mode of the message being enqueued.
Visibility	Specifies whether or not the new message is enqueued as part of the current transaction.

OracleAQEnqueueOptions Public Methods

The `OracleAQEnqueueOptions` public method is listed in Table 12-11.

Table 12-11 OracleAQEnqueueOptions Public Methods

Public Method	Description
Clone	Creates a copy of an `OracleAQEnqueueOptions` object.

12.3.2 OracleAQEnqueueOptions Constructor

This constructor creates an instance of the `OracleAQEnqueueOptions` class with default property values.

Declaration

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

12.3.3 OracleAQEnqueueOptions Properties

`OracleAQEnqueueOptions` properties are listed in Table 12-12.

Table 12-12 OracleAQEnqueueOptions Properties

Property	Description
DeliveryMode	Specifies the delivery mode of the message being enqueued.
Visibility	Specifies whether or not the new message is enqueued as part of the current transaction.

12.3.3.1 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.PersistentOrBuffered` cannot be set on this property.

12.3.3.2 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.

12.3.4 OracleAQEnqueueOptions Public Methods

`OracleAQEnqueueOptions` public method is listed in Table 12-13.

Table 12-13 OracleAQEnqueueOptions Public Methods

Public Method	Description
Clone	Creates a copy of an `OracleAQEnqueueOptions` object.

12.3.4.1 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.

12.4 OracleAQMessage Class

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

Class Inheritance

```
System.Object
  OracleAQMessage
```

Declaration

```
// C#
public sealed class OracleAQMessage
```

Requirements

Provider	ODP.NET, Unmanaged Driver
Assembly	`Oracle.DataAccess.dll`
Namespace	`Oracle.DataAccess.Client`
.NET Framework	3.5, 4.5, 4.6

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.

> **Note:**
>
> An instance of `OracleAQMessage` cannot be re-used across multiple operations of `OracleAQQueue` public method Enqueue() or EnqueueArray(), if the payload is an `XmlReader`. This is a direct consequence of the forward-only semantics of the `XmlReader`, as an Enqueue() or EnqueueArray() operation internally invokes a read operation on the `XmlReader` to extract the data to be enqueued.

12.4.1 OracleAQMessage Members

`OracleAQMessage` members are listed in the following tables.

OracleAQMessage Constructor

`OracleAQMessage` constructors are listed in Table 12-14.

Table 12-14 OracleAQMessage Constructors

Constructor	Description
OracleAQMessage Constructors	Instantiates a new instance of the `OracleAQMessage` class (Overloaded).

OracleAQMessage Properties

`OracleAQMessage` properties are listed in Table 12-15.

Table 12-15 OracleAQMessage Properties

Property	Description
Correlation	Specifies an identification for the message.
Delay	Specifies the duration, in seconds, after which an enqueued message is available for dequeuing.
DeliveryMode	Specifies the delivery mode of the dequeued message.
DequeueAttempts	Returns the number of attempts that have been made to dequeue the message.

Table 12-15 (Cont.) OracleAQMessage Properties

Property	Description
EnqueueTime	Specifies the time when the message was enqueued.
ExceptionQueue	Specifies the name of the queue that the message should be moved to if it cannot be processed successfully.
Expiration	Specifies the duration, in seconds, for which an enqueued message is available for dequeuing.
MessageId	Returns the message identifier.
OriginalMessageId	Specifies the identifier of the message in the last queue that generated this message.
Payload	Specifies the payload of the message.
Priority	Specifies the priority of the message.
Recipients	Specifies the list of recipients that overrides the default queue subscribers.
SenderId	Identifies the original sender of the message.
State	Specifies the state of the message at the time of dequeue.
TransactionGroup	Specifies the transaction group for the dequeued message.

12.4.2 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.

12.4.2.1 OracleAQMessage()

This constructor instantiates the `OracleAQMessage` class.

Declaration

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

12.4.2.2 OracleAQMessage(Object)

This constructor instantiates the `OracleAQMessage` class using the `Object` provided as the *`payload`*.

Declaration

```
// 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.

12.4.3 OracleAQMessage Properties

`OracleAQMessage` properties are listed in Table 12-16.

Table 12-16 OracleAQMessage Properties

Property	Description
Correlation	Specifies an identification for the message.
Delay	Specifies the duration, in seconds, after which an enqueued message is available for dequeuing.
DeliveryMode	Specifies the delivery mode of the dequeued message.
DequeueAttempts	Returns the number of attempts that have been made to dequeue the message.
EnqueueTime	Specifies the time when the message was enqueued.
ExceptionQueue	Specifies the name of the queue that the message should be moved to if it cannot be processed successfully.
Expiration	Specifies the duration, in seconds, for which an enqueued message is available for dequeuing.
MessageId	Returns the message identifier.
OriginalMessageId	Specifies the identifier of the message in the last queue that generated this message.

Table 12-16 (Cont.) OracleAQMessage Properties

Property	Description
Payload	Specifies the payload of the message.
Priority	Specifies the priority of the message.
Recipients	Specifies the list of recipients that overrides the default queue subscribers.
SenderId	Identifies the original sender of the message.
State	Specifies the state of the message at the time of dequeue.
TransactionGroup	Specifies the transaction group for the dequeued message.

12.4.3.1 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.

12.4.3.2 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.

12.4.3.3 DeliveryMode

This instance property specifies the delivery mode of the dequeued message.

Declaration

```
// C#
public OracleAQMessageDeliveryMode DeliveryMode {get;}
```

Property Value

An `OracleAQMessageDeliveryMode` enumerated value (`OracleAQMessageDeliveryMode.Persistent` or `OracleAQMessageDeliveryMode.Buffered`).

12.4.3.4 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.

12.4.3.5 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.

12.4.3.6 ExceptionQueue

This instance property specifies the name of the queue that the message should be moved to if it cannot be processed successfully.

Declaration

```
// 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.

12.4.3.7 Expiration

This instance property specifies the duration, in seconds, for which an enqueued message is available for dequeuing.

Declaration

```
// C#
public 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.

12.4.3.8 MessageId

This instance property returns the message identifier.

Declaration

```
// 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`.

12.4.3.9 OriginalMessageId

This instance property specifies the identifier 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 identifier.

12.4.3.10 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.

12.4.3.11 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.

12.4.3.12 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.

12.4.3.13 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.

12.4.3.14 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`.

12.4.3.15 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 10*g* 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.

12.5 OracleAQMessageAvailableEventArgs Class

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

Class Inheritance

```
System.Object

  System.EventArgs

    Oracle.DataAccess.Client.OracleAQMessageAvailableEventArgs
```

Declaration

```
// C#
public sealed class OracleAQMessageAvailableEventArgs
```

Requirements

Provider	ODP.NET, Unmanaged Driver
Assembly	`Oracle.DataAccess.dll`
Namespace	`Oracle.DataAccess.Client`
.NET Framework	3.5, 4.5, 4.6

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.

12.5.1 OracleAQMessageAvailableEventArgs Members

`OracleAQMessageAvailableEventArgs` members are listed in the following tables.

OracleAQMessageAvailableEventArgs Constructor

`OracleAQMessageAvailableEventArgs` properties are listed in Table 12-17

Table 12-17 OracleAQMessageAvailableEventArgs Constructor

Property	Description
OracleAQMessageAvailableEventArgs Constructor	Instantiates a new instance of the OracleAQMessageAvailableEventArgs class.

OracleAQMessageAvailableEventArgs Properties

`OracleAQMessageAvailableEventArgs` properties are listed in Table 12-18.

Table 12-18 OracleAQMessageAvailableEventArgs Properties

Property	Description
AvailableMessages	Specifies the number of messages that raised this notification.
ConsumerName	Provides the name of the consumer for which the message is available for dequeuing.
Correlation	Provides the name of the consumer for which the message is available for dequeuing.
Delay	Specifies the duration, in seconds, after which an enqueued message is available for dequeuing.
DeliveryMode	Specifies the delivery mode of the message.
EnqueueTime	Specifies the time when the message was enqueued.
ExceptionQueue	Specifies the name of the queue that the message is moved to if it cannot be processed successfully.
Expiration	Specifies the duration, in seconds, for which an enqueued message is available for dequeuing before expiring.
MessageId	Returns an array of message identifiers.
NotificationType	Indicates the type of notification such as regular, grouping, or timeout.
OriginalMessageId	Specifies the ID of the message, in the last queue, that generated this message.
Priority	Specifies the priority of the message.
QueueName	Indicates the name of the queue that contains the message to be dequeued.
SenderId	Identifies the original sender of the message.
State	Specifies the state of the message.

12.5.2 OracleAQMessageAvailableEventArgs Constructor

This constructor creates an instance of the `OracleAQMessageAvailableEventArgs` class with default property values.

Declaration

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

12.5.3 OracleAQMessageAvailableEventArgs Properties

`OracleAQMessageAvailableEventArgs` properties are listed in Table 12-19.

Table 12-19 OracleAQMessageAvailableEventArgs Properties

Property	Description
AvailableMessages	Specifies the number of messages that raised this notification.
ConsumerName	Provides the name of the consumer for which the message is available for dequeuing.
Correlation	Provides the name of the consumer for which the message is available for dequeuing.
Delay	Specifies the duration, in seconds, after which an enqueued message is available for dequeuing.
DeliveryMode	Specifies the delivery mode of the message.
EnqueueTime	Specifies the time when the message was enqueued.
ExceptionQueue	Specifies the name of the queue that the message is moved to if it cannot be processed successfully.
Expiration	Specifies the duration, in seconds, for which an enqueued message is available for dequeuing before expiring.
MessageId	Returns an array of message identifiers.
NotificationType	Indicates the type of notification such as regular, grouping, or timeout.
OriginalMessageId	Specifies the ID of the message, in the last queue, that generated this message.
Priority	Specifies the priority of the message.
QueueName	Indicates the name of the queue that contains the message to be dequeued.
SenderId	Identifies the original sender of the message.
State	Specifies the state of the message.

12.5.3.1 AvailableMessages

This instance property specifies the number of messages that raised this notification.

Declaration

```
// C#
public int AvailableMessages{get;}
```

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`.

12.5.3.2 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.

12.5.3.3 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.

12.5.3.4 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.

12.5.3.5 DeliveryMode

This instance property specifies the delivery mode of the message.

Declaration

```
// C#
public OracleAQMessageDeliveryMode DeliveryMode {get;}
```

Property Value

An `OracleAQMessageDeliveryMode` enumerated value.

12.5.3.6 EnqueueTime

This instance property specifies the time when the message was enqueued.

Declaration

```
// C#
public DateTime EnqueueTime {get;}
```

Property Value

A `DateTime` object.

12.5.3.7 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.

12.5.3.8 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.

12.5.3.9 MessageId

This instance property returns an array of message identifiers.

Declaration

```
// C#
public byte[][] MessageId{get;}
```

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`.

12.5.3.10 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.

12.5.3.11 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.

12.5.3.12 Priority

This instance property specifies the priority of the message.

Declaration

```
// C#
public int Priority {get;}
```

Property Value

An `integer` that specifies the priority of the message.

12.5.3.13 QueueName

This property indicates the name of the queue that contains the message to be dequeued.

Declaration

```
// C#
public string QueueName {get;}
```

Property Value

A `string`.

12.5.3.14 SenderId

This property identifies the original sender of the message.

Declaration

```
// C#
public OracleAQAgent SenderId {get;}
```

Property Value

An `OracleAQAgent` object.

12.5.3.15 State

This instance property specifies the state of the message.

Declaration

```
// C#
public OracleAQMessageState State {get;}
```

Property Value

An `OracleAQMessageState` enumerated value.

12.6 OracleAQMessageAvailableEventHandler Delegate

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

Declaration

```
// C#
public delegate void OracleAQMessageAvailableEventHandler (object
  sender,OracleAQMessageAvailableEventArg eventArgs);
```

Parameters

- *`sender`*

  The source of the event.

- *`eventArgs`*

  The `OracleAQMessageAvailableEventArgs` object that contains the event data.

12.7 OracleAQQueue Class

An `OracleAQQueue` object represents a queue.

Class Inheritance

`System.Object`

  `OracleAQQueue`

Declaration

```
// C#
public class OracleAQQueue : IDisposable
```

Requirements

Provider	ODP.NET, Unmanaged Driver
Assembly	`Oracle.DataAccess.dll`
Namespace	`Oracle.DataAccess.Client`
.NET Framework	3.5, 4.5, 4.6

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.

12.7.1 OracleAQQueue Members

`OracleAQQueue` members are listed in the following tables.

OracleAQQueue Constructors

`OracleAQQueue` constructors are listed in Table 12-20.

Table 12-20 OracleAQQueue Constructors

Constructor	Description
OracleAQQueue Constructors	Instantiate a new instance of the `OracleAQQueue` class (Overloaded).

OracleAQQueue Static Methods

The `OracleAQQueue` static method is listed in Table 12-21.

Table 12-21 OracleAQQueue Static Methods

Static Method	Description
Listen	Listens for messages on one or more queues for one or more consumers, as specified in the array of `OracleAQAgent` objects (Overloaded).

OracleAQQueue Properties

`OracleAQQueue` properties are listed in Table 12-22.

Table 12-22 OracleAQQueue Properties

Property	Description
Connection	Specifies the `OracleConnection` object associated with the queue.
DequeueOptions	Specifies the dequeueing options to use when dequeuing a message from a queue.
EnqueueOptions	Specifies the enqueueing options used to enqueue a message to a queue.
MessageType	Specifies the type of queue table associated with this queue.
Name	Returns the name of the queue.
Notification	Specifies the various notification options for notifications that are registered using the `MessageAvailable` event.
NotificationConsumers	Specifies the array of consumers, for a multiconsumer queue, that are to be notified asynchronously for any incoming messages on the queue.
UdtTypeName	Specifies the type name on which the queue and the corresponding queue table is based if the `MessageType` is `OracleAQMessageType.UDT`.

OracleAQQueue Public Methods

The `OracleAQQueue` public methods are listed in Table 12-23.

Table 12-23 OracleAQQueue Public Methods

Public Method	Description
Dequeue	Dequeues messages from queues (Overloaded).
DequeueArray	Dequeues multiple messages from queues (Overloaded).
Dispose	Releases any resources or memory allocated by the object
Enqueue	Enqueues messages to queues (Overloaded).
EnqueueArray	Enqueues multiple messages to a queue (Overloaded).
Listen	Listens for messages on the queue on behalf of `listenConsumers` (Overloaded).

OracleAQQueue Events

The `OracleAQQueue` event is listed in Table 12-24.

Table 12-24 OracleAQQueue Events

Event Name	Description
MessageAvailable Event	Notifies when a message is available in the queue for `NotificationConsumers`.

12.7.2 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.

12.7.2.1 OracleAQQueue(string)

This constructor takes a queue name to initialize a queue object.

Declaration

```
// 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.

12.7.2.2 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.

12.7.2.3 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`.

12.7.2.4 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

```
// 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.

12.7.3 OracleAQQueue Static Methods

OracleAQQueue static methods are listed in Table 12-25.

Table 12-25 OracleAQQueue Static Methods

Static Method	Description
Listen	Listens for messages on one or more queues for one or more consumers, as specified in the array of OracleAQAgent objects (Overloaded).

12.7.3.1 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.

12.7.3.2 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.

12.7.3.3 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.

12.7.4 OracleAQQueue Properties

`OracleAQQueue` properties are listed in Table 12-26.

Table 12-26   OracleAQQueue Properties

Property	Description
Connection	Specifies the `OracleConnection` object associated with the queue.
DequeueOptions	Specifies the dequeueing options to use when dequeuing a message from a queue.
EnqueueOptions	Specifies the enqueueing options used to enqueue a message to a queue.
MessageType	Specifies the type of queue table associated with this queue.
Name	Returns the name of the queue.

Table 12-26 (Cont.) OracleAQQueue Properties

Property	Description
Notification	Specifies the various notification options for notifications that are registered using the `MessageAvailable` event.
NotificationConsumers	Specifies the array of consumers, for a multiconsumer queue, that are to be notified asynchronously for any incoming messages on the queue.
UdtTypeName	Specifies the type name on which the queue and the corresponding queue table is based if the `MessageType` is `OracleAQMessageType.UDT`.

12.7.4.1 Connection

This property specifies the `OracleConnection` object associated with the queue.

Declaration

```
// 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`.

12.7.4.2 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.

12.7.4.3 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.

12.7.4.4 MessageType

This instance property specifies the type of queue table associated with this queue.

Declaration

```
// 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 12-27 lists the allowed payload types for various message types.

Table 12-27 Message Types and Payloads

OracleAQQueue.MessageType	Allowed OracleAQMessage.Payload type to Enqueue
`OracleAQMessageType.Raw`	`OracleBinary`, `byte[]`
`OracleAQMessageType.Xml`	`OracleXmlType`, `XmlReader`, `String` (well-formed XML, else exception is raised)
`OracleAQMessageType.UDT`	`UDT` Custom Object

Table 12-28 lists the payload types for dequeued messages.

Table 12-28 Payload Types for Dequeued Messages

OracleAQQueue.MessageType	DequeueOptions.ProviderSpecificType	OracleAQMessage.Payload of dequeued message
`OracleAQMessageType.Xml`	`true`	`OracleXmlType`
`OracleAQMessageType.Xml`	`false`	`XmlReader`
`OracleAQMessageType.Raw`	`true`	`OracleBinary`
`OracleAQMessageType.Raw`	`false`	`Byte[]`
`OracleAQMessageType.UDT`	N.A.	`UDT` Custom Object

12.7.4.5 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.

12.7.4.6 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.

12.7.4.7 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.

12.7.4.8 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.

12.7.5 OracleAQQueue Public Methods

`OracleAQQueue` public methods are listed in Table 12-29.

Table 12-29 OracleAQQueue Public Methods

Public Method	Description
Dequeue	Dequeues messages from queues (Overloaded).
DequeueArray	Dequeues multiple messages from queues (Overloaded).
Dispose	Releases any resources or memory allocated by the object
Enqueue	Enqueues messages to queues (Overloaded).
EnqueueArray	Enqueues multiple messages to a queue (Overloaded).
Listen	Listens for messages on the queue on behalf of `listenConsumers` (Overloaded).

12.7.5.1 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.

12.7.5.2 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.

12.7.5.3 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`.

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.

12.7.5.4 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.

12.7.5.5 DequeueArray(int)

This instance method dequeues multiple messages from a queue using the DequeueOptions of the instance.

Declaration

```
// 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 10*g* 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 12*c* Release 2 (12.2), 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.

12.7.5.6 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.

`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 10*g* 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 dependes on the actual number of messages present in the queue.

For database versions earlier than Oracle Database 12*c* Release 2 (12.2), 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.

12.7.5.7 Dispose

This method releases any resources or memory allocated by the object.

Declaration

```
// C#
public void Dispose();
```

Implements

`IDisposable`.

12.7.5.8 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.

12.7.5.9 Enqueue(OracleAQMessage)

This instance method enqueues messages to a queue using the `EnqueueOptions` of the instance.

Declaration

```
// 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.

12.7.5.10 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.

12.7.5.11 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.

12.7.5.12 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 10*g* 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.

12.7.5.13 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 10*g* 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.

12.7.5.14 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.

12.7.5.15 Listen(string[])

This method listens for messages on the queue on behalf of `listenConsumers`.

Declaration

```
// C#
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.

```
-- 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;

------------------------------------------------------------------
-- 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 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.Visibility = OracleAQVisibilityMode.OnCommit;
        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();
      }
    }
  }
```

12.7.5.16 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.

12.7.6 OracleAQQueue Events

The `OracleAQQueue` event is listed in Table 12-30.

Table 12-30 OracleAQQueue Events

Event Name	Description
MessageAvailable Event	Notifies when a message is available in the queue for `NotificationConsumers`.

12.7.6.1 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 continuous query 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:

```
<configuration>
  <oracle.dataaccess.client>
    <settings>
      <add name="DbNotificationPort" value="1200"/>
    </settings>
  </oracle.dataaccess.client>
</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 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();
      }
    }
  }
```

12.8 OracleAQDequeueMode Enumeration

Table 12-31 lists all the `OracleAQDequeueMode` enumeration values with a description of each enumerated value.

Table 12-31 OracleAQDequeueMode Members

Member Name	Description
`Browse`	Reads the message without acquiring any lock on the message. This is equivalent to a `SELECT` statement.
`Locked`	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.
`Remove`	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
`RemoveNoData`	Confirms receipt of the message but does not deliver the actual message content.

Requirements

Provider	ODP.NET, Unmanaged Driver
Assembly	`Oracle.DataAccess.dll`
Namespace	`Oracle.DataAccess.Client`
.NET Framework	3.5, 4.5, 4.6

12.9 OracleAQMessageDeliveryMode Enumeration

The `OracleAQMessageDeliveryMode` enumeration type specifies the delivery mode of the message.

Table 12-32 lists all the `OracleAQMessageDeliveryMode` enumeration values with a description of each enumerated value.

Table 12-32 OracleAQMessageDeliveryMode Members

Member Name	Description
`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. Dequeuing 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`. Transaction grouping queues and array enqueues are not supported for buffered messages in Oracle Database 11*g* 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

Provider	ODP.NET, Unmanaged Driver
Assembly	`Oracle.DataAccess.dll`
Namespace	`Oracle.DataAccess.Client`
.NET Framework	3.5, 4.5, 4.6

12.10 OracleAQMessageState Enumeration

The `OracleAQMessageState` enumeration type identifies the state of the message at the time of dequeue.

Table 12-33 lists all the `OracleAQMessageState` enumeration values with a description of each enumerated value.

Table 12-33 OracleAQMessageState Members

Member Name	Description
`Expired`	Indicates that the message has been moved to the exception queue.
`Processed`	Indicates that the message has been processed and retained.
`Ready`	Indicates that the message is ready to be processed.
`Waiting`	Indicates that the message delay has not been reached.

Requirements

Provider	ODP.NET, Unmanaged Driver
Assembly	`Oracle.DataAccess.dll`
Namespace	`Oracle.DataAccess.Client`
.NET Framework	3.5, 4.5, 4.6

12.11 OracleAQMessageType Enumeration

The `OracleAQMessageType` enumeration type specifies the message payload type.

Table 12-34 lists all the `OracleAQMessageType` enumeration values with a description of each enumerated value.

Table 12-34 OracleAQMessageType Members

Member Name	Description
`Raw`	Indicates the Raw message type. The data type of the payload must be either `OracleBinary` or `byte[]` to enqueue the message.
`Udt`	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.
`Xml`	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.

Requirements

Provider	ODP.NET, Unmanaged Driver
Assembly	`Oracle.DataAccess.dll`
Namespace	`Oracle.DataAccess.Client`

Provider	ODP.NET, Unmanaged Driver
.NET Framework	3.5, 4.5, 4.6

12.12 OracleAQNavigationMode Enumeration

Table 12-35 lists all the `OracleAQNavigationMode` enumeration values with a description of each enumerated value.

Table 12-35 OracleAQNavigationMode Members

Member Name	Description
`FirstMessage`	Retrieves the first message that is available and matches the search criteria. This resets the position to the beginning of the queue.
`FirstMessageMultiGroup`	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.
`NextMessage`	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.
`NextMessageMultiGroup`	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.
`NextTransaction`	Skips the remainder of the current transaction group (if any) and retrieves the first message of the next transaction group.

Requirements

Provider	ODP.NET, Unmanaged Driver
Assembly	`Oracle.DataAccess.dll`
Namespace	`Oracle.DataAccess.Client`
.NET Framework	3.5, 4.5, 4.6

12.13 OracleAQNotificationGroupingType Enumeration

The `OracleAQNotificationGroupingType` enumeration type specifies the notification grouping type.

Table 12-36 lists all the `OracleAQNotificationGroupingType` enumeration values with a description of each enumerated value.

Table 12-36 OracleAQNotificationGroupingType Members

Member Name	Description
`Last`	Indicates that only details of the last message in the notification group are provided.
`Summary`	Indicates that the `Summary` of all messages in the notification group is provided.

Requirements

Provider	ODP.NET, Unmanaged Driver
Assembly	`Oracle.DataAccess.dll`
Namespace	`Oracle.DataAccess.Client`
.NET Framework	3.5, 4.5, 4.6

12.14 OracleAQNotificationType Enumeration

The `OracleAQNotificationType` enumeration type specifies the notification type of the received notification.

Table 12-37 lists all the `OracleAQNotificationType` enumeration values with a description of each enumerated value.

Table 12-37 OracleAQNotificationType Members

Member Name	Description
`Group`	Indicates that the received notification is a grouping notification.
`Regular`	Indicates that the received notification is a regular notification.
`Timeout`	Indicates that the received notification is raised due to a timeout.

Requirements

Provider	ODP.NET, Unmanaged Driver
Assembly	`Oracle.DataAccess.dll`
Namespace	`Oracle.DataAccess.Client`
.NET Framework	3.5, 4.5, 4.6

12.15 OracleAQVisibilityMode Enumeration

Table 12-38 lists all the `OracleAQVisibilityMode` enumeration values with a description of each enumerated value.

Table 12-38 OracleAQVisibilityMode Members

Member Name	Description
`Immediate`	Indicates that the enqueue or dequeue operation is not part of the current transaction. The operation constitutes a transaction of its own.
`OnCommit`	Indicates that the enqueue or dequeue operation is part of the current transaction. This is the default case.

Requirements

Provider	**ODP.NET, Unmanaged Driver**
Assembly	`Oracle.DataAccess.dll`
Namespace	`Oracle.DataAccess.Client`
.NET Framework	3.5, 4.5, 4.6

13

Oracle Data Provider for .NET Types Classes

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.

13.1 OracleBFile Class

An `OracleBFile` is an object that has a reference to `BFILE` data. It provides methods for performing operations on `BFILE`s.

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
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver
Assembly	`Oracle.DataAccess.dll`	`Oracle.ManagedDataAccess.dll`

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver
Namespace	`Oracle.DataAccess.Types`	`Oracle.ManagedDataAccess.Types`
.NET Framework	3.5, 4.5, 4.6	4.5, 4.6

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 constr = "User Id=scott;Password=tiger;Data Source=oracle";
    OracleConnection con = new OracleConnection(constr);
    con.Open();

    OracleBFile bFile = new OracleBFile(con, "MYDIR", "MyFile.txt");

    // Open the OracleBFile
    bFile.OpenFile();

    // Read 7 bytes into readBuffer, starting at buffer offset 0
    byte[] readBuffer = new byte[7];
    int bytesRead = bFile.Read(readBuffer, 0, 7);

    // Prints "bytesRead  = 7"
    Console.WriteLine("bytesRead  = " + bytesRead);

    // Prints "readBuffer = 65666768656667"
    Console.Write("readBuffer = ");
    for(int index = 0; index <  readBuffer.Length; index++)
    {
      Console.Write(readBuffer[index]);
    }
```

```
        Console.WriteLine();

        // Search for the 2nd occurrence of a byte pattern {66,67}
        // starting from byte offset 1 in the OracleBFile
        byte[] pattern = new byte[2] {66, 67};
        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();
    }
}
```

13.1.1 OracleBFile Members

`OracleBFile` members are listed in the following tables.

OracleBFile Constructors

`OracleBFile` constructors are listed in Table 13-1.

Table 13-1 OracleBFile Constructors

Constructor	Description
OracleBFile Constructors	Creates an instance of the `OracleBFile` class (Overloaded)

OracleBFile Static Fields

`OracleBFile` static fields are listed in Table 13-2.

Table 13-2 OracleBFile Static Fields

Field	Description
MaxSize	The static field holds the maximum number of bytes a `BFILE` can hold, which is 4,294,967,295 (2^32 - 1) bytes
Null	Represents a null value that can be assigned to the value of an `OracleBFile` instance

OracleBFile Static Methods

`OracleBFile` static methods are listed in Table 13-3.

Table 13-3 OracleBFile Static Methods

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

OracleBFile Instance Properties

`OracleBFile` instance properties are listed in Table 13-4.

Table 13-4 OracleBFile Instance Properties

Properties	Description
CanRead	Indicates whether or not the LOB stream can be read
CanSeek	Indicates whether or not forward and backward seek operations can be performed
CanWrite	Indicates whether or not the LOB object supports writing
Connection	Indicates the connection used to read from a `BFILE`
DirectoryName	Indicates the directory alias of the `BFILE`
FileExists	Indicates whether or not the specified `BFILE` exists
FileName	Indicates the name of the `BFILE`
IsEmpty	Indicates whether the `BFILE` is empty or not
IsNull	Indicates whether or not the current instance has a null value
IsOpen	Indicates whether the `BFILE` has been opened by this instance or not
Length	Indicates the size of the `BFILE` data in bytes
Position	Indicates the current read position in the LOB stream
Value	Returns the data, starting from the first byte in `BFILE`, as a byte array

OracleBFile Instance Methods

`OracleBFile` instance methods are listed in Table 13-5.

Table 13-5 OracleBFile Instance Methods

Methods	Description
`BeginRead`	Inherited from `System.IO.Stream`
`BeginWrite`	*Not Supported*
Clone	Creates a copy of an `OracleBFile` object
Close	Closes the current stream and releases any resources associated with the stream

Table 13-5 (Cont.) OracleBFile Instance Methods

Methods	Description
CloseFile	Closes the `BFILE` referenced by the current `BFILE` instance
Compare	Compares data referenced by the two `OracleBFiles`
`CreateObjRef`	Inherited from `System.MarshalByRefObject`
CopyTo	Copies data as specified (Overloaded)
Dispose	Releases resources allocated by this object
`EndRead`	Inherited from `System.IO.Stream`
`EndWrite`	*Not Supported*
`Equals`	Inherited from `System.Object` (Overloaded)
Flush	*Not Supported*
FlushAsync	*Not Supported*
`GetHashCode`	Inherited from `System.Object`
`GetLifetimeService`	Inherited from `System.MarshalByRefObject`
`GetType`	Inherited from `System.Object`
`InitializeLifetimeService`	Inherited from `System.MarshalByRefObject`
IsEqual	Compares the LOB references
OpenFile	Opens the `BFILE` specified by the `FileName` and `DirectoryName`
Read	Reads a specified amount of bytes from the `OracleBFile` instance and populates the `buffer`
`ReadByte`	Inherited from `System.IO.Stream`
Search	Searches for a binary pattern in the current instance of an `OracleBFile`
Seek	Sets the position on the current LOB stream
SetLength	*Not Supported*
`ToString`	Inherited from `System.Object`
Write	*Not Supported*
`WriteByte`	*Not Supported*

13.1.2 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`.

13.1.2.1 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.

13.1.2.2 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.

13.1.3 OracleBFile Static Fields

OracleBFile static fields are listed in Table 13-6.

Table 13-6 OracleBFile Static Fields

Field	Description
MaxSize	The static field holds the maximum number of bytes a BFILE can hold, which is 4,294,967,295 (2^32 - 1) bytes
Null	Represents a null value that can be assigned to the value of an OracleBFile instance

13.1.3.1 MaxSize

This static field holds the maximum number of bytes a BFILE can hold, which is 4,294,967,295 (2^32 - 1) bytes.

Declaration

```
// C#
public static readonly Int64 MaxSize = 4294967295;
```

Remarks

This field is useful in code that checks whether or not the operation exceeds the maximum length allowed.

13.1.3.2 Null

This static field represents a null value that can be assigned to the value of an OracleBFile instance.

Declaration

```
// C#
public static readonly OracleBFile Null;
```

13.1.4 OracleBFile Static Methods

OracleBFile static methods are listed in Table 13-7.

Table 13-7 OracleBFile Static Methods

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

13.1.5 OracleBFile Instance Properties

`OracleBFile` instance properties are listed in Table 13-8.

Table 13-8 OracleBFile Instance Properties

Properties	Description
CanRead	Indicates whether or not the LOB stream can be read
CanSeek	Indicates whether or not forward and backward seek operations can be performed
CanWrite	Indicates whether or not the LOB object supports writing
Connection	Indicates the connection used to read from a `BFILE`
DirectoryName	Indicates the directory alias of the `BFILE`
FileExists	Indicates whether or not the specified `BFILE` exists
FileName	Indicates the name of the `BFILE`
IsEmpty	Indicates whether the `BFILE` is empty or not
IsNull	Indicates whether or not the current instance has a null value
IsOpen	Indicates whether the `BFILE` has been opened by this instance or not
Length	Indicates the size of the `BFILE` data in bytes
Position	Indicates the current read position in the LOB stream
Value	Returns the data, starting from the first byte in `BFILE`, as a byte array

13.1.5.1 CanRead

Overrides `Stream`

This instance property indicates whether or not the LOB stream can be read.

Declaration

```
// C#
public override bool CanRead{get;}
```

Property Value

If the LOB stream can be read, returns `true`; otherwise, returns `false`.

13.1.5.2 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`.

13.1.5.3 CanWrite

Overrides `Stream`

This instance property indicates whether or not the LOB object supports writing.

Declaration

```
// C#
public override bool CanWrite{get;}
```

Property Value

`BFILE` is read only.

Remarks

`BFILE` is read-only, therefore, the boolean value is always `false`.

13.1.5.4 Connection

This instance property indicates the connection used to read from a `BFILE`.

Declaration

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

Property Value

An object of `OracleConnection`.

Exceptions

`ObjectDisposedException` - The object is already disposed.

13.1.5.5 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.

13.1.5.6 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.

13.1.5.7 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.

13.1.5.8 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.

13.1.5.9 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`.

13.1.5.10 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`.

13.1.5.11 Length

Overrides `Stream`

This instance property indicates the size of the `BFILE` data in bytes.

Declaration

```
// C#
public override Int64 Length {get;}
```

Property Value

`Int64`

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

13.1.5.12 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.

13.1.5.13 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.

13.1.6 OracleBFile Instance Methods

`OracleBFile` instance methods are listed in Table 13-9.

Table 13-9 OracleBFile Instance Methods

Methods	Description
`BeginRead`	Inherited from `System.IO.Stream`
`BeginWrite`	*Not Supported*
Clone	Creates a copy of an `OracleBFile` object
Close	Closes the current stream and releases any resources associated with the stream
CloseFile	Closes the `BFILE` referenced by the current `BFILE` instance
Compare	Compares data referenced by the two `OracleBFiles`
`CreateObjRef`	Inherited from `System.MarshalByRefObject`
CopyTo	Copies data as specified (Overloaded)
Dispose	Releases resources allocated by this object
`EndRead`	Inherited from `System.IO.Stream`
`EndWrite`	*Not Supported*
`Equals`	Inherited from `System.Object` (Overloaded)
Flush	*Not Supported*
FlushAsync	*Not Supported*
`GetHashCode`	Inherited from `System.Object`
`GetLifetimeService`	Inherited from `System.MarshalByRefObject`
`GetType`	Inherited from `System.Object`

Table 13-9 (Cont.) OracleBFile Instance Methods

Methods	Description
`InitializeLifetimeService`	Inherited from `System.MarshalByRefObject`
IsEqual	Compares the LOB references
OpenFile	Opens the `BFILE` specified by the `FileName` and `DirectoryName`
Read	Reads a specified amount of bytes from the `OracleBFile` instance and populates the `buffer`
`ReadByte`	Inherited from `System.IO.Stream`
Search	Searches for a binary pattern in the current instance of an `OracleBFile`
Seek	Sets the position on the current LOB stream
SetLength	*Not Supported*
`ToString`	Inherited from `System.Object`
Write	*Not Supported*
`WriteByte`	*Not Supported*

13.1.6.1 Clone

This instance method creates a copy of an `OracleBFile` object.

Declaration

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

Return Value

An `OracleBFile` object.

Implements

`ICloneable`

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

Remarks

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

Example

```
// 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 above 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");

    // Open the OracleBFile
    bFile1.OpenFile();

    // 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();

    // Open the OracleBFile
    bFile2.OpenFile();

    // Prints "bFile2.Position = 1"
    Console.WriteLine("bFile2.Position = " + bFile2.Position);

    // Close the OracleBFile
    bFile1.CloseFile();

    bFile1.Close();
    bFile1.Dispose();

    // Close the Cloned OracleBFile
    bFile2.CloseFile();

    bFile2.Close();
    bFile2.Dispose();

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

13.1.6.2 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.

13.1.6.3 CloseFile

This instance method closes the `BFILE` referenced by the current `BFILE` instance.

Declaration

```
// C#
public void CloseFile();
```

Remarks

No error is returned if the `BFILE` exists, but is not opened.

13.1.6.4 Compare

This instance method compares data referenced by the two `OracleBFile`s.

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 65666768656667

    string constr = "User Id=scott;Password=tiger;Data Source=oracle";
    OracleConnection con = new OracleConnection(constr);
    con.Open();

    OracleBFile bFile1 = new OracleBFile(con, "MYDIR", "MyFile.txt");
    OracleBFile bFile2 = new OracleBFile(con, "MYDIR", "MyFile.txt");

    // Open the OracleBFiles
    bFile1.OpenFile();
    bFile2.OpenFile();

    // Compare 2 bytes from the 1st byte of bFile1 and
    // the 5th byte of bFile2 onwards
    int result = bFile1.Compare(1, bFile2, 5, 2);

    // Prints "result = 0" (Indicates the data is identical)
    Console.WriteLine("result = " + result);

    // Close the OracleBFiles
    bFile1.CloseFile();
    bFile2.CloseFile();

    bFile1.Close();
    bFile1.Dispose();

    bFile2.Close();
    bFile2.Dispose();

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

13.1.6.5 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.

13.1.6.6 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.

13.1.6.7 CopyTo(OracleBlob, Int64)

This instance method copies data from the current `OracleBFile` instance to the provided `OracleBlob` object with the specified destination offset.

Declaration

```
// C#
public Int64 CopyTo(OracleBlob obj, Int64 dst_offset);
```

Parameters

- *`obj`*

  The `OracleBlob` object to which the data is copied.
- *`dst_offset`*

  The offset (in bytes) at which the `OracleBlob` object is copied.

Return Value

The return value is the amount copied.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`ArgumentOutOfRangeException` - The *`dst_offset`* is less than `0`.

`InvalidOperationException` - This exception is thrown if any of the following conditions exist:

- The `OracleConnection` is not open or has been closed during the lifetime of the object.
- The LOB object parameter has a different connection than the object.

Remarks

If the *`dst_offset`* is beyond the end of the `OracleBlob` data, spaces are written into the `OracleBlob` until the *`dst_offset`* is met.

The offsets are `0`-based. No character conversion is performed by this operation.

The provided object and the current instance must be using the same connection; that is, the same `OracleConnection` object.

13.1.6.8 CopyTo(Int64, OracleBlob, Int64, Int64)

This instance method copies data from the current `OracleBFile` instance to the provided `OracleBlob` object with the specified source offset, destination offset, and character amounts.

Declaration

```
// C#
public Int64 CopyTo(Int64 src_offset,OracleBlob obj,Int64 dst_offset,
   Int64 amount);
```

Parameters

- *`src_offset`*

  The offset (in bytes) in the current instance, from which the data is read.
- *`obj`*

  An `OracleBlob` object to which the data is copied.
- *`dst_offset`*

  The offset (in bytes) to which the `OracleBlob` object is copied.
- *`amount`*

  The amount of data to be copied.

Return Value

The return value is the amount copied.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`ArgumentOutOfRangeException` - The *`src_offset`*, the *`dst_offset`*, or the *`amount`* is less than `0`.

`InvalidOperationException` - This exception is thrown if any of the following conditions exist:

- The `OracleConnection` is not open or has been closed during the lifetime of the object.
- The LOB object parameter has a different connection than the object.

Remarks

If the *`dst_offset`* is beyond the end of the `OracleBlob` data, spaces are written into the `OracleBlob` until the *`dst_offset`* is met.

The offsets are `0`-based. No character conversion is performed by this operation.

The provided object and the current instance must be using the same connection; that is, the same `OracleConnection` object.

13.1.6.9 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.

13.1.6.10 CopyTo(OracleClob, Int64)

This instance method copies data from the current `OracleBFile` instance to the provided `OracleClob` object with the specified destination offset.

Declaration

```
// C#
public Int64 CopyTo(OracleClob obj, Int64 dst_offset);
```

Parameters

- *`obj`*

  The `OracleClob` object that the data is copied to.

- *`dst_offset`*

  The offset (in characters) at which the `OracleClob` object is copied to.

Return Value

The amount copied.

Exceptions

Exceptions

`ObjectDisposedException` - The object is already disposed.

`ArgumentOutOfRangeException` - The *`dst_offset`* is less than `0`.

`InvalidOperationException` - This exception is thrown if any of the following conditions exist:

- The `OracleConnection` is not open or has been closed during the lifetime of the object.
- The LOB object parameter has a different connection than the object.

Remarks

If the *`dst_offset`* is beyond the end of the `OracleClob` data, spaces are written into the `OracleClob` until the *`dst_offset`* is met.

The offsets are `0`-based. No character conversion is performed by this operation.

The provided object and the current instance must be using the same connection, that is, the same `OracleConnection` object.

13.1.6.11 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.

13.1.6.12 Dispose

This instance method releases resources allocated by this object.

Declaration

```
// C#
public void Dispose();
```

Implements

`IDisposable`

Remarks

Although some properties can still be accessed, their values may not be accountable. Since resources are freed, method calls may lead to exceptions. The object cannot be reused after being disposed.

13.1.6.13 Flush

This method is not supported.

13.1.6.14 FlushAsync

This method is not supported.

13.1.6.15 IsEqual

This instance method compares the LOB references.

Declaration

```
// C#
public bool IsEqual(OracleBFile obj);
```

Parameters

- *obj*

  The provided `OracleBFile` object.

Return Value

Returns `true` if the current `OracleBFile` and the provided `OracleBFile` object refer to the same external LOB. Returns `false` otherwise.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

Remarks

Note that this method can return `true` even if the two `OracleBFile` objects return `false` for `==` or `Equals()` since two different `OracleBFile` instances can refer to the same external LOB.

The provided object and the current instance must be using the same connection; that is, the same `OracleConnection` object.

13.1.6.16 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.

13.1.6.17 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';

*/

// C#

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]);
      }
      Console.WriteLine();

      // Close the OracleBFile
      bFile.CloseFile();

      bFile.Close();
      bFile.Dispose();

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

13.1.6.18 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 65666768656667

    string constr = "User Id=scott;Password=tiger;Data Source=oracle";
    OracleConnection con = new OracleConnection(constr);
    con.Open();

    OracleBFile bFile = new OracleBFile(con, "MYDIR", "MyFile.txt");

    // Open the OracleBFile
    bFile.OpenFile();

    // Search for the 2nd occurrence of a byte pattern {66,67}
    // starting from byte offset 1 in the OracleBFile
    byte[] pattern = new byte[2] {66, 67};
    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();
  }
}
```

13.1.6.19 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"
    Console.Write("readBuffer      = ");
    for(int index = 0; index <  readBuffer.Length; index++)
    {
      Console.Write(readBuffer[index]);
    }
    Console.WriteLine();

    // Close the OracleBFile
    bFile.CloseFile();

    bFile.Close();
    bFile.Dispose();

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

13.1.6.20 SetLength

This method is not supported.

13.1.6.21 Write

This method is not supported.

13.2 OracleBlob Class

An `OracleBlob` object is an object that has a reference to `BLOB` data. It provides methods for performing operations on `BLOB`s.

Class Inheritance

```
System.Object

  System.MarshalByRefObject

    System.IO.Stream

      Oracle.DataAccess.Types.OracleBlob
```

Declaration

```
// C#
public sealed class OracleBlob : Stream, ICloneable, INullable
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver
Assembly	`Oracle.DataAccess.dll`	`Oracle.ManagedDataAccess.dll`
Namespace	`Oracle.DataAccess.Types`	`Oracle.ManagedDataAccess.Types`
.NET Framework	3.5, 4.5, 4.6	4.5, 4.6

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();
    }
  }
```

13.2.1 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

Constructor	Description
OracleBlob Constructors	Creates an instance of the `OracleBlob` class (Overloaded)

OracleBlob Static Fields

`OracleBlob` static fields are listed in Table 13-11.

Table 13-11 OracleBlob Static Fields

Field	Description
MaxSize	Holds the maximum number of bytes a `BLOB` can hold, which is 4,294,967,295 (2^32 - 1) bytes
Null	Represents a null value that can be assigned to the value of an `OracleBlob` instance

OracleBlob Static Methods

`OracleBlob` static methods are listed in Table 13-12.

Table 13-12 OracleBlob Static Methods

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

OracleBlob Instance Properties

`OracleBlob` instance properties are listed in Table 13-13.

Table 13-13 OracleBlob Instance Properties

Properties	Description
CanRead	Indicates whether or not the LOB stream can be read
CanSeek	Indicates whether or not forward and backward seek operations be performed
CanWrite	Indicates whether or not the LOB object supports writing
Connection	Indicates the `OracleConnection` that is used to retrieve and write `BLOB` data
IsEmpty	Indicates whether the `BLOB` is empty or not
IsInChunkWriteMode	Indicates whether or not the `BLOB` has been opened to defer index updates
IsNull	Indicates whether or not the current instance has a null value

Table 13-13 (Cont.) OracleBlob Instance Properties

Properties	Description
IsTemporary	Indicates whether or not the current instance is bound to a temporary `BLOB`
Length	Indicates the size of the `BLOB` data
OptimumChunkSize	Indicates the optimal data buffer length (or multiples thereof) that read and write operations should use to improve performance
Position	Indicates the current read or write position in the LOB stream
Value	Returns the data, starting from the first byte in `BLOB`, as a byte array

OracleBlob Instance Methods

`OracleBlob` instance methods are listed in Table 13-14.

Table 13-14 OracleBlob Instance Methods

Methods	Description
Append	Appends the supplied data to the current `OracleBlob` instance (Overloaded)
BeginChunkWrite	Opens the `BLOB`
`BeginRead`	Inherited from `System.IO.Stream`
`BeginWrite`	Inherited from `System.IO.Stream`
Clone	Creates a copy of an `OracleBlob` object
Close	Closes the current stream and releases any resources associated with it
Compare	Compares data referenced by the current instance and that of the supplied object
CopyTo	Copies from the current `OracleBlob` instance to an `OracleBlob` object (Overloaded)
`CreateObjRef`	Inherited from `System.MarshalByRefObject`
Dispose	Releases resources allocated by this `object`
EndChunkWrite	Closes the `BLOB` referenced by the current `OracleBlob` instance
`EndRead`	Inherited from `System.IO.Stream`
`EndWrite`	Inherited from `System.IO.Stream`
`Equals`	Inherited from `System.Object` (Overloaded)
Erase	Erases data (Overloaded)
Flush	*Not supported*
`GetHashCode`	Inherited from `System.Object`
`GetLifetimeService`	Inherited from `System.MarshalByRefObject`

Table 13-14 (Cont.) OracleBlob Instance Methods

Methods	Description
`GetType`	Inherited from `System.Object`
`InitializedLifetimeService`	Inherited from `System.MarshalByRefObject`
IsEqual	Compares the LOB data referenced by the two `OracleBlobs`
Read	Reads a specified amount of bytes from the ODP.NET LOB Type instance and populates the `buffer`
`ReadByte`	Inherited from `System.IO.Stream`
Search	Searches for a binary pattern in the current instance of an `OracleBlob`
Seek	Sets the position in the current LOB stream
SetLength	Trims or truncates the `BLOB` value to the specified length
`ToString`	Inherited from `System.Object`
Write	Writes the supplied buffer into the `OracleBlob`
`WriteByte`	Inherited from `System.IO.Stream`

See Also:

- "Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"
- OracleBlob Members

13.2.2 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.

13.2.2.1 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.

13.2.2.2 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.

13.2.3 OracleBlob Static Fields

`OracleBlob` static fields are listed in Table 13-15.

Table 13-15 OracleBlob Static Fields

Field	Description
MaxSize	Holds the maximum number of bytes a `BLOB` can hold, which is 4,294,967,295 (2^32 - 1) bytes
Null	Represents a null value that can be assigned to the value of an `OracleBlob` instance

13.2.3.1 MaxSize

The `MaxSize` field holds the maximum number of bytes a `BLOB` can hold, which is 4,294,967,295 (2^32 - 1) bytes.

Declaration

```
// C#
public static readonly Int64 MaxSize = 4294967295;
```

Remarks

This field can be useful in code that checks whether or not the operation exceeds the maximum length allowed.

13.2.3.2 Null

This static field represents a null value that can be assigned to the value of an `OracleBlob` instance.

Declaration

```
// C#
public static readonly OracleBlob Null;
```

13.2.4 OracleBlob Static Methods

`OracleBlob` static methods are listed in Table 13-16.

Table 13-16 OracleBlob Static Methods

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

13.2.5 OracleBlob Instance Properties

`OracleBlob` instance properties are listed in Table 13-17.

Table 13-17 OracleBlob Instance Properties

Properties	Description
CanRead	Indicates whether or not the LOB stream can be read
CanSeek	Indicates whether or not forward and backward seek operations be performed
CanWrite	Indicates whether or not the LOB object supports writing
Connection	Indicates the `OracleConnection` that is used to retrieve and write `BLOB` data
IsEmpty	Indicates whether the `BLOB` is empty or not
IsInChunkWriteMode	Indicates whether or not the `BLOB` has been opened to defer index updates
IsNull	Indicates whether or not the current instance has a null value
IsTemporary	Indicates whether or not the current instance is bound to a temporary `BLOB`
Length	Indicates the size of the `BLOB` data
OptimumChunkSize	Indicates the optimal data buffer length (or multiples thereof) that read and write operations should use to improve performance
Position	Indicates the current read or write position in the LOB stream
Value	Returns the data, starting from the first byte in `BLOB`, as a byte array

13.2.5.1 CanRead

Overrides `Stream`

This instance property indicates whether or not the LOB stream can be read.

Declaration

```
// C#
public override bool CanRead{get;}
```

Property Value

If the LOB stream can be read, returns `true`; otherwise, returns `false`.

13.2.5.2 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`.

13.2.5.3 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`.

13.2.5.4 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.

13.2.5.5 IsEmpty

This instance property indicates whether the `BLOB` is empty or not.

Declaration

```
// C#
public bool IsEmpty {get;}
```

Property Value

A `bool` that indicates whether or not the `BLOB` is empty.

Exceptions

`ObjectDisposedException` - The object is already disposed.

13.2.5.6 IsInChunkWriteMode

This instance property indicates whether or not the `BLOB` has been opened to defer index updates.

Declaration

```
// C#
public bool IsInChunkWriteMode{get;}
```

Property Value

If the `BLOB` has been opened, returns `true`; otherwise, returns `false`.

13.2.5.7 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`.

13.2.5.8 IsTemporary

This instance property indicates whether or not the current instance is bound to a temporary `BLOB`.

Declaration

```
// C#
public bool IsTemporary {get;}
```

Property Value

`bool`

13.2.5.9 Length

Overrides `Stream`

This instance property indicates the size of the `BLOB` data in bytes.

Declaration

```
// C#
public override Int64 Length {get;}
```

Property Value

A number indicating the size of the `BLOB` data in bytes.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

13.2.5.10 OptimumChunkSize

This instance property indicates the optimal data buffer length (or multiples thereof) that read and write operations should use to improve performance.

Declaration

```
// C#
public int OptimumChunkSize{get;}
```

Property Value

A number representing the minimum bytes to retrieve or send.

Exceptions

`ObjectDisposedException` - The object is already disposed.

13.2.5.11 Position

Overrides `Stream`

This instance property indicates the current read or write position in the LOB stream.

Declaration

```
// C#
public override Int64 Position{get; set;}
```

Property Value

An `Int64` that indicates the read or write position.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

`ArgumentOutOfRangeException` - The `Position` is less than `0`.

13.2.5.12 Value

This instance property returns the data, starting from the first byte in the BLOB, as a byte array.

Declaration

```
// C#
public Byte[] Value{get;}
```

Property Value

A byte array.

Exceptions

ObjectDisposedException - The object is already disposed.

InvalidOperationException - The OracleConnection is not open or has been closed during the lifetime of the object.

ArgumentOutOfRangeException - The Value is less than 0.

Remarks

The value of Position is not used or changed by using this property. 2 GB is the maximum byte array length that can be returned by this property.

13.2.6 OracleBlob Instance Methods

OracleBlob instance methods are listed in Table 13-18.

Table 13-18 OracleBlob Instance Methods

Methods	Description
Append	Appends the supplied data to the current OracleBlob instance (Overloaded)
BeginChunkWrite	Opens the BLOB
BeginRead	Inherited from System.IO.Stream
BeginWrite	Inherited from System.IO.Stream
Clone	Creates a copy of an OracleBlob object
Close	Closes the current stream and releases any resources associated with it
Compare	Compares data referenced by the current instance and that of the supplied object
CopyTo	Copies from the current OracleBlob instance to an OracleBlob object (Overloaded)
CreateObjRef	Inherited from System.MarshalByRefObject
Dispose	Releases resources allocated by this object

Table 13-18 (Cont.) OracleBlob Instance Methods

Methods	Description
EndChunkWrite	Closes the `BLOB` referenced by the current `OracleBlob` instance
`EndRead`	Inherited from `System.IO.Stream`
`EndWrite`	Inherited from `System.IO.Stream`
`Equals`	Inherited from `System.Object` (Overloaded)
Erase	Erases data (Overloaded)
Flush	*Not supported*
`GetHashCode`	Inherited from `System.Object`
`GetLifetimeService`	Inherited from `System.MarshalByRefObject`
`GetType`	Inherited from `System.Object`
`InitializedLifetimeService`	Inherited from `System.MarshalByRefObject`
IsEqual	Compares the LOB data referenced by the two `OracleBlobs`
Read	Reads a specified amount of bytes from the ODP.NET LOB Type instance and populates the `buffer`
`ReadByte`	Inherited from `System.IO.Stream`
Search	Searches for a binary pattern in the current instance of an `OracleBlob`
Seek	Sets the position in the current LOB stream
SetLength	Trims or truncates the `BLOB` value to the specified length
`ToString`	Inherited from `System.Object`
Write	Writes the supplied buffer into the `OracleBlob`
`WriteByte`	Inherited from `System.IO.Stream`

13.2.6.1 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.

13.2.6.2 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.

13.2.6.3 Append(byte[ ], int, int)

This instance method appends data from the supplied byte array buffer to the end of the current `OracleBlob` instance.

Declaration

```
// C#
public void Append(byte[] buffer, int offset, int count);
```

Parameters

- *`buffer`*

  An array of bytes.

- *`offset`*

  The zero-based byte offset in the buffer from which data is read.

- *`count`*

  The number of bytes to be appended.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

Example

```
// C#

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 = ");
    for(int index = 0; index < appendBuffer.Length; index++)
    {
      Console.Write(appendBuffer[index]);
    }
    Console.WriteLine();

    blob.Close();
    blob.Dispose();

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

13.2.6.4 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.

13.2.6.5 Clone

This instance method creates a copy of an `OracleBlob` object.

Declaration

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

Return Value

An `OracleBlob` object.

Implements

`ICloneable`

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

Remarks

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

Example

```
// C#

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

class CloneSample
{
  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();
  }
}
```

13.2.6.6 Close

Overrides `Stream`

This instance method closes the current stream and releases any resources associated with it.

Declaration

```
// C#
public override void Close();
```

13.2.6.7 Compare

This instance method compares data referenced by the current instance and that of the supplied object.

Declaration

```
// C#
public int Compare(Int64 src_offset, OracleBlob obj, Int64 dst_offset,
   Int64 amount);
```

Parameters

- *`src_offset`*

  The comparison starting point (in bytes) for the current instance.
- *`obj`*

  The provided `OracleBlob` object.
- *`dst_offset`*

  The comparison starting point (in bytes) for the provided `OracleBlob`.
- *`amount`*

  The number of bytes to compare.

Return Value

Returns a value that is:

- Less than zero: if the data referenced by the current instance is less than that of the supplied instance
- Zero: if both objects reference the same data
- Greater than zero: if the data referenced by the current instance is greater than that of the supplied instance

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - The parameter has a different connection than the object, `OracleConnection` is not opened, or `OracleConnection` has been reopened.

`ArgumentOutOfRangeException` - The *`src_offset`*, the *`dst_offset`*, or the *`amount`* parameter is less than `0`.

Remarks

The provided object and the current instance must be using the same connection, that is, the same `OracleConnection` object.

13.2.6.8 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.

13.2.6.9 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.

13.2.6.10 CopyTo(OracleBlob, Int64)

This instance method copies data from the current OracleBlob instance to the provided OracleBlob object with the specified destination offset.

Declaration

```
// C#
public Int64 CopyTo(OracleBlob obj, Int64 dst_offset);
```

Parameters

- *obj*

  The OracleBlob object to which the data is copied.
- *dst_offset*

  The offset (in bytes) at which the OracleBlob object is copied.

Return Value

The return value is the amount copied.

Exceptions

ObjectDisposedException - The object is already disposed.

ArgumentOutOfRangeException - The *dst_offset* is less than 0.

InvalidOperationException - This exception is thrown if any of the following conditions exist:

- The `OracleConnection` is not open or has been closed during the lifetime of the object.
- The LOB object parameter has a different connection than the object.

Remarks

If the *`dst_offset`* is beyond the end of the `OracleBlob` data, spaces are written into the `OracleBlob` until the *`dst_offset`* is met.

The offsets are `0`-based. No character conversion is performed by this operation.

The provided object and the current instance must be using the same connection; that is, the same `OracleConnection` object.

13.2.6.11 CopyTo(Int64, OracleBlob, Int64, Int64)

This instance method copies data from the current `OracleBlob` instance to the provided `OracleBlob` object with the specified source offset, destination offset, and character amounts.

Declaration

```
// C#
public Int64 CopyTo(Int64 src_offset,OracleBlob obj,Int64 dst_offset,
   Int64 amount);
```

Parameters

- *`src_offset`*

  The offset (in bytes) in the current instance, from which the data is read.
- *`obj`*

  The `OracleBlob` object to which the data is copied.
- *`dst_offset`*

  The offset (in bytes) at which the `OracleBlob` object is copied.
- *`amount`*

  The amount of data to be copied.

Return Value

The return value is the amount copied.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - The parameter has a different connection than the object, `OracleConnection` is not opened, or `OracleConnection` has been reopened.

`ArgumentOutOfRangeException` - The *`src_offset`*, the *`dst_offset`*, or the *`amount`* parameter is less than `0`.

Remarks

If the *`dst_offset`* is beyond the end of the `OracleBlob` data, spaces are written into the `OracleBlob` until the *`dst_offset`* is met.

The offsets are `0`-based. No character conversion is performed by this operation.

The provided object and the current instance must be using the same connection; that is, the same `OracleConnection` object.

Example

```
// C#

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();
  }
}
```

13.2.6.12 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.

13.2.6.13 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.

13.2.6.14 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.

13.2.6.15 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.

13.2.6.16 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.

13.2.6.17 Flush

This method is not supported.

13.2.6.18 IsEqual

This instance method compares the LOB data referenced by the two `OracleBlob`s.

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.

13.2.6.19 Read

Overrides `Stream`

This instance method reads a specified amount of bytes from the ODP.NET LOB instance and populates the `buffer`.

Declaration

```
// C#
public override int Read(byte[] buffer, int offset, int count);
```

Parameters

- *`buffer`*

  The byte array buffer to be populated.

- *`offset`*

  The starting offset (in bytes) at which the buffer is populated.

- *`count`*

The amount of bytes to read.

Return Value

The return value indicates the number of bytes read from the LOB.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

`ArgumentOutOfRangeException` - This exception is thrown if any of the following conditions exist:

- The *`offset`* or the *`count`* parameter is less than `0`.
- The *`offset`* is greater than or equal to the *`buffer`*`.Length`.
- The *`offset`* and the *`count`* together are greater than the *`buffer`*`.Length`.

Remarks

The LOB data is read starting from the position specified by the `Position` property.

Example

```
// C#

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

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

    OracleBlob blob = new OracleBlob(con);

    // Write 3 bytes, starting at buffer offset 1
    byte[] writeBuffer = new byte[4] {1, 2, 3, 4};
    blob.Write(writeBuffer, 1, 3);

    // Reset the Position for Read
    blob.Position = 1;

    // Read 2 bytes into buffer starting at buffer offset 1
    byte[] readBuffer = new byte[4];
    int bytesRead = blob.Read(readBuffer, 1, 2);

    // Prints "bytesRead  = 2"
    Console.WriteLine("bytesRead  = " + bytesRead);

    // Prints "readBuffer = 0340"
    Console.Write("readBuffer = ");
    for(int index = 0; index <  readBuffer.Length; index++)
    {
      Console.Write(readBuffer[index]);
    }
    Console.WriteLine();

    blob.Close();
    blob.Dispose();

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

13.2.6.20 Search

This instance method searches for a binary pattern in the current instance of an `OracleBlob`.

Declaration

```
// C#
public Int64 Search(byte[] val, int64 offset, int64 nth);
```

Parameters

- *`val`*

  The binary pattern being searched for.
- *`offset`*

  The `0`-based offset (in bytes) starting from which the `OracleBlob` is searched.
- *`nth`*

  The specific occurrence (1-based) of the match for which the absolute offset (in bytes) is returned.

Return Value

Returns the absolute *`offset`* of the start of the matched pattern (in bytes) for the *`nth`* occurrence of the match. Otherwise, `0` is returned.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

`ArgumentOutOfRangeException` - This exception is thrown if any of the following conditions exist:

- The *`offset`* is less than `0`.
- The *`nth`* is less than or equal to `0`.
- The *`val`*`.Length` is greater than `16383`.
- The *`nth`* is greater than or equal to `OracleBlob.MaxSize`.
- The *`offset`* is greater than or equal to `OracleBlob.MaxSize`.

Remarks

The limit of the search pattern is 16383 bytes.

Example

```
// C#

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();
  }
}
```

13.2.6.21 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 `Int64` for the position.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

Remarks

If *`offset`* is negative, the new position precedes the position specified by *`origin`* by the number of bytes specified by `offset`.

If *`offset`* is zero, the new position is the position specified by *`origin`*.

If *`offset`* is positive, the new position follows the position specified by *`origin`* by the number of bytes specified by *`offset`*.

`SeekOrigin.Begin` specifies the beginning of a stream.

`SeekOrigin.Current` specifies the current position within a stream.

`SeekOrigin.End` specifies the end of a stream.

13.2.6.22 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`.

13.2.6.23 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();
    }
  }
```

13.3 OracleClob Class

An `OracleClob` is an object that has a reference to `CLOB` data. It provides methods for performing operations on `CLOB`s.

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
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver
Assembly	`Oracle.DataAccess.dll`	`Oracle.ManagedDataAccess.dll`
Namespace	`Oracle.DataAccess.Types`	`Oracle.ManagedDataAccess.Types`
.NET Framework	3.5, 4.5, 4.6	4.5, 4.6

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();
  }
}
```

13.3.1 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

Constructor	Description
OracleClob Constructors	Creates an instance of the `OracleClob` class bound to a temporary `CLOB` (Overloaded)

OracleClob Static Fields

`OracleClob` static fields are listed in Table 13-20.

Table 13-20 OracleClob Static Fields

Field	Description
MaxSize	Holds the maximum number of bytes a `CLOB` can hold, which is 4,294,967,295 (2^32 - 1) bytes
Null	Represents a null value that can be assigned to the value of an `OracleClob` instance

OracleClob Static Methods

`OracleClob` static methods are listed in Table 13-21.

Table 13-21 OracleClob Static Methods

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

OracleClob Instance Properties

`OracleClob` instance properties are listed in Table 13-22.

Table 13-22 OracleClob Instance Properties

Properties	Description
CanRead	Indicates whether or not the LOB stream can be read
CanSeek	Indicates whether or not forward and backward seek operations can be performed
CanWrite	Indicates whether or not the LOB stream can be written
Connection	Indicates the `OracleConnection` that is used to retrieve and write `CLOB` data
IsEmpty	Indicates whether the `CLOB` is `empty` or not
IsInChunkWriteMode	Indicates whether or not the `CLOB` has been opened
IsNClob	Indicates whether or not the `OracleClob` object represents an `NCLOB`.
IsNull	Indicates whether or not the current instance has a null value
IsTemporary	Indicates whether or not the current instance is bound to a temporary `CLOB`
Length	Indicates the size of the `CLOB` data in bytes
OptimumChunkSize	Indicates the minimum number of bytes to retrieve or send from the database during a read or write operation
Position	Indicates the current read or write position in the LOB stream in bytes
Value	Returns the data, starting from the first character in the `CLOB` or `NCLOB`, as a string

OracleClob Instance Methods

The `OracleClob` instance methods are listed in Table 13-23.

Table 13-23 OracleClob Instance Methods

Methods	Description
Append	Appends data to the current `OracleClob` instance (Overloaded)
BeginChunkWrite	Opens the `CLOB`
`BeginRead`	Inherited from `System.IO.Stream`
`BeginWrite`	Inherited from `System.IO.Stream`
Clone	Creates a copy of an `OracleClob` object
Close	Closes the current stream and releases resources associated with it
Compare	Compares data referenced by the current instance to that of the supplied object
CopyTo	Copies the data to an `OracleClob` (Overloaded)
`CreateObjRef`	Inherited from `System.MarshalByRefObject`
Dispose	Releases resources allocated by this object
EndChunkWrite	Closes the `CLOB` referenced by the current `OracleClob` instance
`EndRead`	Inherited from `System.IO.Stream`
`EndWrite`	Inherited from `System.IO.Stream`
`Equals`	Inherited from `System.Object` (Overloaded)
Erase	Erases the specified `amount` of data (Overloaded)
Flush	*Not supported*
GetHashCode	Returns a hash code for the current instance
`GetLifetimeService`	Inherited from `System.MarshalByRefObject`
`GetType`	Inherited from `System.Object`
`InitializeLifetimeService`	Inherited from `System.MarshalByRefObject`
IsEqual	Compares the LOB data referenced by two `OracleClobs`
Read	Reads from the current instance (Overloaded)
`ReadByte`	Inherited from `System.IO.Stream`
Search	Searches for a character pattern in the current instance of `OracleClob` (Overloaded)
Seek	Sets the position in the current LOB stream
SetLength	Trims or truncates the `CLOB` value
`ToString`	Inherited from `System.Object`
Write	Writes the provided `buffer` into the `OracleClob` (Overloaded)
`WriteByte`	Inherited from `System.IO.Stream`

See Also:

- "Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"
- OracleClob Class

13.3.2 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`.

13.3.2.1 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.

13.3.2.2 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.

13.3.3 OracleClob Static Fields

`OracleClob` static fields are listed in Table 13-24.

Table 13-24 OracleClob Static Fields

Field	Description
MaxSize	Holds the maximum number of bytes a `CLOB` can hold, which is 4,294,967,295 (2^32 - 1) bytes
Null	Represents a null value that can be assigned to the value of an `OracleClob` instance

13.3.3.1 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.

13.3.3.2 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;
```

13.3.4 OracleClob Static Methods

`OracleClob` static methods are listed in Table 13-25.

Table 13-25 OracleClob Static Methods

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

13.3.5 OracleClob Instance Properties

`OracleClob` instance properties are listed in Table 13-26.

Table 13-26 OracleClob Instance Properties

Properties	Description
CanRead	Indicates whether or not the LOB stream can be read
CanSeek	Indicates whether or not forward and backward seek operations can be performed
CanWrite	Indicates whether or not the LOB stream can be written
Connection	Indicates the `OracleConnection` that is used to retrieve and write `CLOB` data
IsEmpty	Indicates whether the `CLOB` is `empty` or not
IsInChunkWriteMode	Indicates whether or not the `CLOB` has been opened
IsNClob	Indicates whether or not the `OracleClob` object represents an `NCLOB`.
IsNull	Indicates whether or not the current instance has a null value

Table 13-26 (Cont.) OracleClob Instance Properties

Properties	Description
IsTemporary	Indicates whether or not the current instance is bound to a temporary `CLOB`
Length	Indicates the size of the `CLOB` data in bytes
OptimumChunkSize	Indicates the minimum number of bytes to retrieve or send from the database during a read or write operation
Position	Indicates the current read or write position in the LOB stream in bytes
Value	Returns the data, starting from the first character in the `CLOB` or `NCLOB`, as a string

13.3.5.1 CanRead

Overrides `Stream`

This instance property indicates whether or not the LOB stream can be read.

Declaration

```
// C#
public override bool CanRead{get;}
```

Property Value

If the LOB stream can be read, returns `true`; otherwise, returns `false`.

13.3.5.2 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`.

13.3.5.3 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`.

13.3.5.4 Connection

This instance property indicates the `OracleConnection` that is used to retrieve and write `CLOB` data.

Declaration

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

Property Value

An `OracleConnection`.

Exceptions

`ObjectDisposedException` - The object is already disposed.

13.3.5.5 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.

13.3.5.6 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`.

13.3.5.7 IsNClob

This instance property indicates whether or not the `OracleClob` object represents an `NClob`.

Declaration

```
// C#
public bool IsNClob {get;}
```

Property Value

A `bool`.

13.3.5.8 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`.

13.3.5.9 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`.

13.3.5.10 Length

Overrides `Stream`

This instance property indicates the size of the `CLOB` data in bytes.

Declaration

```
// C#
public override Int64 Length {get;}
```

Property Value

An `Int64` that indicates the size of the `CLOB` in bytes.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

13.3.5.11 OptimumChunkSize

This instance property indicates the minimum number of bytes to retrieve or send from the database during a read or write operation.

Declaration

```
// C#
public int OptimumChunkSize{get;}
```

Property Value

A number representing the minimum bytes to retrieve or send.

Exceptions

`ObjectDisposedException` - The object is already disposed.

13.3.5.12 Position

Overrides `Stream`

This instance property indicates the current read or write position in the LOB stream in bytes.

Declaration

```
// C#
public override Int64 Position{get; set;}
```

Property Value

An `Int64` that indicates the read or write position.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

`ArgumentOutOfRangeException` - The `Position` is less than `0`.

13.3.5.13 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.

13.3.6 OracleClob Instance Methods

The `OracleClob` instance methods are listed in Table 13-27.

Table 13-27 OracleClob Instance Methods

Methods	Description
Append	Appends data to the current `OracleClob` instance (Overloaded)
BeginChunkWrite	Opens the `CLOB`
`BeginRead`	Inherited from `System.IO.Stream`
`BeginWrite`	Inherited from `System.IO.Stream`
Clone	Creates a copy of an `OracleClob` object
Close	Closes the current stream and releases resources associated with it
Compare	Compares data referenced by the current instance to that of the supplied object
CopyTo	Copies the data to an `OracleClob` (Overloaded)
`CreateObjRef`	Inherited from `System.MarshalByRefObject`
Dispose	Releases resources allocated by this object

Table 13-27 (Cont.) OracleClob Instance Methods

Methods	Description
EndChunkWrite	Closes the CLOB referenced by the current OracleClob instance
EndRead	Inherited from System.IO.Stream
EndWrite	Inherited from System.IO.Stream
Equals	Inherited from System.Object (Overloaded)
Erase	Erases the specified amount of data (Overloaded)
Flush	*Not supported*
GetHashCode	Returns a hash code for the current instance
GetLifetimeService	Inherited from System.MarshalByRefObject
GetType	Inherited from System.Object
InitializeLifetimeService	Inherited from System.MarshalByRefObject
IsEqual	Compares the LOB data referenced by two OracleClobs
Read	Reads from the current instance (Overloaded)
ReadByte	Inherited from System.IO.Stream
Search	Searches for a character pattern in the current instance of OracleClob (Overloaded)
Seek	Sets the position in the current LOB stream
SetLength	Trims or truncates the CLOB value
ToString	Inherited from System.Object
Write	Writes the provided buffer into the OracleClob (Overloaded)
WriteByte	Inherited from System.IO.Stream

13.3.6.1 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.

13.3.6.2 Append(OracleClob)

This instance method appends the `CLOB` data referenced by the provided `OracleClob` object to the current `OracleClob` instance.

Declaration

```
// C#
public void Append(OracleClob obj);
```

Parameters

- *`obj`*

  An `OracleClob` object.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - The parameter has a different connection than the object, `OracleConnection` is not opened, or `OracleConnection` has been reopened.

Remarks

No character set conversions are made.

The provided object and the current instance must be using the same connection; that is, the same `OracleConnection` object.

13.3.6.3 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.

13.3.6.4 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();
  }
}
```

13.3.6.5 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.

13.3.6.6 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();
  }
}
```

13.3.6.7 Close

Overrides `Stream`

This instance method closes the current stream and releases resources associated with it.

Declaration

```
// C#
public override void Close();
```

13.3.6.8 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.

13.3.6.9 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.

13.3.6.10 CopyTo(OracleClob)

This instance method copies data from the current instance to the provided `OracleClob` object.

Declaration

```
// C#
public Int64 CopyTo(OracleClob obj);
```

Parameters

- *obj*

  The `OracleClob` object to which the data is copied.

Return Value

The return value is the amount copied.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - This exception is thrown if any of the following conditions exist:

- The `OracleConnection` is not open or has been closed during the lifetime of the object.
- The LOB object parameter has a different connection than the object.

Remarks

The provided object and the current instance must be using the same connection, that is, the same `OracleConnection` object.

13.3.6.11 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.

13.3.6.12 CopyTo(Int64, OracleClob, Int64, Int64)

This instance method copies data from the current `OracleClob` instance to the provided `OracleClob` object with the specified source offset, destination offset, and character amounts.

Declaration

```
// C#
public Int64 CopyTo(Int64 src_offset,OracleClob obj,Int64 dst_offset,
    Int64 amount);
```

Parameters

- *`src_offset`*

  The offset (in characters) in the current instance, from which the data is read.
- *`obj`*

  The `OracleClob` object to which the data is copied.
- *`dst_offset`*

  The offset (in characters) at which the `OracleClob` object is copied.
- *`amount`*

  The amount of data to be copied.

Return Value

The return value is the amount copied.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - The parameter has a different connection than the object, `OracleConnection` is not opened, or `OracleConnection` has been reopened.

`ArgumentOutOfRangeException` - The *`src_offset`*, the *`dst_offset`*, or the *`amount`* parameter is less than `0`.

Remarks

If the *`dst_offset`* is beyond the end of the `OracleClob` data, spaces are written into the `OracleClob` until the *`dst_offset`* is met.

The offsets are `0`-based. No character conversion is performed by this operation.

The provided object and the current instance must be using the same connection, that is, the same `OracleConnection` object.

Example

```
// C#

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

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

    OracleClob clob1 = new OracleClob(con);
    OracleClob clob2 = new OracleClob(con);

    // Write 4 chars, starting at buffer offset 0
    char[] buffer = new char[4] {'a', 'b', 'c', 'd'};
    clob1.Write(buffer, 0, 4);

    // Copy 2 chars from char 0 of clob1 to char 1 of clob2
    clob1.CopyTo(0, clob2, 1, 2);

    //Prints "clob2.Value =  ab"
    Console.WriteLine("clob2.Value = " + clob2.Value);

    clob1.Close();
    clob1.Dispose();

    clob2.Close();
    clob2.Dispose();

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

13.3.6.13 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.

13.3.6.14 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.

13.3.6.15 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).

13.3.6.16 Erase()

This instance method erases all data.

Declaration

```
// C#
public Int64 Erase();
```

Return Value

The number of characters erased.

13.3.6.17 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`.

13.3.6.18 Flush

This method is not supported.

13.3.6.19 GetHashCode

Overrides `Object`

This method returns a hash code for the current instance.

Declaration

```
// C#
public override int GetHashCode();
```

Return Value

An `int` representing a hash code.

13.3.6.20 IsEqual

This instance method compares the LOB data referenced by two `OracleClob`s.

Declaration

```
// C#
public bool IsEqual(OracleClob obj);
```

Parameters

- *`obj`*

  An `OracleClob` object.

Return Value

Returns `true` if the current `OracleClob` and the provided `OracleClob` refer to the same LOB. Otherwise, returns `false`.

Remarks

Note that this method can return `true` even if the two `OracleClob` objects returns `false` for == or `Equals()` because two different `OracleClob` instances can refer to the same LOB.

The provided object and the current instance must be using the same connection, that is, the same `OracleConnection` object.

13.3.6.21 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.

13.3.6.22 Read(byte [ ], int, int)

Overrides `Stream`

This instance method reads a specified amount of bytes from the current instance and populates the byte array `buffer`.

Declaration

```
// C#
public override int Read(byte [] buffer, int offset, int count);
```

Parameters

- *`buffer`*

  The byte array buffer that is populated.
- *`offset`*

  The offset (in bytes) at which the buffer is populated.
- *`count`*

  The amount of bytes to be read.

Return Value

The number of bytes read from the `CLOB`.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

Remarks

Both *`offset`* and *`count`* must be even numbers for `CLOB` and `NCLOB` because every two bytes represent a Unicode character.

The LOB data is read starting from the position specified by the `Position` property, which must also be an even number.

`OracleClob` is free to return fewer bytes than requested, even if the end of the stream has not been reached.

13.3.6.23 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();
  }
}
```

13.3.6.24 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`.

13.3.6.25 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.

13.3.6.26 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();
  }
}
```

13.3.6.27 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.

13.3.6.28 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`.

13.3.6.29 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`.

13.3.6.30 Write(byte[ ], int, int)

Overrides `Stream`

This instance method writes data from the provided byte array buffer into the `OracleClob`.

Declaration

```
// C#
public override void Write(byte[ ] buffer, int offset, int count);
```

Parameters

- *`buffer`*

  The byte array buffer that represents a Unicode string.

- *`offset`*

  The offset (in bytes) from which the `buffer` is read.

- *`count`*

  The amount of data (in bytes) from the buffer to be written into the `OracleClob`.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

`ArgumentOutOfRangeException` - This exception is thrown if any of the following conditions exist:

- The *`offset`* or the *`count`* is less than `0`.
- The *`offset`* is greater than or equal to the *`buffer`*`.Length`.
- The *`offset`* and the *`count`* together are greater than the *`buffer`*`.Length`.
- The *`offset`*, the *`count`*, or the `Position` is not even.

Remarks

Both *`offset`* and *`count`* must be even numbers for `CLOB` and `NCLOB` because every two bytes represent a Unicode character.

The LOB data is read starting from the position specified by the `Position` property. The `Position` property must be an even number.

If necessary, proper data conversion is carried out from the client character set to the database character set.

13.3.6.31 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();
  }
}
```

13.4 OracleRefCursor Class

An `OracleRefCursor` object represents an Oracle `REF CURSOR`..

Class Inheritance

```
System.Object

  System.MarshalRefByObject

    Oracle.DataAccess.Types.OracleRefCursor
```

Declaration

```
// C#
public sealed class OracleRefCursor : MarshalByRefObject, IDisposable, INullable
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver
Assembly	`Oracle.DataAccess.dll`	`Oracle.ManagedDataAccess.dll`
Namespace	`Oracle.DataAccess.Types`	`Oracle.ManagedDataAccess.Types`
.NET Framework	3.5, 4.5, 4.6	4.5, 4.6

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();

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

13.4.1 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

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

OracleRefCursor Static Fields

`OracleRefCursor` static field is listed in Table 13-29.

Table 13-29 OracleRefCursor Static Field

Methods	Description
Null	Represents a null value that can be assigned to an `OracleRefCursor` instance

OracleRefCursor Properties

`OracleRefCursor` properties are listed in Table 13-30.

Table 13-30 OracleRefCursor Properties

Properties	Description
Connection	A reference to the `OracleConnection` used to fetch the `REF CURSOR` data
FetchSize	Specifies the size that the `OracleDataReader` internal cache needs to store result set data
IsNull	Indicates whether or not the `OracleRefCursor` is null
RowSize	Specifies the amount of memory the `OracleRefcursor` internal cache needs to store one row of data

OracleRefCursor Instance Methods

`OracleRefCursor` instance methods are listed in Table 13-31.

Table 13-31 OracleRefCursor Instance Methods

Methods	Description
Dispose	Disposes the resources allocated by the `OracleRefCursor` object
`Equals`	Inherited from `System.Object` (Overloaded)
GetDataReader	Returns an `OracleDataReader` object for the `REF CURSOR`
`GetHashCode`	Inherited from `System.Object`
`GetType`	Inherited from `System.Object`
`ToString`	Inherited from `System.Object`

13.4.2 OracleRefCursor Static Methods

`OracleRefCursor` static methods are listed in Table 13-32.

Table 13-32 OracleRefCursor Static Methods

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

13.4.3 OracleRefCursor Static Fields

`OracleRefCursor` static field is listed in Table 13-32.

Table 13-33 OracleRefCursor Static Field

Methods	Description
Null	Represents a null value that can be assigned to an `OracleRefCursor` instance

13.4.3.1 Null

This static field represents a null value that can be assigned to an `OracleRefCursor` instance.

Declaration

```
// C#
public static readonly OracleRefCursor Null;
```

13.4.4 OracleRefCursor Properties

`OracleRefCursor` properties are listed in Table 13-34.

Table 13-34 OracleRefCursor Properties

Properties	Description
Connection	A reference to the `OracleConnection` used to fetch the `REF CURSOR` data
FetchSize	Specifies the size that the `OracleDataReader` internal cache needs to store result set data
IsNull	Indicates whether or not the `OracleRefCursor` is null
RowSize	Specifies the amount of memory the `OracleRefcursor` internal cache needs to store one row of data

13.4.4.1 Connection

This property refers to the `OracleConnection` used to fetch the `REF CURSOR` data.

Declaration

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

Property Value

An `OracleConnection`.

Exceptions

`ObjectDisposedException` - The object is already disposed.

Remarks

This property is bound to a `REF CURSOR` once it is set. After the `OracleRefCursor` object is created by the constructor, this property is initially `null`. An `OracleRefCursor` object can be bound to a `REF CURSOR` after a command execution.

If the connection is closed or returned to the connection pool, the `OracleRefCursor` is placed in an uninitialized state and no operation can be carried out from it. However, the uninitialized `OracleRefCursor` can be reassigned to another `REF CURSOR`.

13.4.4.2 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.

13.4.4.3 IsNull

This property indicates whether or not the OracleRefCursor is null.

Declaration

```
// C#
public bool IsNull {get;}
```

Property Value

Returns true if the OracleRefCursor represents a null value. Returns false otherwise.

Exception

ObjectDisposedException - The object is already disposed.

InvalidOperationException - The OracleConnection is not open or has been closed during the lifetime of the object.

13.4.4.4 RowSize

This property specifies the amount of memory the OracleRefcursor internal cache needs 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 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 RowSize * *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.

13.4.5 OracleRefCursor Instance Methods

`OracleRefCursor` instance methods are listed in Table 13-35.

Table 13-35 OracleRefCursor Instance Methods

Methods	Description
Dispose	Disposes the resources allocated by the `OracleRefCursor` object
`Equals`	Inherited from `System.Object` (Overloaded)
GetDataReader	Returns an `OracleDataReader` object for the `REF CURSOR`
`GetHashCode`	Inherited from `System.Object`
`GetType`	Inherited from `System.Object`
`ToString`	Inherited from `System.Object`

13.4.5.1 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.

13.4.5.2 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`.

14 Oracle Data Provider for .NET Types Structures

This chapter describes the ODP.NET Types structures.

This chapter contains these topics:

14.1 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

```
// C#
public struct OracleBinary : IComparable, INullable, IXmlSerializable
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver
Assembly	`Oracle.DataAccess.dll`	`Oracle.ManagedDataAccess.dll`
Namespace	`Oracle.DataAccess.Types`	`Oracle.ManagedDataAccess.Types`
.NET Framework	3.5, 4.5, 4.6	4.5, 4.6

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");
  }
}
```

14.1.1 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

Constructor	Description
OracleBinary Constructor	Instantiates a new instance of `OracleBinary` structure

OracleBinary Static Fields

The `OracleBinary` static fields are listed in Table 14-2.

Table 14-2 OracleBinary Static Fields

Field	Description
Null	Represents a null value that can be assigned to an instance of the `OracleBinary` structure

OracleBinary Static Methods

The `OracleBinary` static methods are listed in Table 14-3.

Table 14-3 OracleBinary Static Methods

Methods	Description
Concat	Returns the concatenation of two `OracleBinary` structures
Equals	Determines if two `OracleBinary` values are equal (Overloaded)
GetXsdType	Returns the XML Schema definition language (XSD) of the specified `XmlSchemaSet`
GreaterThan	Determines if the first of two `OracleBinary` values is greater than the second
GreaterThanOrEqual	Determines if the first of two `OracleBinary` values is greater than or equal to the second
LessThan	Determines if the first of two `OracleBinary` values is less than the second
LessThanOrEqual	Determines if the first of two `OracleBinary` values is less than or equal to the second
NotEquals	Determines if two `OracleBinary` values are not equal

OracleBinary Static Operators

The `OracleBinary` static operators are listed in Table 14-4.

Table 14-4 OracleBinary Static Operators

Operator	Description
operator +	Concatenates two `OracleBinary` values
operator ==	Determines if two `OracleBinary` values are equal
operator >	Determines if the first of two `OracleBinary` values is greater than the second
operator >=	Determines if the first of two `OracleBinary` values is greater than or equal to the second
operator !=	Determines if two `OracleBinary` values are not equal
operator <	Determines if the first of two `OracleBinary` value is less than the second
operator <=	Determines if the first of two `OracleBinary` value is less than or equal to the second

OracleBinary Static Type Conversion Operators

The `OracleBinary` static type conversion operators are listed in Table 14-5.

Table 14-5 OracleBinary Static Type Conversion Operators

Operator	Description
explicit operator byte[ ]	Converts an instance value to a byte array
implicit operator OracleBinary	Converts an instance value to an `OracleBinary` structure

OracleBinary Properties

The `OracleBinary` properties are listed in Table 14-6.

Table 14-6 OracleBinary Properties

Properties	Description
IsNull	Indicates whether or not the current instance has a null value
Item	Obtains the particular `byte` in an `OracleBinary` structure using an index
Length	Returns the length of the binary data
Value	Returns the binary data that is stored in an `OracleBinary` structure

OracleBinary Instance Methods

The `OracleBinary` instance methods are listed in Table 14-7.

Table 14-7 OracleBinary Instance Methods

Methods	Description
CompareTo	Compares the current instance to an object and returns an integer that represents their relative values
Equals	Determines if two objects contain the same binary data (Overloaded)
GetHashCode	Returns a hash code for the current instance
GetType	Inherited from `System.Object`
ToString	Converts the current `OracleBinary` structure to a string

14.1.2 OracleBinary Constructor

The `OracleBinary` constructor instantiates a new instance of the `OracleBinary` structure and sets its value to the provided array of bytes.

Declaration

```
// C#
public OracleBinary(byte[ ] bytes);
```

Parameters

- *`bytes`*

A byte array.

14.1.3 OracleBinary Static Fields

The `OracleBinary` static fields are listed in Table 14-8.

Table 14-8 OracleBinary Static Fields

Field	Description
Null	Represents a null value that can be assigned to an instance of the `OracleBinary` structure

14.1.3.1 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;
```

14.1.4 OracleBinary Static Methods

The `OracleBinary` static methods are listed in Table 14-9.

Table 14-9 OracleBinary Static Methods

Methods	Description
Concat	Returns the concatenation of two `OracleBinary` structures
Equals	Determines if two `OracleBinary` values are equal (Overloaded)
GetXsdType	Returns the XML Schema definition language (XSD) of the specified `XmlSchemaSet`
GreaterThan	Determines if the first of two `OracleBinary` values is greater than the second
GreaterThanOrEqual	Determines if the first of two `OracleBinary` values is greater than or equal to the second
LessThan	Determines if the first of two `OracleBinary` values is less than the second
LessThanOrEqual	Determines if the first of two `OracleBinary` values is less than or equal to the second
NotEquals	Determines if two `OracleBinary` values are not equal

14.1.4.1 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.

14.1.4.2 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 `OracleBinary`s that contain a null value are equal.

14.1.4.3 GetXsdType

This method returns the XML Schema definition language (XSD) of the specified `XmlSchemaSet`.

Declaration

```
// C#
public static XmlQualifiedName GetXsdType(XmlSchemaSet schemaSet);
```

Parameters

- *`schemaSet`*

  An `XmlSchemaSet`.

Return Value

Returns a string that indicates the XSD of the specified `XmlSchemaSet`.

14.1.4.4 GreaterThan

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 `OracleBinary`s 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");
  }
}
```

14.1.4.5 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 `OracleBinary`s that contain a null value are equal.

14.1.4.6 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 `OracleBinary`s that contain a null value are equal.

14.1.4.7 LessThanOrEqual

This method determines whether or not the first of two `OracleBinary` values is less than or equal to the second.

Declaration

```
// C#
public static bool LessThanOrEqual(OracleBinary value1, OracleBinary value2);
```

Parameters

- *value1*

  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 `OracleBinary`s that contain a null value are equal.

14.1.4.8 NotEquals

This method determines whether or not two `OracleBinary` values are not equal.

Declaration

```
// 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.

14.1.5 OracleBinary Static Operators

The `OracleBinary` static operators are listed in Table 14-10.

Table 14-10 OracleBinary Static Operators

Operator	Description
operator +	Concatenates two `OracleBinary` values
operator ==	Determines if two `OracleBinary` values are equal
operator >	Determines if the first of two `OracleBinary` values is greater than the second
operator >=	Determines if the first of two `OracleBinary` values is greater than or equal to the second
operator !=	Determines if two `OracleBinary` values are not equal
operator <	Determines if the first of two `OracleBinary` value is less than the second
operator <=	Determines if the first of two `OracleBinary` value is less than or equal to the second

14.1.5.2 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 `OracleBinary`s that contain a null value are equal.

14.1.5.3 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 `OracleBinary`s that contain a null value are equal.

Example

```
// 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");
  }
}
```

14.1.5.4 operator >=

This method determines if the first of two `OracleBinary` values is greater than or equal to the second.

Declaration

```
// C#
public static bool operator >= (OracleBinary value1, OracleBinary value2);
```

Parameters

- *`value1`*

  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.

14.1.5.5 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`.

14.1.5.6 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 `OracleBinary`s that contain a null value are equal.

14.1.5.7 operator <=

This method determines if the first of two `OracleBinary` values is less than or equal to the second.

Declaration

```
// C#
public static bool operator <= (OracleBinary value1, OracleBinary value1);
```

Parameters

- *value1*

  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 `OracleBinary`s that contain a null value are equal.

14.1.6 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

Operator	Description
explicit operator byte[ ]	Converts an instance value to a byte array
implicit operator OracleBinary	Converts an instance value to an `OracleBinary` structure

14.1.6.1 explicit operator byte[ ]

This method converts an `OracleBinary` value to a byte array.

Declaration

```
// C#
public static explicit operator byte[] (OracleBinary val);
```

Parameters

- *val*

  An OracleBinary.

Return Value

A byte array.

Exceptions

OracleNullValueException - The OracleBinary structure has a null value.

14.1.6.2 implicit operator OracleBinary

This method converts a byte array to an OracleBinary structure.

Declaration

```
// C#
public static implicit operator OracleBinary(byte[ ] bytes);
```

Parameters

- *bytes*

  A byte array.

Return Value

OracleBinary

14.1.7 OracleBinary Properties

The OracleBinary properties are listed in Table 14-12.

Table 14-12 OracleBinary Properties

Properties	Description
IsNull	Indicates whether or not the current instance has a null value
Item	Obtains the particular byte in an OracleBinary structure using an index
Length	Returns the length of the binary data
Value	Returns the binary data that is stored in an OracleBinary structure

14.1.7.1 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`.

14.1.7.2 Item

This property obtains the particular `byte` in an `OracleBinary` structure using an index.

Declaration

```
// C#
public byte this[int index] {get;}
```

Property Value

A byte in the specified index.

Exceptions

`OracleNullValueException` - The current instance has a null value.

Example

```
// C#

using System;
using Oracle.DataAccess.Types;

class ItemSample
{
  static void Main(string[] args)
  {
    OracleBinary binary = new OracleBinary(new byte[] {1,2,3,4});

    // Prints the value 4
    Console.WriteLine(binary[binary.Length - 1]);
  }
}
```

14.1.7.3 Length

This property returns the length of the binary data.

Declaration

```
// C#
public int length {get;}
```

Property Value

Length of the binary data.

Exceptions

`OracleNullValueException` - The current instance has a null value.

Example

```
// C#

using System;
using Oracle.DataAccess.Types;

class LengthSample
{
  static void Main(string[] args)
  {
    OracleBinary binary = new OracleBinary(new byte[] {1,2,3,4});

    // Prints the value 4
    Console.WriteLine(binary.Length);
  }
}
```

14.1.7.4 Value

This property returns the binary data that is stored in the `OracleBinary` structure.

Declaration

```
// C#
public byte[] Value {get;}
```

Property Value

Binary data.

Exceptions

`OracleNullValueException` - The current instance has a null value.

14.1.8 OracleBinary Instance Methods

The `OracleBinary` instance methods are listed in Table 14-13.

Table 14-13 OracleBinary Instance Methods

Methods	Description
CompareTo	Compares the current instance to an object and returns an integer that represents their relative values
Equals	Determines if two objects contain the same binary data (Overloaded)
GetHashCode	Returns a hash code for the current instance
GetType	Inherited from `System.Object`
ToString	Converts the current `OracleBinary` structure to a string

14.1.8.1 CompareTo

This method compares the current instance to an object and returns an integer that represents their relative values

Declaration

```
// C#
public int CompareTo(object obj);
```

Parameters

- *obj*

  The object being compared.

Return Value

The method returns a number that is:

- Less than zero: if the current `OracleBinary` instance value is less than *obj*.
- Zero: if the current `OracleBinary` instance and *obj* values have the same binary data.
- Greater than zero: if the current `OracleBinary` instance value is greater than *obj*.

Implements

`IComparable`

Exceptions

`ArgumentException` - The parameter is not of type `OracleBinary`.

Remarks

The following rules apply to the behavior of this method.

- The comparison must be between `OracleBinary`s. For example, comparing an `OracleBinary` instance with an `OracleTimeStamp` instance is not allowed. When an `OracleBinary` is compared with a different type, an `ArgumentException` is thrown.
- Any `OracleBinary` that has a value is greater than an `OracleBinary` that has a null value.
- Two `OracleBinary`s that contain a null value are equal.

Example

```
// C#

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");
  }
}
```

14.1.8.2 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

```
// 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.

14.1.8.3 GetHashCode

Overrides `Object`

This method returns a hash code for the `OracleBinary` instance.

Declaration

```
// C#
public override int GetHashCode();
```

Return Value

An `int` that represents the hash.

14.1.8.4 ToString

Overrides `Object`

This method converts an `OracleBinary` instance to a string instance.

Declaration

```
// C#
public override string ToString();
```

Return Value

`string`

Remarks

If the current `OracleBinary` instance has a null value, the returned string "null".

14.2 OracleBoolean Structure

The `OracleBoolean` structure represents a logical value that is either `TRUE` or `FALSE`.

ODP.NET, Unmanaged Driver can access Oracle Database PL/SQL Booleans in Oracle Database Release 12.1 and later. ODP.NET, Managed Driver can access Oracle Database PL/SQL Booleans in Oracle Database Release 12.2 and later.

Class Inheritance

```
System.Object

  System.ValueType

    Oracle.DataAccess.Types.OracleBoolean
```

Declaration

```
// C#
public struct OracleBoolean : IComparable, INullable, IXmlSerializable
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver
Assembly	`Oracle.DataAccess.dll`	`Oracle.ManagedDataAccess.dll`
Namespace	`Oracle.DataAccess.Types`	`Oracle.ManagedDataAccess.Types`
.NET Framework	3.5, 4.5, 4.6	4.5, 4.6

Thread Safety

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

Remarks

A `OracleBoolean` structure represents three possible values: `TRUE`, `FALSE`, and `NULL`. A non-zero value is interpreted as `TRUE`.

Example

```
// C#
using System;
using System.Data;
using Oracle.DataAccess.Types; // for use with ODP.NET, Unmanaged Driver
// using Oracle.ManagedDataAccess.Types; // for use with ODP.NET, Managed Driver

 class OracleBooleanSample
 {
    static void Main(string[] args)
    {
      OracleBoolean oracleBoolean1 = new OracleBoolean(true);
      OracleBoolean oracleBoolean2 = new OracleBoolean(0);

      Console.WriteLine("oracleBoolean1 : " + oracleBoolean1);
      Console.WriteLine("oracleBoolean2 : " + oracleBoolean2);
    }
 }
```

14.2.1 OracleBoolean Members

`OracleBoolean` members are listed in the following tables:

OracleBoolean Constructors

`OracleBoolean` constructors are listed in Table 14-14

Table 14-14 OracleBoolean Constructors

Constructor	Description
OracleBoolean Constructors	Instantiates a new instance of `OracleBoolean` structure (Overloaded)

OracleBoolean Static Fields

The `OracleBoolean` static fields are listed in Table 14-15.

Table 14-15 OracleBoolean Static Fields

Field	Description
False	Represents a false value that can be assigned to an `OracleBoolean` instance
Null	Represents a null value that can be assigned to an `OracleBoolean` instance
One	Indicates a constant representing the positive one value
True	Represents a true value that can be assigned to an `OracleBoolean` instance
Zero	Indicates a constant representing the zero value

OracleBoolean Static Methods

`OracleBoolean` static methods are listed in Table 14-16

Table 14-16 OracleBoolean Static Methods

Methods	Description
And	Returns the result of bitwise `AND` operation of two `OracleBoolean` instances
Equals	Determines whether or not the two `OracleBoolean` values are equal
GreaterThan	Determines whether or not the first of two `OracleBoolean` values is greater than the second
GreaterThanOrEquals	Determines whether or not the first of two `OracleBoolean` values is greater than or equal to the second
LessThan	Determines whether or not the first of two `OracleBoolean` values is less than the second
LessThanOrEquals	Determines whether or not the first of two `OracleBoolean` values is less than or equal to the second
NotEquals	Determines whether or not two `OracleBoolean` values are not equal
OnesComplement	Returns the result of a one's complement operation on the specified `OracleBoolean` value
Or	Returns the result of bitwise `OR` operation of two `OracleBoolean` instances
Parse	Returns an `OracleBoolean` structure and sets its value using a `string`
Xor	Returns the result of a bitwise exclusive `OR` operation of two `OracleBoolean` instances

OracleBoolean Static Operators

The `OracleBoolean` static operators are listed in Table 14-17.

Table 14-17 OracleBoolean Static Operators

Field	Description
operator >	Determines whether or not the first of two `OracleBoolean` values is greater than the second
operator >=	Determines whether or not the first of two `OracleBoolean` values is greater than or equal to the second
operator <	Determines whether or not the first of two `OracleBoolean` values is less than the second
operator <=	Dtermines whether or not the first of two `OracleBoolean` values is less than or equal to the second
operator ==	Indicates whether or not the two `OracleBoolean` instances are equal

Table 14-17 (Cont.) OracleBoolean Static Operators

Field	Description
operator !=	Determines whether or not two `OracleBoolean` values are not equal
operator !	Determines the result of a `NOT` operation on a `OracleBoolean`
operator ~	Returns the result of a one's complement operation on the specified `OracleBoolean` value
operator false	Determines whether or not the specified `OracleBoolean` value is false
operator true	Determines whether or not the specified `OracleBoolean` value is true
operator &	Returns the result of bitwise `AND` operation of two `OracleBoolean` instances
operator |	Returns the result of bitwise `OR` operation of two `OracleBoolean` instances
operator ^	Returns the result of bitwise exclusive `OR` operation of two `OracleBoolean` instances

The OracleBoolean Static Type conversions

The `OracleBoolean` static type conversions are listed in Table 14-18

Table 14-18 OracleBoolean Static Type Conversions

Field	Description
implicit operator OracleBoolean	Returns the `OracleBoolean` representation of a boolean value
explicit operator bool	Returns the boolean representation of the `OracleBoolean` value
explicit operator OracleBoolean	Converts a structure to an `OracleBoolean` structure (Overloaded)

OracleBoolean Properties

The `OracleBoolean` properties are listed in Table 14-25.

Table 14-19 OracleBoolean Properties

Properties	Description
ByteValue	Returns a `byte` that represents the `OracleBoolean` structure
IsFalse	Indicates whether or not the value of the current instance is false
IsNull	Indicates whether or not the current instance has a null value
IsTrue	Indicates whether or not the value of the current instance is true
Value	Returns a boolean value that represents the current instance

OracleBoolean Instance Methods

The `OracleBoolean` instance methods are listed in Table 14-20.

Table 14-20 OracleBoolean Instance Methods

Method	Description
CompareTo	Compares the current instance to the supplied object and returns an integer that represents their relative values
Equals	Determines whether or not an object is an instance of `OracleBoolean`, and whether or not the value of the object is equal to the current instance
GetHashCode	Returns a hash code for the current instance
ToString	Returns the `string` representation of the current instance

14.2.2 OracleBoolean Constructors

The `OracleBoolean` constructors instantiates a new instance of the `OracleBoolean` structure.

Overload List:

- OracleBoolean(bool)

  This constructor creates a new instance of the `OracleBoolean` structure and sets its value to the supplied `Boolean` value.
- OracleBoolean(int)

  This constructor creates a new instance of the `OracleBoolean` structure and sets its value to the supplied `Int32` value.

14.2.2.1 OracleBoolean(bool)

This constructor creates a new instance of the `OracleBoolean` structure and sets its value to the supplied `Boolean` value.

Declaration

```
// C#
public OracleBoolean(bool value) ;
```

Parameters

- *`value`*

  The provided `Boolean` value.

14.2.2.2 OracleBoolean(int)

This constructor creates a new instance of the `OracleBoolean` structure and sets its value to the supplied `Int32` value.

Declaration

```
// C#
public OracleBoolean(int value)
```

Parameters

- *value*

  The provided `Int32` value.

14.2.3 OracleBoolean Static Fields

The `OracleBoolean` static fields are listed in Table 14-21.

Table 14-21 OracleBoolean Static Fields

Field	Description
False	Represents a false value that can be assigned to an `OracleBoolean` instance
Null	Represents a null value that can be assigned to an `OracleBoolean` instance
One	Indicates a constant representing the positive one value
True	Represents a true value that can be assigned to an `OracleBoolean` instance
Zero	Indicates a constant representing the zero value

14.2.3.1 False

This static field represents a false value that can be assigned to an `OracleBoolean` instance.

Declaration

```
// C#
public static readonly OracleBoolean False;
```

14.2.3.2 Null

This static field represents a null value that can be assigned to an `OracleBoolean` instance.

Declaration

```
// C#
public static readonly OracleBoolean Null;
```

14.2.3.3 One

This static field indicates a constant representing the positive one value.

Declaration

```
// C#
public static readonly OracleBoolean One;
```

14.2.3.4 True

This static field represents a true value that can be assigned to an `OracleBoolean` instance.

Declaration

```
// C#
public static readonly OracleBoolean True;
```

14.2.3.5 Zero

This static field indicates a constant representing the zero value.

Declaration

```
// C#
public static readonly OracleBoolean Zero;
```

14.2.4 OracleBoolean Static Methods

`OracleBoolean` static methods are listed in Table 14-22

Table 14-22 OracleBoolean Static Methods

Methods	Description
And	Returns the result of bitwise `AND` operation of two `OracleBoolean` instances
Equals	Determines whether or not the two `OracleBoolean` values are equal
GreaterThan	Determines whether or not the first of two `OracleBoolean` values is greater than the second
GreaterThanOrEquals	Determines whether or not the first of two `OracleBoolean` values is greater than or equal to the second
LessThan	Determines whether or not the first of two `OracleBoolean` values is less than the second
LessThanOrEquals	Determines whether or not the first of two `OracleBoolean` values is less than or equal to the second
NotEquals	Determines whether or not two `OracleBoolean` values are not equal
OnesComplement	Returns the result of a one's complement operation on the specified `OracleBoolean` value
Or	Returns the result of bitwise `OR` operation of two `OracleBoolean` instances

Table 14-22 (Cont.) OracleBoolean Static Methods

Methods	Description
Parse	Returns an `OracleBoolean` structure and sets its value using a `string`
Xor	Returns the result of a bitwise exclusive `OR` operation of two `OracleBoolean` instances

14.2.4.1 And

This method returns the result of bitwise `AND` operation of two `OracleBoolean` instances.

Declaration

```
// C#
public static OracleBoolean And(OracleBoolean value1, OracleBoolean value2);
```

Parameters

- *`value1`*

  An `OracleBoolean` instance
- *`value2`*

  An `OracleBoolean` instance

Return Value

An `OracleBoolean` that contains the value of the result of bitwise `AND` operation of two `OracleBoolean` instances.

Remarks

If either of the specified `OracleBoolean` instances is `null`, an `OracleBoolean` with a null value will be returned.

14.2.4.2 Equals

This method returns an `OracleBoolean` that indicates whether or not the two `OracleBoolean` instances are equal.

Declaration

```
// C#
public static OracleBoolean Equal(OracleBoolean value1, OracleBoolean value2);
```

Parameters

- *`value1`*

  An `OracleBoolean` instance
- *`value2`*

  An `OracleBoolean` instance

Return Value

An `OracleBoolean` that is `true` if the specified two `OracleBoolean` instances are equal; otherwise, returns an `OracleBoolean` that is `false`.

Remarks

If either of the specified `OracleBoolean` instances is `null`, an `OracleBoolean` with a null value will be returned.

14.2.4.3 GreaterThan

This method determines if the first of two `OracleBoolean` values is greater than the second.

Declaration

```
// C#
public static OracleBoolean GreaterThan(OracleBoolean value1, OracleBoolean value2);
```

Parameters

- *`value1`*

  The first `OracleBoolean`
- *`value2`*

  The second `OracleBoolean`

Return Value

An `OracleBoolean` that is `true` if the first of two `OracleBoolean` values is greater than the second; otherwise, returns `false`.

Remarks

If either of the specified `OracleBoolean` instances is `null`, an `OracleBoolean` with a null value will be returned.

14.2.4.4 GreaterThanOrEquals

This method determines if the first of two `OracleBoolean` values is greater than or equal to the second.

Declaration

```
// C#
public static OracleBoolean GreaterThanOrEquals(OracleBoolean value1, OracleBoolean
value2);
```

Parameters

- *`value1`*

  The first `OracleBoolean`
- *`value2`*

The second `OracleBoolean`

Return Value

An `OracleBoolean` that is `true` if the first of two `OracleBoolean` values is greater than or equal to the second; otherwise, returns `false`.

Remarks

If either of the specified `OracleBoolean` instances is `null`, an `OracleBoolean` with a null value will be returned.

14.2.4.5 LessThan

This method determines if the first of two `OracleBoolean` values is less than the second.

Declaration

```
// C#
public static OracleBoolean LessThan(OracleBoolean value1, OracleBoolean value2);
```

Parameters

- *`value1`*

  The first `OracleBoolean`
- *`value2`*

  The second `OracleBoolean`

Return Value

An `OracleBoolean` that is `true` if the first of two `OracleBoolean` values is less than the second; otherwise, returns `false`.

Remarks

If either of the specified `OracleBoolean` instances is `null`, an `OracleBoolean` with a null value will be returned.

14.2.4.6 LessThanOrEquals

This method determines if the first of two `OracleBoolean` values is less or equal than the second.

Declaration

```
// C#
public static OracleBoolean LessThanOrEquals(OracleBoolean value1, OracleBoolean
value2);
```

Parameters

- *`value1`*

  The first `OracleBoolean`
- *`value2`*

The second `OracleBoolean`

Return Value

An `OracleBoolean` that is `true` if the first of two `OracleBoolean` values is less than or equal to the second; otherwise, returns `false`.

Remarks

If either of the specified `OracleBoolean` instances is `null`, an `OracleBoolean` with a null value will be returned.

14.2.4.7 NotEquals

This method determines if two `OracleBoolean` values are not equal.

Declaration

```
// C#
public static OracleBoolean NotEquals(OracleBoolean value1, OracleBoolean value2);
```

Parameters

- *`value1`*

  The first `OracleBoolean`
- *`value2`*

  The second `OracleBoolean`

Return Value

An `OracleBoolean` that is `true` if two `OracleBoolean` values are not equal; otherwise, returns `false`.

Remarks

If either of the specified `OracleBoolean` instances is `null`, an `OracleBoolean` with a null value will be returned.

14.2.4.8 OnesComplement

This method returns the result of a one's complement operation on the specified `OracleBoolean` value.

Declaration

```
// C#
public static OracleBoolean OnesComplement(OracleBoolean value1);
```

Parameters

- *`value1`*

  An `OracleBoolean` instance

Return Value

An `OracleBoolean` that contains the value of the result of a one's complement operation on the specified `OracleBoolean` value.

Remarks

If the specified `OracleBoolean` instance is `null`, an `OracleBoolean` with a null value will be returned.

14.2.4.9 Or

This method returns the result of bitwise `OR` operation of two `OracleBoolean` instances.

Declaration

```
// C#
public static OracleBoolean Or(OracleBoolean value1, OracleBoolean value2);
```

Parameters

- *`value1`*

  An `OracleBoolean` instance
- *`value2`*

  An `OracleBoolean` instance

Return Value

An `OracleBoolean` that contains the value of the result of bitwise `OR` operation of two `OracleBoolean` instances.

Remarks

If either of the specified `OracleBoolean` instances is `null`, an `OracleBoolean` with a null value will be returned.

14.2.4.10 Parse

This method converts a `string` to an `OracleBoolean`.

Declaration

```
// C#
public static OracleBoolean Parse(string str);
```

Parameters

- *`str`*

  The `string` being converted.

Return Value

A new `OracleBoolean` structure.

Exceptions

`ArgumentNullException` – The `str` parameter is `null`.

`IndexOutOfRangeException` – The `str` parameter is an empty string.

14.2.4.11 Xor

This method returns the result of a bitwise exclusive `OR` operation of two `OracleBoolean` instances.

Declaration

```
// C#
public static OracleBoolean Xor(OracleBoolean value1, OracleBoolean value2);
```

Parameters

- *`value1`*

  An `OracleBoolean` instance
- *`value2`*

  An `OracleBoolean` instance

Return Value

An `OracleBoolean` that contains the value of the result of bitwise exclusive `OR` operation of two `OracleBoolean` instances.

Remarks

If either of the specified `OracleBoolean` instances is `null`, an `OracleBoolean` with a null value will be returned.

14.2.5 OracleBoolean Static Operators

The `OracleBoolean` static operators are listed in Table 14-23.

Table 14-23 OracleBoolean Static Operators

Field	Description
operator >	Determines whether or not the first of two `OracleBoolean` values is greater than the second
operator >=	Determines whether or not the first of two `OracleBoolean` values is greater than or equal to the second
operator <	Determines whether or not the first of two `OracleBoolean` values is less than the second
operator <=	Dtermines whether or not the first of two `OracleBoolean` values is less than or equal to the second
operator ==	Indicates whether or not the two `OracleBoolean` instances are equal

Table 14-23 (Cont.) OracleBoolean Static Operators

Field	Description
operator !=	Determines whether or not two `OracleBoolean` values are not equal
operator !	Determines the result of a `NOT` operation on a `OracleBoolean`
operator ~	Returns the result of a one's complement operation on the specified `OracleBoolean` value
operator false	Determines whether or not the specified `OracleBoolean` value is false
operator true	Determines whether or not the specified `OracleBoolean` value is true
operator &	Returns the result of bitwise `AND` operation of two `OracleBoolean` instances
operator |	Returns the result of bitwise `OR` operation of two `OracleBoolean` instances
operator ^	Returns the result of bitwise exclusive `OR` operation of two `OracleBoolean` instances

14.2.5.1 operator >

This method determines whether or not the first of two `OracleBoolean` values is greater than the second.

Declaration

```
// C#
public static operator  > (OracleBoolean value1, OracleBoolean value2);
```

Parameters

- *`value1`*

  An `OracleBoolean` instance

- *`value2`*

  An `OracleBoolean` instance

Return Value

An `OracleBoolean` that is `true` if the first of two `OracleBoolean` values is greater than the second; otherwise, returns false.

Remarks

If either of the specified `OracleBoolean` instances is `null`, an `OracleBoolean` with a null value will be returned.

14.2.5.2 operator >=

This method determines whether or not the first of two `OracleBoolean` values is greater than or equal to the second.

Declaration

```
// C#
public static operator  >= (OracleBoolean value1, OracleBoolean value2);
```

Parameters

- *`value1`*

  An `OracleBoolean` instance
- *`value2`*

  An `OracleBoolean` instance

Return Value

An `OracleBoolean` that is `true` if the first of two `OracleBoolean` values is greater than or equal to the second; otherwise, returns false.

Remarks

If either of the specified `OracleBoolean` instances is `null`, an `OracleBoolean` with a null value will be returned.

14.2.5.3 operator <

This method determines whether or not the first of two `OracleBoolean` values is less than the second.

Declaration

```
// C#
public static operator  < (OracleBoolean value1, OracleBoolean value2);
```

Parameters

- *`value1`*

  An `OracleBoolean` instance
- *`value2`*

  An `OracleBoolean` instance

Return Value

An `OracleBoolean` that is `true` if the first of two `OracleBoolean` values is less than the second; otherwise, returns false.

Remarks

If either of the specified `OracleBoolean` instances is `null`, an `OracleBoolean` with a null value will be returned.

14.2.5.4 operator <=

This method determines whether or not the first of two `OracleBoolean` values is less than or equal to the second.

Declaration

```
// C#
public static operator  <= (OracleBoolean value1, OracleBoolean value2);
```

Parameters

- *`value1`*

  An `OracleBoolean` instance
- *`value2`*

  An `OracleBoolean` instance

Return Value

An `OracleBoolean` that is `true` if the first of two `OracleBoolean` values is less than or equal to the second; otherwise, returns false.

Remarks

If either of the specified `OracleBoolean` instances is `null`, an `OracleBoolean` with a null value will be returned.

14.2.5.5 operator ==

This method returns an `OracleBoolean` that indicates whether or not the two `OracleBoolean` instances are equal.

Declaration

```
// C#
public static operator  == (OracleBoolean value1, OracleBoolean value2);
```

Parameters

- *`value1`*

  An `OracleBoolean` instance
- *`value2`*

  An `OracleBoolean` instance

Return Value

An `OracleBoolean` that is `true` if the specified two `OracleBoolean` instances are equal; otherwise, returns false.

Remarks

If either of the specified `OracleBoolean` instances is `null`, an `OracleBoolean` with a null value will be returned.

14.2.5.6 operator !=

This method determines whether or not two `OracleBoolean` values are not equal.

Declaration

```
// C#
public static operator  != (OracleBoolean value1, OracleBoolean value2);
```

Parameters

- *`value1`*

  An `OracleBoolean` instance
- *`value2`*

  An `OracleBoolean` instance

Return Value

An `OracleBoolean` that is `true` if two `OracleBoolean` values are not equal; otherwise, returns `false`.

Remarks

If either of the specified `OracleBoolean` instances is `null`, an `OracleBoolean` with a null value will be returned.

14.2.5.7 operator !

This method determines the result of a `NOT` operation on a `OracleBoolean`.

Declaration

```
// C#
public static operator  ! (OracleBoolean value1);
```

Parameters

- *`value1`*

  An `OracleBoolean` instance

Return Value

An `OracleBoolean` that is `true` if the specified `OracleBoolean` value is true; otherwise, returns false.

Remarks

If the specified `OracleBoolean` instance is `null`, an `OracleBoolean` with a null value will be returned.

14.2.5.8 operator ~

This method returns the result of a one's complement operation on the specified `OracleBoolean` value.

Declaration

```
// C#
public static operator ~ (OracleBoolean value1);
```

Parameters

- *`value1`*

  An `OracleBoolean` instance

Return Value

An `OracleBoolean` that contains the value of the result of a one's complement operation on the specified `OracleBoolean` value.

Remarks

If the specified `OracleBoolean` instance is `null`, an `OracleBoolean` with a null value will be returned.

14.2.5.9 operator false

This method determines whether or not the specified `OracleBoolean` value is false.

Declaration

```
// C#
public static operator false (OracleBoolean value1);
```

Parameters

- *`value1`*

  An `OracleBoolean` instance

Return Value

An `OracleBoolean` that is `true` if specified `OracleBoolean` value is false; otherwise, returns false.

Remarks

This property will return `false` if the current instance is `null`.

14.2.5.10 operator true

This method determines whether or not the specified `OracleBoolean` value is true.

Declaration

```
// C#
public static operator true (OracleBoolean value1);
```

Parameters

- *value1*

  An OracleBoolean instance

Return Value

An OracleBoolean that is true if specified OracleBoolean value is true; otherwise, returns false.

Remarks

This property will return false if the current instance is null.

14.2.5.11 operator &

This method returns the result of bitwise AND operation of two OracleBoolean instances.

Declaration

```
// C#
public static operator & (OracleBoolean value1, OracleBoolean value2);
```

Parameters

- *value1*

  An OracleBoolean instance
- *value2*

  An OracleBoolean instance

Return Value

An OracleBoolean that contains the value of the result of bitwise AND operation of two OracleBoolean instances.

Remarks

If either of the specified OracleBoolean instances is null, an OracleBoolean with a null value will be returned.

14.2.5.12 operator |

This method returns the result of bitwise OR operation of two OracleBoolean instances.

Declaration

```
// C#
public static operator | (OracleBoolean value1, OracleBoolean value2);
```

Parameters

- *value1*

  An `OracleBoolean` instance

- *value2*

  An `OracleBoolean` instance

Return Value

An `OracleBoolean` that contains the value of the result of bitwise `OR` operation of two `OracleBoolean` instances.

Remarks

If either of the specified `OracleBoolean` instances is `null`, an `OracleBoolean` with a null value will be returned.

14.2.5.13 operator ^

This method returns the result of bitwise exclusive `OR` operation of two `OracleBoolean` instances.

Declaration

```
// C#
public static operator ^ (OracleBoolean value1, OracleBoolean value2);
```

Parameters

- *value1*

  An `OracleBoolean` instance

- *value2*

  An `OracleBoolean` instance

Return Value

An `OracleBoolean` that contains the value of the result of bitwise exclusive `OR` operation of two `OracleBoolean` instances.

Remarks

If either of the specified `OracleBoolean` instances is `null`, an `OracleBoolean` with a null value will be returned.

14.2.6 OracleBoolean Static Type Conversions

The `OracleBoolean` static type conversions are listed in Table 14-24

Table 14-24 OracleBoolean Static Type Conversions

Field	Description
implicit operator OracleBoolean	Returns the `OracleBoolean` representation of a boolean value
explicit operator bool	Returns the boolean representation of the `OracleBoolean` value
explicit operator OracleBoolean	Converts a structure to an `OracleBoolean` structure (Overloaded)

14.2.6.1 implicit operator OracleBoolean

This method returns the `OracleBoolean` representation of a boolean value.

Declaration

```
// C#
public static implicit operator OracleBoolean(bool value1);
```

Parameters

- *`value1`*

  An `OracleBoolean` instance

Return Value

An `OracleBoolean`.

14.2.6.2 explicit operator bool

This method returns the boolean representation of the `OracleBoolean` value.

Declaration

```
// C#
public static explicit operator bool(OracleBoolean value1);
```

Parameters

- *`value1`*

  An `OracleBoolean` structure

Return Value

A boolean

Exception

`OracleNullValueException` – `OracleBoolean` has a null value.

14.2.6.3 explicit operator OracleBoolean

`explicit operator OracleBoolean` converts the provided structure to an `OracleBoolean` structure.

Overload List

- explicit operator OracleBoolean(byte)

  This method converts the supplied byte to an `OracleBoolean` structure.
- explicit operator OracleBoolean(Decimal)

  This method converts the supplied Decimal to an `OracleBoolean` structure.
- explicit operator OracleBoolean(Double)

  This method converts the supplied Double to an `OracleBoolean` structure.
- explicit operator OracleBoolean(Int16)

  This method converts the supplied Int16 to an `OracleBoolean` structure.
- explicit operator OracleBoolean(int)

  This method converts the supplied int to an `OracleBoolean` structure.
- explicit operator OracleBoolean(Int64)

  This method converts the supplied Int64 to an `OracleBoolean` structure.
- explicit operator OracleBoolean(Single)

  This method converts the supplied Single to an `OracleBoolean` structure.
- explicit operator OracleBoolean(String)

  This method converts the supplied String to an `OracleBoolean` structure.

14.2.6.4 explicit operator OracleBoolean(byte)

This method converts the supplied byte to an `OracleBoolean` structure.

Declaration

```
// C#
public static explicit operator OracleBoolean(byte value1);
```

Parameters

- *`value1`*

  A byte

Return Value

An `OracleBoolean` structure.

14.2.6.5 explicit operator OracleBoolean(Decimal)

This method converts the supplied Decimal to an `OracleBoolean` structure.

Declaration

```
// C#
public static explicit operator OracleBoolean(Decimal value1);
```

Parameters

- *value1*

  A Decimal

Return Value

An `OracleBoolean` structure.

14.2.6.6 explicit operator OracleBoolean(Double)

This method converts the supplied Double to an `OracleBoolean` structure.

Declaration

```
// C#
public static explicit operator OracleBoolean(Double value1);
```

Parameters

- *value1*

  A Double

Return Value

An `OracleBoolean` structure.

14.2.6.7 explicit operator OracleBoolean(Int16)

This method converts the supplied Int16 to an `OracleBoolean` structure.

Declaration

```
// C#
public static explicit operator OracleBoolean(Int16 value1);
```

Parameters

- *value1*

  An Int16

Return Value

An `OracleBoolean` structure.

14.2.6.8 explicit operator OracleBoolean(int)

This method converts the supplied int to an `OracleBoolean` structure.

Declaration

```
// C#
public static explicit operator OracleBoolean(int value1);
```

Parameters

- *`value1`*

  An int

Return Value

An `OracleBoolean` structure.

14.2.6.9 explicit operator OracleBoolean(Int64)

This method converts the supplied Int64 to an `OracleBoolean` structure.

Declaration

```
// C#
public static explicit operator OracleBoolean(Int64 value1);
```

Parameters

- *`value1`*

  An Int64

Return Value

An `OracleBoolean` structure.

14.2.6.10 explicit operator OracleBoolean(Single)

This method converts the supplied Single to an `OracleBoolean` structure.

Declaration

```
// C#
public static explicit operator OracleBoolean(Single value1);
```

Parameters

- *`value1`*

  A Single

Return Value

An `OracleBoolean` structure.

14.2.6.11 explicit operator OracleBoolean(String)

This method converts the supplied String to an `OracleBoolean` structure.

Declaration

```
// C#
public static explicit operator OracleBoolean(String value1);
```

Parameters

- *value1*

  A String

Return Value

An `OracleBoolean` structure.

14.2.7 OracleBoolean Properties

The `OracleBoolean` properties are listed in Table 14-25.

Table 14-25 OracleBoolean Properties

Properties	Description
ByteValue	Returns a `byte` that represents the `OracleBoolean` structure
IsFalse	Indicates whether or not the value of the current instance is false
IsNull	Indicates whether or not the current instance has a null value
IsTrue	Indicates whether or not the value of the current instance is true
Value	Returns a boolean value that represents the current instance

14.2.7.1 ByteValue

This property returns a byte that represents the `OracleBoolean` structure.

Declaration

```
// C#
public byte ByteValue {get;}
```

Property Value

A byte that represents the value of `OracleBoolean` structure.

Exceptions

`OracleNullValueException` – The current instance has a null value.

14.2.7.2 IsFalse

This property indicates whether or not the value of the current instance is false.

Declaration

```
// C#
public bool IsFalse {get;}
```

Property Value

A bool value that returns `true` if the current instance is false; otherwise, returns `false`.

Remarks

This property will return false if the current instance is null.

14.2.7.3 IsNull

This property indicates whether or not the current instance has a null value.

Declaration

```
// C#
public bool IsNull {get;}
```

Property Value

A bool value that returns `true` if the current instance has a null value; otherwise, returns `false`.

14.2.7.4 IsTrue

This property indicates whether or not the value of the current instance is true.

Declaration

```
// C#
public bool IsTrue {get;}
```

Property Value

A bool value that returns `true` if the current instance is true; otherwise, returns `false`.

Remarks

This property will return `false` if the current instance is null.

14.2.7.5 Value

This property returns a boolean value that represents the current instance.

Declaration

```
// C#
public bool Value {get;}
```

Property Value

A bool value that returns `true` if the current instance is true; otherwise, returns `false`.

Exceptions

`OracleNullValueException` – The current instance has a null value.

14.2.8 OracleBoolean Instance Methods

The `OracleBoolean` instance methods are listed in Table 14-26.

Table 14-26 OracleBoolean Instance Methods

Method	Description
CompareTo	Compares the current instance to the supplied object and returns an integer that represents their relative values
Equals	Determines whether or not an object is an instance of `OracleBoolean`, and whether or not the value of the object is equal to the current instance
GetHashCode	Returns a hash code for the current instance
ToString	Returns the `string` representation of the current instance

14.2.8.1 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);
```

Parameter

- *`obj`*

  The supplied instance.

Return Value

The method returns a number:

- Less than zero: if the value of the current instance is less than obj.
- Zero: if the value of the current instance is equal to obj.
- Greater than zero: if the value of the current instance is greater than obj.

Implements

`IComparable`

Exceptions

`ArgumentException` - The parameter is not of type `OracleBoolean`.

Remarks

The following rules apply to the behavior of this method.

- The comparison must be between `OracleBoolean`. For example, comparing an `OracleBoolean` instance with an `OracleBinary` instance is not allowed. When an `OracleBoolean` is compared with a different type, an `ArgumentException` is thrown.
- Any `OracleBoolean` that has a value compares greater than an `OracleBoolean` that has a null value.
- Two `OracleBoolean` that contain a null value are equal.

14.2.8.2 Equals

Overrides `Object`

This method determines whether or not an object is an instance of `OracleBoolean`, and whether or not the value of the object is equal to the current instance.

Declaration

```
// C#
public override bool Equals(object obj);
```

Parameter

- obj

  An `OracleBoolean` instance.

Return Value

Returns `true` if obj is an instance of `OracleBoolean`, 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 `OracleBoolean` that has a value compares greater than an `OracleBoolean` that has a null value.
- Two `OracleBooleans` that contain a null value are equal.

14.2.8.3 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.

14.2.8.4 ToString

Overrides `Object`

This method returns the string representation of the current instance.

Declaration

```
// C#
public override string ToString();
```

Return Value

Returns the `OracleBoolean` value in a string representation.

Remarks

If the current instance has a null value, the returned string is `null`.

14.3 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

```
// C#
public struct OracleDate : IComparable, INullable, IXmlSerializable
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver
Assembly	`Oracle.DataAccess.dll`	`Oracle.ManagedDataAccess.dll`
Namespace	`Oracle.DataAccess.Types`	`Oracle.ManagedDataAccess.Types`
.NET Framework	3.5, 4.5, 4.6	4.5, 4.6

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);
  }
}
```

14.3.1 OracleDate Members

`OracleDate` members are listed in the following tables:

OracleDate Constructors

`OracleDate` constructors are listed in Table 14-27

Table 14-27 OracleDate Constructors

Constructor	Description
OracleDate Constructors	Instantiates a new instance of `OracleDate` structure (Overloaded)

OracleDate Static Fields

The `OracleDate` static fields are listed in Table 14-28.

Table 14-28 OracleDate Static Fields

Field	Description
MaxValue	Represents the maximum valid date for an `OracleDate` structure, which is December 31, 9999 23:59:59
MinValue	Represents the minimum valid date for an `OracleDate` structure, which is January 1, -4712 0:0:0
Null	Represents a null value that can be assigned to the value of an `OracleDate` structure instance

OracleDate Static Methods

The `OracleDate` static methods are listed in Table 14-29.

Table 14-29 OracleDate Static Methods

Methods	Description
Equals	Determines if two `OracleDate` values are equal (Overloaded)
GreaterThan	Determines if the first of two `OracleDate` values is greater than the second
GreaterThanOrEqual	Determines if the first of two `OracleDate` values is greater than or equal to the second
LessThan	Determines if the first of two `OracleDate` values is less than the second
LessThanOrEqual	Determines if the first of two `OracleDate` values is less than or equal to the second
NotEquals	Determines if two `OracleDate` values are not equal
GetSysDate	Returns an `OracleDate` structure that represents the current date and time
Parse	Returns an `OracleDate` structure and sets its value using a string

OracleDate Static Operators

The `OracleDate` static operators are listed in Table 14-30.

Table 14-30 OracleDate Static Operators

Operator	Description
operator ==	Determines if two `OracleDate` values are the same
operator >	Determines if the first of two `OracleDate` values is greater than the second
operator >=	Determines if the first of two `OracleDate` values is greater than or equal to the second
operator !=	Determines if the two `OracleDate` values are not equal
operator <	Determines if the first of two `OracleDate` values is less than the second
operator <=	Determines if the first of two `OracleDate` values is less than or equal to the second

OracleDate Static Type Conversions

The `OracleDate` static type conversions are listed in Table 14-31.

Table 14-31 OracleDate Static Type Conversions

Operator	Description
explicit operator DateTime	Converts a structure to a `DateTime` structure

Table 14-31 (Cont.) OracleDate Static Type Conversions

Operator	Description
explicit operator OracleDate	Converts a structure to an `OracleDate` structure (Overloaded)

OracleDate Properties

The `OracleDate` properties are listed in Table 14-32.

Table 14-32 OracleDate Properties

Properties	Description
BinData	Gets an array of bytes that represents an Oracle `DATE` in Oracle internal format
Day	Gets the day component of an `OracleDate` method
IsNull	Indicates whether or not the current instance has a null value
Hour	Gets the `hour` component of an `OracleDate`
Minute	Gets the minute component of an `OracleDate`
Month	Gets the `month` component of an `OracleDate`
Second	Gets the `second` component of an `OracleDate`
Value	Gets the date and time that is stored in the `OracleDate` structure
Year	Gets the `year` component of an `OracleDate`

OracleDate Methods

The `OracleDate` methods are listed in Table 14-33.

Table 14-33 OracleDate Methods

Methods	Description
CompareTo	Compares the current `OracleDate` instance to an object, and returns an integer that represents their relative values
Equals	Determines whether or not an object has the same date and time as the current `OracleDate` instance (Overloaded)
GetHashCode	Returns a hash code for the `OracleDate` instance
GetDaysBetween	Calculates the number of days between the current `OracleDate` instance and an `OracleDate` structure
GetType	Inherited from `System.Object`
ToOracleTimeStamp	Converts the current `OracleDate` structure to an `OracleTimeStamp` structure
ToString	Converts the current `OracleDate` structure to a `string`

14.3.2 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.

14.3.2.1 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.

14.3.2.2 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);

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

14.3.2.3 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).

14.3.2.4 OracleDate(int, int, int, int, int, int)

This constructor creates a new instance of the `OracleDate` structure and set its value for time using the supplied year, month, day, hour, minute, and second.

Declaration

```
// C#
public OracleDate (int year, int month, int day, int hour, int minute, int second);
```

Parameters

- *`year`*

  The supplied year. Range of *`year`* is (-4712 to 9999).
- *`month`*

  The supplied month. Range of *`month`* is (1 to 12).
- *`day`*

  The supplied day. Range of *`day`* is (1 to 31).
- *`hour`*

  The supplied hour. Range of *`hour`* is (0 to 23).
- *`minute`*

The supplied minute. Range of *minute* is (0 to 59).

- *second*

  The supplied second. Range of *second* is (0 to 59).

Exceptions

`ArgumentOutOfRangeException` - The argument value for one or more of the parameters is out of the specified range.

`ArgumentException` - The argument values of the parameters cannot be used to construct a valid `OracleDate` (that is, the day is out of range for the month).

14.3.2.5 OracleDate(byte [ ])

This constructor creates a new instance of the `OracleDate` structure and sets its value to the provided byte array, which is in the internal Oracle `DATE` format.

Declaration

```
// C#
public OracleDate(byte [] bytes);
```

Parameters

- *bytes*

  A byte array that represents Oracle `DATE` in the internal Oracle `DATE` format.

Exceptions

`ArgumentException` - *bytes* is null or *bytes* is not in internal Oracle `DATE` format or `bytes` is not a valid Oracle `DATE`.

14.3.3 OracleDate Static Fields

The `OracleDate` static fields are listed in Table 14-34.

Table 14-34 OracleDate Static Fields

Field	Description
MaxValue	Represents the maximum valid date for an `OracleDate` structure, which is December 31, 9999 23:59:59
MinValue	Represents the minimum valid date for an `OracleDate` structure, which is January 1, -4712 0:0:0
Null	Represents a null value that can be assigned to the value of an `OracleDate` structure instance

14.3.3.1 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;
```

14.3.3.2 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;
```

14.3.3.3 Null

This static field represents a null value that can be assigned to the value of an `OracleDate` instance.

Declaration

```
// C#
public static readonly OracleDate Null;
```

14.3.4 OracleDate Static Methods

The `OracleDate` static methods are listed in Table 14-35.

Table 14-35 OracleDate Static Methods

Methods	Description
Equals	Determines if two `OracleDate` values are equal (Overloaded)
GreaterThan	Determines if the first of two `OracleDate` values is greater than the second
GreaterThanOrEqual	Determines if the first of two `OracleDate` values is greater than or equal to the second
LessThan	Determines if the first of two `OracleDate` values is less than the second
LessThanOrEqual	Determines if the first of two `OracleDate` values is less than or equal to the second
NotEquals	Determines if two `OracleDate` values are not equal
GetSysDate	Returns an `OracleDate` structure that represents the current date and time
Parse	Returns an `OracleDate` structure and sets its value using a string

14.3.4.1 Equals

Overloads `Object`

This method determines if two `OracleDate` values are equal.

Declaration

```
// 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.

- Any `OracleDate` that has a value compares greater than an `OracleDate` that has a null value.
- Two `OracleDate`s that contain a null value are equal.

14.3.4.2 GreaterThan

This method determines if the first of two `OracleDate` values is greater than the second.

Declaration

```
// C#
public static bool GreaterThan(OracleDate value1, OracleDate value2);
```

Parameters

- *`value1`*

  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 `OracleDate`s that contain a null value are equal.

14.3.4.3 GreaterThanOrEqual

This method determines if the first of two `OracleDate` values is greater than or equal to the second.

Declaration

```
// C#
public static bool GreaterThanOrEqual(OracleDate value1, OracleDate value2);
```

Parameters

- *`value1`*

  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`s that contain a null value are equal.

14.3.4.4 LessThan

This method determines if the first of two `OracleDate` values is less than the second.

Declaration

```
// C#
public static bool LessThan(OracleDate value1, OracleDate value2);
```

Parameters

- *`value1`*

  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`s that contain a null value are equal.

14.3.4.5 LessThanOrEqual

This method determines if the first of two `OracleDate` values is less than or equal to the second.

Declaration

```
// C#
public static bool LessThanOrEqual(OracleDate value1, OracleDate value2);
```

Parameters

- *`value1`*

  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 `OracleDate`s that contain a null value are equal.

14.3.4.6 NotEquals

This method determines if two `OracleDate` values are not equal.

Declaration

```
// C#
public static bool NotEquals(OracleDate value1, OracleDate value2);
```

Parameters

- *value1*

  The first `OracleDate`.
- *value2*

  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 `OracleDate`s that contain a null value are equal.

14.3.4.7 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.

14.3.4.8 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());
  }
}
```

14.3.5 OracleDate Static Operators

The `OracleDate` static operators are listed in Table 14-36.

Table 14-36 OracleDate Static Operators

Operator	Description
operator ==	Determines if two `OracleDate` values are the same
operator >	Determines if the first of two `OracleDate` values is greater than the second

Table 14-36 (Cont.) OracleDate Static Operators

Operator	Description
operator >=	Determines if the first of two `OracleDate` values is greater than or equal to the second
operator !=	Determines if the two `OracleDate` values are not equal
operator <	Determines if the first of two `OracleDate` values is less than the second
operator <=	Determines if the first of two `OracleDate` values is less than or equal to the second

14.3.5.1 operator ==

This method determines if two `OracleDate` values are the same.

Declaration

```
// 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 `OracleDate`s that contain a null value are equal.

14.3.5.2 operator >

This method determines if the first of two `OracleDate` values is greater than the second.

Declaration

```
// C#
public static bool operator > (OracleDate value1, OracleDate value2);
```

Parameters

- *`value1`*

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 `OracleDate`s that contain a null value are equal.

14.3.5.3 operator >=

This method determines if the first of two `OracleDate` values is greater than or equal to the second.

Declaration

```
// C#
public static bool operator >= (OracleDate value1, OracleDate value2);
```

Parameters

- *`value1`*

  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`s that contain a null value are equal.

14.3.5.4 operator !=

This method determines if the two `OracleDate` values are not equal.

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 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 `OracleDate`s that contain a null value are equal.

14.3.5.5 operator <

This method determines if the first of two `OracleDate` values is less than the second.

Declaration

```
// C#
public static bool operator < (OracleDate value1, OracleDate value2);
```

Parameters

- *value1*

  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`s that contain a null value are equal.

14.3.5.6 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 `OracleDate`s that contain a null value are equal.

14.3.6 OracleDate Static Type Conversions

The `OracleDate` static type conversions are listed in Table 14-37.

Table 14-37 OracleDate Static Type Conversions

Operator	Description
explicit operator DateTime	Converts a structure to a `DateTime` structure
explicit operator OracleDate	Converts a structure to an `OracleDate` structure (Overloaded)

14.3.6.1 explicit operator DateTime

This method converts an `OracleDate` structure to a `DateTime` structure.

Declaration

```
// C#
public static explicit operator DateTime(OracleDate val);
```

Parameters

- *val*

  An `OracleDate` structure.

Return Value

A `DateTime` structure.

14.3.6.2 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.

14.3.6.3 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.

14.3.6.4 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.

14.3.6.5 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());
  }
}
```

14.3.7 OracleDate Properties

The `OracleDate` properties are listed in Table 14-38.

Table 14-38 OracleDate Properties

Properties	Description
BinData	Gets an array of bytes that represents an Oracle `DATE` in Oracle internal format
Day	Gets the day component of an `OracleDate` method
IsNull	Indicates whether or not the current instance has a null value
Hour	Gets the `hour` component of an `OracleDate`
Minute	Gets the minute component of an `OracleDate`
Month	Gets the `month` component of an `OracleDate`
Second	Gets the `second` component of an `OracleDate`
Value	Gets the date and time that is stored in the `OracleDate` structure
Year	Gets the `year` component of an `OracleDate`

14.3.7.1 BinData

This property gets a array of bytes that represents an Oracle `DATE` in Oracle internal format.

Declaration

```
// C#
public byte[] BinData{get;}
```

Property Value

An array of bytes.

Exceptions

`OracleNullValueException` - `OracleDate` has a null value.

14.3.7.2 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.

14.3.7.3 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`.

14.3.7.4 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.

14.3.7.5 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.

14.3.7.6 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.

14.3.7.7 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.

14.3.7.8 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.

14.3.7.9 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.

14.3.8 OracleDate Methods

The `OracleDate` methods are listed in Table 14-39.

Table 14-39 OracleDate Methods

Methods	Description
CompareTo	Compares the current `OracleDate` instance to an object, and returns an integer that represents their relative values
Equals	Determines whether or not an object has the same date and time as the current `OracleDate` instance (Overloaded)

Table 14-39 (Cont.) OracleDate Methods

Methods	Description
GetHashCode	Returns a hash code for the `OracleDate` instance
GetDaysBetween	Calculates the number of days between the current `OracleDate` instance and an `OracleDate` structure
GetType	Inherited from `System.Object`
ToOracleTimeStamp	Converts the current `OracleDate` structure to an `OracleTimeStamp` structure
ToString	Converts the current `OracleDate` structure to a `string`

14.3.8.1 CompareTo

This method compares the current `OracleDate` instance to an object, and returns an integer that represents their relative values.

Declaration

```
// C#
public int CompareTo(object obj);
```

Parameters

- *obj*

  An object.

Return Value

The method returns:

- Less than zero: if the current `OracleDate` instance value is less than that of *obj*.
- Zero: if the current `OracleDate` instance and *obj* values are equal.
- Greater than zero: if the current `OracleDate` instance value is greater than *obj*.

Implements

`IComparable`

Exceptions

`ArgumentException` - The *obj* parameter is not an instance of `OracleDate`.

Remarks

The following rules apply to the behavior of this method.

- The comparison must be between `OracleDate`s. 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 `OracleDate`s that contain a null value are equal.

14.3.8.2 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 `OracleDate`s that contain a null value are equal.

14.3.8.3 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.

14.3.8.4 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.

14.3.8.5 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.

14.3.8.6 ToString

Overrides `ValueType`

This method converts the current `OracleDate` structure to a `string`.

Declaration

```
// C#
public override string ToString();
```

Return Value

A string.

Remarks

The returned value is a string representation of the `OracleDate` in the format specified by the thread's `OracleGlobalization.DateFormat` property. The names and abbreviations used for months and days are in the language specified by the thread's `OracleGlobalization.DateLanguage` and `OracleGlobalization.Calendar` properties. If any of the thread's globalization properties are set to null or an empty string, the client computer's settings are used.

Example

```
// C#

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

class ToStringSample
{
  static void Main(string[] args)
  {
    // Set the thread's DateFormat to a specific format
    OracleGlobalization info = OracleGlobalization.GetClientInfo();
    info.DateFormat = "YYYY-MON-DD";
    OracleGlobalization.SetThreadInfo(info);

    // Construct OracleDate from a string using the DateFormat specified
    OracleDate date = (OracleDate)"1999-DEC-01";

    // Set a different DateFormat on the thread for ToString()
    info.DateFormat = "YYYY/MM/DD";
    OracleGlobalization.SetThreadInfo(info);

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

14.4 OracleDecimal Structure

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

```
// C#
 public struct OracleDecimal : IComparable, INullable, IXmlSerializable
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver
Assembly	`Oracle.DataAccess.dll`	`Oracle.ManagedDataAccess.dll`
Namespace	`Oracle.DataAccess.Types`	`Oracle.ManagedDataAccess.Types`
.NET Framework	3.5, 4.5, 4.6	4.5, 4.6

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}",
      decimal3, decimal4);
  }
}
```

See Also:

- "Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"
- 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

14.4.1 OracleDecimal Members

`OracleDecimal` members are listed in the following tables:

OracleDecimal Constructors

`OracleDecimal` constructors are listed in Table 14-40

Table 14-40 OracleDecimal Constructors

Constructor	Description
OracleDecimal Constructors	Instantiates a new instance of `OracleDecimal` structure (Overloaded)

OracleDecimal Static Fields

The `OracleDecimal` static fields are listed in Table 14-41.

Table 14-41 OracleDecimal Static Fields

Field	Description
MaxPrecision	A constant representing the maximum precision, which is 38
MaxScale	A constant representing the maximum scale, which is 127

Table 14-41 (Cont.) OracleDecimal Static Fields

Field	Description
MaxValue	A constant representing the maximum value for this structure, which is 9.9…9 x 10^{125}
MinScale	A constant representing the minimum scale, which is -84
MinValue	A constant representing the minimum value for this structure, which is -1.0 x 10^{130}
NegativeOne	A constant representing the negative one value
Null	Represents a null value that can be assigned to an `OracleDecimal` instance
One	A constant representing the positive one value
Pi	A constant representing the numeric Pi value
Zero	A constant representing the zero value

OracleDecimal Static (Comparison) Methods

The `OracleDecimal` static (comparison) methods are listed in Table 14-42.

Table 14-42 OracleDecimal Static (Comparison) Methods

Methods	Description
Equals	Determines if two `OracleDecimal` values are equal (Overloaded)
GreaterThan	Determines if the first of two `OracleDecimal` values is greater than the second
GreaterThanOrEqual	Determines if the first of two `OracleDecimal` values is greater than or equal to the second
LessThan	Determines if the first of two `OracleDecimal` values is less than the second
LessThanOrEqual	Determines if the first of two `OracleDecimal` values is less than or equal to the second.
NotEquals	Determines if two `OracleDecimal` values are not equal

OracleDecimal Static (Manipulation) Methods

The `OracleDecimal` static (manipulation) methods are listed in Table 14-43.

Table 14-43 OracleDecimal Static (Manipulation) Methods

Methods	Description
Abs	Returns the absolute value of an `OracleDecimal`
Add	Adds two `OracleDecimal` structures
AdjustScale	Returns a new `OracleDecimal` with the specified number of digits and indicates whether or not to round or truncate the number if the scale is less than original

Table 14-43 (Cont.) OracleDecimal Static (Manipulation) Methods

Methods	Description
Ceiling	Returns a new `OracleDecimal` structure with its value set to the ceiling of an `OracleDecimal` structure
ConvertToPrecScale	Returns a new `OracleDecimal` structure with a new precision and scale
Divide	Divides one `OracleDecimal` value by another
Floor	Returns a new `OracleDecimal` structure with its value set to the floor of an `OracleDecimal` structure
Max	Returns the maximum value of the two supplied `OracleDecimal` structures
Min	Returns the minimum value of the two supplied `OracleDecimal` structures
Mod	Returns a new `OracleDecimal` structure with its value set to the modulus of two `OracleDecimal` structures
Multiply	Returns a new `OracleDecimal` structure with its value set to the result of multiplying two `OracleDecimal` structures
Negate	Returns a new `OracleDecimal` structure with its value set to the negation of the supplied `OracleDecimal` structure
Parse	Converts a `string` to an `OracleDecimal`
Round	Returns a new `OracleDecimal` structure with its value set to that of the supplied `OracleDecimal` structure and rounded off to the specified place
SetPrecision	Returns a new `OracleDecimal` structure with a new specified precision.
Shift	Returns a new `OracleDecimal` structure with its value set to that of the supplied `OracleDecimal` structure, and its decimal place shifted to the specified number of places to the right
Sign	Determines the sign of an `OracleDecimal` structure
Sqrt	Returns a new `OracleDecimal` structure with its value set to the square root of the supplied `OracleDecimal` structure
Subtract	Returns a new `OracleDecimal` structure with its value set to result of subtracting one `OracleDecimal` structure from another
Truncate	Truncates the `OracleDecimal` at a specified position

OracleDecimal Static (Logarithmic) Methods

The `OracleDecimal` static (logarithmic) methods are listed in Table 14-44.

Table 14-44 OracleDecimal Static (Logarithmic) Methods

Methods	Description
Exp	Returns a new `OracleDecimal` structure with its value set to e raised to the supplied power

Table 14-44 (Cont.) OracleDecimal Static (Logarithmic) Methods

Methods	Description
Log	Returns the supplied `OracleDecimal` structure with its value set to the logarithm of the supplied `OracleDecimal` structure (Overloaded)
Pow	Returns a new `OracleDecimal` structure with its value set to the supplied `OracleDecimal` structure raised to the supplied power (Overloaded)

OracleDecimal Static (Trigonometric) Methods

The `OracleDecimal` static (trigonometric) methods are listed in Table 14-45.

Table 14-45 OracleDecimal Static (Trigonometric) Methods

Methods	Description
Acos	Returns an angle in radians whose cosine is the supplied `OracleDecimal` structure
Asin	Returns an angle in radians whose sine is the supplied `OracleDecimal` structure
Atan	Returns an angle in radians whose tangent is the supplied `OracleDecimal` structure
Atan2	Returns an angle in radians whose tangent is the quotient of the two supplied `OracleDecimal` structures
Cos	Returns the cosine of the supplied angle in radians
Sin	Returns the sine of the supplied angle in radians
Tan	Returns the tangent of the supplied angle in radians
Cosh	Returns the hyperbolic cosine of the supplied angle in radians
Sinh	Returns the hyperbolic sine of the supplied angle in radians
Tanh	Returns the hyperbolic tangent of the supplied angle in radians

OracleDecimal Static (Comparison) Operators

The `OracleDecimal` static (comparison) operators are listed in Table 14-46.

Table 14-46 OracleDecimal Static (Comparison) Operators

Operator	Description
operator +	Adds two `OracleDecimal` values
operator /	Divides one `OracleDecimal` value by another
operator ==	Determines if the two `OracleDecimal` values are equal
operator >	Determines if the first of two `OracleDecimal` values is greater than the second
operator >=	Determines if the first of two `OracleDecimal` values is greater than or equal to the second

Table 14-46 (Cont.) OracleDecimal Static (Comparison) Operators

Operator	Description
operator !=	Determines if the two `OracleDecimal` values are not equal
operator <	Determines if the first of two `OracleDecimal` values is less than the second
operator <=	Determines if the first of two `OracleDecimal` values is less than or equal to the second
operator *	Multiplies two `OracleDecimal` structures
operator -	Subtracts one `OracleDecimal` structure from another
operator -	Negates an `OracleDecimal` structure
operator%	Returns a new `OracleDecimal` structure with its value set to the modulus of two `OracleDecimal` structures.

OracleDecimal Static Operators (Conversion from .NET Type to OracleDecimal)

The `OracleDecimal` static operators (Conversion from .NET Type to `OracleDecimal`) are listed in Table 14-47.

Table 14-47 OracleDecimal Static Operators (Conversion from .NET Type to OracleDecimal)

Operator	Description
implicit operator OracleDecimal	Converts an instance value to an `OracleDecimal` structure (Overloaded)
explicit operator OracleDecimal	Converts an instance value to an `OracleDecimal` structure (Overloaded)

OracleDecimal Static Operators (Conversion from OracleDecimal to .NET)

The `OracleDecimal` static operators (Conversion from `OracleDecimal` to .NET) are listed in Table 14-48.

Table 14-48 OracleDecimal Static Operators (Conversion from OracleDecimal to .NET)

Operator	Description
explicit operator byte	Returns the `byte` representation of the `OracleDecimal` value
explicit operator decimal	Returns the `decimal` representation of the `OracleDecimal` value
explicit operator double	Returns the `double` representation of the `OracleDecimal` value
explicit operator short	Returns the `short` representation of the `OracleDecimal` value
explicit operator int	Returns the `int` representation of the `OracleDecimal` value
explicit operator long	Returns the `long` representation of the `OracleDecimal` value
explicit operator float	Returns the `float` representation of the `OracleDecimal` value

OracleDecimal Properties

The `OracleDecimal` properties are listed in Table 14-49.

Table 14-49 OracleDecimal Properties

Properties	Description
BinData	Returns a byte array that represents the Oracle `NUMBER` in Oracle internal format
Format	Specifies the format for `ToString()`
IsInt	Indicates whether or not the current instance is an integer
IsNull	Indicates whether or not the current instance has a null value
IsPositive	Indicates whether or not the current instance is greater than `0`
IsZero	Indicates whether or not the current instance has a `zero` value
Value	Returns a `decimal` value

OracleDecimal Instance Methods

The `OracleDecimal` instance methods are listed in Table 14-50.

Table 14-50 OracleDecimal Instance Methods

Method	Description
CompareTo	Compares the current instance to the supplied object and returns an integer that represents their relative values
Equals	Determines whether 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)
GetHashCode	Returns a hash code for the current instance
GetType	Inherited from `System.Object`
ToByte	Returns the `byte` representation of the current instance
ToDouble	Returns the `double` representation of the current instance
ToInt16	Returns the `Int16` representation of the current instance
ToInt32	Returns the `Int32` representation of the current instance
ToInt64	Returns the `Int64` representation of the current instance
ToSingle	Returns the `Single` representation of the current instance
ToString	Overloads `Object.ToString()` Returns the `string` representation of the current instance

14.4.2 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.

14.4.2.1 OracleDecimal(byte [ ])

This constructor creates a new instance of the `OracleDecimal` structure and sets its value to the supplied byte array, which is in an Oracle `NUMBER` format.

Declaration

```
// C#
public OracleDecimal(byte [] bytes);
```

Parameters

- *`bytes`*

  A byte array that represents an Oracle `NUMBER` in an internal Oracle format.

Exceptions

`ArgumentException` - The *`bytes`* parameter is not in a internal Oracle `NUMBER` format or *`bytes`* has an invalid value.

`ArgumentNullException` - The *`bytes`* parameter is null.

14.4.2.2 OracleDecimal(decimal)

This constructor creates a new instance of the `OracleDecimal` structure and sets its value to the supplied `Decimal` value.

Declaration

```
// C#
public OracleDecimal(decimal decX);
```

Parameters

- *`decX`*

  The provided `Decimal` value.

14.4.2.3 OracleDecimal(double)

This constructor creates a new instance of the `OracleDecimal` structure and sets its value to the supplied `double` value.

Declaration

```
// C#
public OracleDecimal(double doubleX)
```

Parameters

- *`doubleX`*

  The provided double value.

Exceptions

`OverFlowException` - The value of the supplied `double` is greater than the maximum value or less than the minimum value of `OracleDecimal`.

Remarks

`OracleDecimal` contains the following values depending on the provided double value:

- `double.PositiveInfinity`: positive infinity value
- `double.NegativeInfinity`: negative infinity value.
- `double.NaN`: null value

14.4.2.4 OracleDecimal(int)

This constructor creates a new instance of the `OracleDecimal` structure and sets its value to the supplied `Int32` value.

Declaration

```
// C#
public OracleDecimal(int intX);
```

Parameters

- *intX*

  The provided Int32 value.

14.4.2.5 OracleDecimal(float)

This constructor creates a new instance of the OracleDecimal structure and sets its value to the supplied Single value.

Declaration

```
// C#
public OracleDecimal(float floatX);
```

Parameters

- *floatX*

  The provided float value.

Remarks

OracleDecimal contains the following values depending on the provided float value:

float.PositiveInfinity: positive infinity value

float.NegativeInfinity: negative infinity value

float.NaN: null value

14.4.2.6 OracleDecimal(long)

This constructor creates a new instance of the OracleDecimal structure and sets its value to the supplied Int64 value.

Declaration

```
// C#
public OracleDecimal(long longX);
```

Parameters

- *longX*

  The provided Int64 value.

14.4.2.7 OracleDecimal(string)

This constructor creates a new instance of the OracleDecimal structure and sets its value to the supplied string value.

Declaration

```
// C#
public OracleDecimal(string numStr);
```

Parameters

- *`numStr`*

  The provided `string` value.

Exceptions

`ArgumentException` - The *`numStr`* parameter is an invalid string representation of an `OracleDecimal`.

`ArgumentNullException` - The *`numStr`* parameter is null.

`OverFlowException` - The value of *`numStr`* is greater than the maximum value or less than the minimum value of `OracleDecimal`.

`input string format is incorrect` - The locale's numeric separator is a comma(`,`).

14.4.2.8 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());
  }
}
```

14.4.3 OracleDecimal Static Fields

The `OracleDecimal` static fields are listed in Table 14-51.

Table 14-51 OracleDecimal Static Fields

Field	Description
MaxPrecision	A constant representing the maximum precision, which is 38
MaxScale	A constant representing the maximum scale, which is 127
MaxValue	A constant representing the maximum value for this structure, which is 9.9…9 x 10^{125}
MinScale	A constant representing the minimum scale, which is -84
MinValue	A constant representing the minimum value for this structure, which is -1.0 x 10^{130}
NegativeOne	A constant representing the negative one value
Null	Represents a null value that can be assigned to an `OracleDecimal` instance
One	A constant representing the positive one value
Pi	A constant representing the numeric Pi value
Zero	A constant representing the zero value

14.4.3.1 MaxPrecision

This static field represents the maximum precision, which is 38.

Declaration

```
// C#
public static readonly byte MaxPrecision;
```

14.4.3.2 MaxScale

This static field a constant representing the maximum scale, which is 127.

Declaration

```
// C#
public static readonly byte MaxScale;
```

14.4.3.3 MaxValue

This static field indicates a constant representing the maximum value for this structure, which is 9.9…9 x 10^{125} (38 nines followed by 88 zeroes).

Declaration

```
// C#
public static readonly OracleDecimal MaxValue;
```

14.4.3.4 MinScale

This static field indicates a constant representing the maximum scale, which is -84.

Declaration

```
// C#
public static readonly int MinScale;
```

14.4.3.5 MinValue

This static field indicates a constant representing the minimum value for this structure, which is -1.0 x 10^{130}.

Declaration

```
// C#
public static readonly OracleDecimal MinValue;
```

14.4.3.6 NegativeOne

This static field indicates a constant representing the negative one value.

Declaration

```
// C#
public static readonly OracleDecimal NegativeOne;
```

14.4.3.7 Null

This static field represents a null value that can be assigned to an `OracleDecimal` instance.

Declaration

```
// C#
public static readonly OracleDecimal Null;
```

14.4.3.8 One

This static field indicates a constant representing the positive one value.

Declaration

```
// C#
public static readonly OracleDecimal One;
```

14.4.3.9 Pi

This static field indicates a constant representing the numeric Pi value.

Declaration

```
// C#
public static readonly OracleDecimal Pi;
```

14.4.3.10 Zero

This static field indicates a constant representing the zero value.

Declaration

```
// C#
public static readonly OracleDecimal Zero;
```

14.4.4 OracleDecimal Static (Comparison) Methods

The `OracleDecimal` static (comparison) methods are listed in Table 14-52.

Table 14-52 OracleDecimal Static (Comparison) Methods

Methods	Description
Equals	Determines if two `OracleDecimal` values are equal (Overloaded)
GreaterThan	Determines if the first of two `OracleDecimal` values is greater than the second
GreaterThanOrEqual	Determines if the first of two `OracleDecimal` values is greater than or equal to the second

Table 14-52 (Cont.) OracleDecimal Static (Comparison) Methods

Methods	Description
LessThan	Determines if the first of two `OracleDecimal` values is less than the second
LessThanOrEqual	Determines if the first of two `OracleDecimal` values is less than or equal to the second.
NotEquals	Determines if two `OracleDecimal` values are not equal

14.4.4.1 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 `OracleDecimal`s that contain a null value are equal.

14.4.4.2 GreaterThan

This method determines if the first of two `OracleDecimal` values is greater than the second.

Declaration

```
// C#
public static bool GreaterThan(OracleDecimal value1, OracleDecimal value2);
```

Parameters

- *`value1`*

  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 `OracleDecimal`s that contain a null value are equal.

14.4.4.3 GreaterThanOrEqual

This method determines if the first of two `OracleDecimal` values is greater than or equal to the second.

Declaration

```
// C#
public static bool GreaterThanOrEqual(OracleDecimal value1, OracleDecimal value2);
```

Parameters

- *value1*

  The first `OracleDecimal`.
- *value2*

  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 `OracleDecimal`s that contain a null value are equal.

14.4.4.4 LessThan

This method determines if the first of two `OracleDecimal` values is less than the second.

Declaration

```
// C#
public static bool LessThan(OracleDecimal value1, OracleDecimal value2);
```

Parameters

- *`value1`*

  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 `OracleDecimal`s that contain a null value are equal.

14.4.4.5 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.

14.4.4.6 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 `OracleDecimal`s that contain a null value are equal.

14.4.5 OracleDecimal Static (Manipulation) Methods

The `OracleDecimal` static (manipulation) methods are listed in Table 14-53.

Table 14-53 OracleDecimal Static (Manipulation) Methods

Methods	Description
Abs	Returns the absolute value of an `OracleDecimal`
Add	Adds two `OracleDecimal` structures
AdjustScale	Returns a new `OracleDecimal` with the specified number of digits and indicates whether or not to round or truncate the number if the scale is less than original
Ceiling	Returns a new `OracleDecimal` structure with its value set to the ceiling of an `OracleDecimal` structure
ConvertToPrecScale	Returns a new `OracleDecimal` structure with a new precision and scale
Divide	Divides one `OracleDecimal` value by another
Floor	Returns a new `OracleDecimal` structure with its value set to the floor of an `OracleDecimal` structure
Max	Returns the maximum value of the two supplied `OracleDecimal` structures

Table 14-53 (Cont.) OracleDecimal Static (Manipulation) Methods

Methods	Description
Min	Returns the minimum value of the two supplied `OracleDecimal` structures
Mod	Returns a new `OracleDecimal` structure with its value set to the modulus of two `OracleDecimal` structures
Multiply	Returns a new `OracleDecimal` structure with its value set to the result of multiplying two `OracleDecimal` structures
Negate	Returns a new `OracleDecimal` structure with its value set to the negation of the supplied `OracleDecimal` structure
Parse	Converts a `string` to an `OracleDecimal`
Round	Returns a new `OracleDecimal` structure with its value set to that of the supplied `OracleDecimal` structure and rounded off to the specified place
SetPrecision	Returns a new `OracleDecimal` structure with a new specified precision.
Shift	Returns a new `OracleDecimal` structure with its value set to that of the supplied `OracleDecimal` structure, and its decimal place shifted to the specified number of places to the right
Sign	Determines the sign of an `OracleDecimal` structure
Sqrt	Returns a new `OracleDecimal` structure with its value set to the square root of the supplied `OracleDecimal` structure
Subtract	Returns a new `OracleDecimal` structure with its value set to result of subtracting one `OracleDecimal` structure from another
Truncate	Truncates the `OracleDecimal` at a specified position

14.4.5.1 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.

14.4.5.2 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.

14.4.5.3 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());
  }
}
```

14.4.5.4 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.

14.4.5.5 ConvertToPrecScale

This method returns a new `OracleDecimal` structure with a new precision and scale.

Declaration

```
// C#
public static OracleDecimal ConvertToPrecScale(OracleDecimal val
    int precision, int scale);
```

Parameters

- *`val`*

  An `OracleDecimal` structure.

- *`precision`*

  The precision. Range of precision is 1 to 38.

- *`scale`*

  The number of digits to the right of the decimal point. Range of scale is -84 to 127.

Return Value

A new `OracleDecimal` structure.

Remarks

If the supplied `OracleDecimal` has a null value, the returned `OracleDecimal` has a null value.

Example

```
// C#

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());
  }
}
```

14.4.5.6 Divide

This method divides one `OracleDecimal` value by another.

Declaration

```
// C#
public static OracleDecimal Divide(OracleDecimal val1, OracleDecimal val2);
```

Parameters

- *`val1`*

  An `OracleDecimal`.

- *`val2`*

  An `OracleDecimal`.

Return Value

A new `OracleDecimal` structure.

Remarks

If either argument has a null value, the returned `OracleDecimal` has a null value.

14.4.5.7 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.

14.4.5.8 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.

14.4.5.9 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.

14.4.5.10 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.

14.4.5.11 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.

14.4.5.12 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.

14.4.5.13 Parse

This method converts a `string` to an `OracleDecimal`.

Declaration

```
// C#
public static OracleDecimal Parse (string str);
```

Parameters

- *`str`*

  The string being converted.

Return Value

A new `OracleDecimal` structure.

Exceptions

`ArgumentException` - The *`numStr`* parameter is an invalid string representation of an `OracleDecimal`.

`ArgumentNullException` - The *`numStr`* parameter is null.

`OverFlowException` - The value of *`numStr`* is greater than the maximum value or less than the minimum value of `OracleDecimal`.

14.4.5.14 Round

This method returns a new `OracleDecimal` structure with its value set to that of the supplied `OracleDecimal` structure and rounded off to the specified place.

Declaration

```
// C#
public static OracleDecimal Round(OracleDecimal val, int decplace);
```

Parameters

- *`val`*

  An `OracleDecimal` structure.
- *`decplace`*

  The specified decimal place. If the value is positive, the function rounds the `OracleDecimal` structure to the right of the decimal point. If the value is negative, the function rounds to the left of the decimal point.

Return Value

An `OracleDecimal` structure.

Remarks

If the supplied `OracleDecimal` structure has a null value, the returned `OracleDecimal` has a null value.

14.4.5.15 SetPrecision

This method returns a new `OracleDecimal` structure with a new specified precision.

Declaration

```
// C#
public static OracleDecimal SetPrecision(OracleDecimal val, int precision);
```

Parameters

- *`val`*

  An `OracleDecimal` structure.

- *`precision`*

  The specified precision. Range of precision is 1 to 38.

Return Value

An `OracleDecimal` structure.

Remarks

The returned `OracleDecimal` is rounded off if the specified precision is smaller than the precision of *`val`*.

If *`val`* has a null value, the returned `OracleDecimal` has a null value.

Example

```
// C#

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());
    }
}
```

14.4.5.16 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.

14.4.5.17 Sign

This method determines the sign of an `OracleDecimal` structure.

Declaration

```
// C#
public static int Sign(OracleDecimal val);
```

Parameters

- *`val`*

  An `OracleDecimal` structure.

Return Value

- `-1`: if the supplied `OracleDecimal` < 0
- `0`: if the supplied `OracleDecimal` == 0
- `1`: if the supplied `OracleDecimal` > 0

Exceptions

`OracleNullValueException` - The argument has a null value.

14.4.5.18 Sqrt

This method returns a new `OracleDecimal` structure with its value set to the square root of the supplied `OracleDecimal` structure.

Declaration

```
// C#
public static OracleDecimal Sqrt(OracleDecimal val);
```

Parameters

- *`val`*

  An `OracleDecimal` structure.

Return Value

An `OracleDecimal` structure.

Exceptions

`ArgumentOutOfRangeException` - The provided `OracleDecimal` structure is less than zero.

Remarks

If either argument has a null value, the returned `OracleDecimal` has a null value.

14.4.5.19 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.

14.4.5.20 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.

14.4.6 OracleDecimal Static (Logarithmic) Methods

The `OracleDecimal` static (logarithmic) methods are listed in Table 14-54.

Table 14-54 OracleDecimal Static (Logarithmic) Methods

Methods	Description
Exp	Returns a new `OracleDecimal` structure with its value set to e raised to the supplied power
Log	Returns the supplied `OracleDecimal` structure with its value set to the logarithm of the supplied `OracleDecimal` structure (Overloaded)
Pow	Returns a new `OracleDecimal` structure with its value set to the supplied `OracleDecimal` structure raised to the supplied power (Overloaded)

14.4.6.1 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.

14.4.6.2 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.

14.4.6.3 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.

14.4.6.4 Log(OracleDecimal, int)

This method returns the supplied `OracleDecimal` structure with its value set to the logarithm of the supplied `OracleDecimal` structure in the supplied base.

Declaration

```
// C#
public static OracleDecimal Log(OracleDecimal val, int logBase);
```

Parameters

- *`val`*

  An `OracleDecimal` structure whose logarithm is to be calculated.
- *`logBase`*

  An `int` that specifies the base of the logarithm.

Return Value

A new `OracleDecimal` structure with its value set to the logarithm of *`val`* in the supplied base.

Exceptions

`ArgumentOutOfRangeException` - Either argument is less than zero.

Remarks

If either argument has a null value, the returned `OracleDecimal` has a null value.

If both arguments have zero value, the result is undefined, and the returned `OracleDecimal` structure has a null value.

14.4.6.5 Log(OracleDecimal, OracleDecimal)

This method returns the supplied `OracleDecimal` structure with its value set to the logarithm of the supplied `OracleDecimal` structure in the supplied base.

Declaration

```
// C#
public static OracleDecimal Log(OracleDecimal val, OracleDecimal logBase);
```

Parameters

- *`val`*

  An `OracleDecimal` structure whose logarithm is to be calculated.
- *`logBase`*

  An `OracleDecimal` structure that specifies the base of the logarithm.

Return Value

Returns the logarithm of *`val`* in the supplied base.

Exceptions

`ArgumentOutOfRangeException` - Either the *`val`* or *`logBase`* parameter is less than zero.

Remarks

If either argument has a null value, the returned `OracleDecimal` has a null value.

If both arguments have zero value, the result is undefined, and the returned `OracleDecimal` structure has a null value.

14.4.6.6 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.

14.4.6.7 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.

14.4.6.8 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.

14.4.7 OracleDecimal Static (Trigonometric) Methods

The `OracleDecimal` static (trigonometric) methods are listed in Table 14-55.

Table 14-55 OracleDecimal Static (Trigonometric) Methods

Methods	Description
Acos	Returns an angle in radians whose cosine is the supplied `OracleDecimal` structure
Asin	Returns an angle in radians whose sine is the supplied `OracleDecimal` structure
Atan	Returns an angle in radians whose tangent is the supplied `OracleDecimal` structure
Atan2	Returns an angle in radians whose tangent is the quotient of the two supplied `OracleDecimal` structures
Cos	Returns the cosine of the supplied angle in radians
Sin	Returns the sine of the supplied angle in radians
Tan	Returns the tangent of the supplied angle in radians
Cosh	Returns the hyperbolic cosine of the supplied angle in radians
Sinh	Returns the hyperbolic sine of the supplied angle in radians
Tanh	Returns the hyperbolic tangent of the supplied angle in radians

14.4.7.1 Acos

This method returns an angle in radians whose cosine is the supplied `OracleDecimal` structure.

Declaration

```
// C#
public static OracleDecimal Acos(OracleDecimal val);
```

Parameters

- *`val`*

  An `OracleDecimal` structure. Range is (-1 to 1).

Return Value

An `OracleDecimal` structure that represents an angle in radians.

Remarks

If either argument has a null value, the returned `OracleDecimal` has a null value.

14.4.7.2 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.

14.4.7.3 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.

14.4.7.4 Atan2

This method returns an angle in radians whose tangent is the quotient of the two supplied `OracleDecimal` structures.

Declaration

```
// C#
public static OracleDecimal Atan2(OracleDecimal val1, OracleDecimal val2);
```

Parameters

- *`val1`*

  An `OracleDecimal` structure that represents the y-coordinate.

- *val2*

  An `OracleDecimal` structure that represents the x-coordinate.

Return Value

An `OracleDecimal` structure that represents an angle in radians.

Remarks

If either argument has a null value, the returned `OracleDecimal` has a null value.

14.4.7.5 Cos

This method returns the cosine of the supplied angle in radians.

Declaration

```
// 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.

14.4.7.6 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.

14.4.7.7 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.

14.4.7.8 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.

14.4.7.9 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.

14.4.7.10 Tanh

This method returns the hyperbolic tangent of the supplied angle in radians.

Declaration

```
// 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.

14.4.8 OracleDecimal Static (Comparison) Operators

The `OracleDecimal` static (comparison) operators are listed in Table 14-56.

Table 14-56 OracleDecimal Static (Comparison) Operators

Operator	Description
operator +	Adds two `OracleDecimal` values

Table 14-56 (Cont.) OracleDecimal Static (Comparison) Operators

Operator	Description
operator /	Divides one `OracleDecimal` value by another
operator ==	Determines if the two `OracleDecimal` values are equal
operator >	Determines if the first of two `OracleDecimal` values is greater than the second
operator >=	Determines if the first of two `OracleDecimal` values is greater than or equal to the second
operator !=	Determines if the two `OracleDecimal` values are not equal
operator <	Determines if the first of two `OracleDecimal` values is less than the second
operator <=	Determines if the first of two `OracleDecimal` values is less than or equal to the second
operator *	Multiplies two `OracleDecimal` structures
operator -	Subtracts one `OracleDecimal` structure from another
operator -	Negates an `OracleDecimal` structure
operator%	Returns a new `OracleDecimal` structure with its value set to the modulus of two `OracleDecimal` structures.

14.4.8.1 operator +

This method adds two `OracleDecimal` values.

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.

14.4.8.2 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.

14.4.8.3 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 `OracleDecimal`s that contain a null value are equal.

14.4.8.4 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 `OracleDecimal`s that contain a null value are equal.

14.4.8.5 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 `OracleDecimal`s that contain a null value are equal.

14.4.8.6 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 `OracleDecimal`s that contain a null value are equal.

14.4.8.7 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 `OracleDecimal`s that contain a null value are equal.

14.4.8.8 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.

14.4.8.9 operator *

This method multiplies two `OracleDecimal` structures.

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.

14.4.8.10 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.

14.4.8.11 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.

14.4.8.12 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.

14.4.9 OracleDecimal Static Operators (Conversion from .NET Type to OracleDecimal)

The `OracleDecimal` static operators (Conversion from .NET Type to `OracleDecimal`) are listed in Table 14-57.

Table 14-57 OracleDecimal Static Operators (Conversion from .NET Type to OracleDecimal)

Operator	Description
implicit operator OracleDecimal	Converts an instance value to an `OracleDecimal` structure (Overloaded)
explicit operator OracleDecimal	Converts an instance value to an `OracleDecimal` structure (Overloaded)

14.4.9.1 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.

14.4.9.2 implicit operator OracleDecimal(decimal)

This method returns the `OracleDecimal` representation of a `decimal` value.

Declaration

```
// C#
public static implicit operator OracleDecimal(decimal val);
```

Parameters

- *`val`*

  A `decimal` value.

Return Value

An `OracleDecimal`.

14.4.9.3 implicit operator OracleDecimal(int)

This method returns the `OracleDecimal` representation of an `int` value.

Declaration

```
// C#
public static implicit operator OracleDecimal(int val);
```

Parameters

- *`val`*

  An `int` value.

Return Value

An `OracleDecimal`.

14.4.9.4 implicit operator OracleDecimal(long)

This method returns the `OracleDecimal` representation of a `long` value.

Declaration

```
// C#
public static implicit operator OracleDecimal(long val);
```

Parameters

- *`val`*

A `long` value.

Return Value

An `OracleDecimal`.

14.4.9.5 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.

14.4.9.6 explicit operator OracleDecimal(double)

This method returns the `OracleDecimal` representation of a double.

Declaration

```
// C#
public static explicit operator OracleDecimal(double val);
```

Parameters

- *`val`*

  A `double`.

Return Value

An `OracleDecimal`.

Exceptions

`OverFlowException` - The value of the supplied `double` is greater than the maximum value of `OracleDecimal` or less than the minimum value of `OracleDecimal`.

Remarks

`OracleDecimal` contains the following values depending on the provided double value:

- `double.PositiveInfinity`: positive infinity value
- `double.NegativeInfinity`: negative infinity value.
- `double.NaN`: null value

14.4.9.7 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`.

14.4.10 OracleDecimal Static Operators (Conversion from OracleDecimal to .NET)

The `OracleDecimal` static operators (Conversion from `OracleDecimal` to .NET) are listed in Table 14-58.

Table 14-58 OracleDecimal Static Operators (Conversion from OracleDecimal to .NET)

Operator	Description
explicit operator byte	Returns the `byte` representation of the `OracleDecimal` value
explicit operator decimal	Returns the `decimal` representation of the `OracleDecimal` value
explicit operator double	Returns the `double` representation of the `OracleDecimal` value
explicit operator short	Returns the `short` representation of the `OracleDecimal` value
explicit operator int	Returns the `int` representation of the `OracleDecimal` value
explicit operator long	Returns the `long` representation of the `OracleDecimal` value
explicit operator float	Returns the `float` representation of the `OracleDecimal` value

14.4.10.1 explicit operator byte

This method returns the `byte` representation of the `OracleDecimal` value.

Declaration

```
// C#
public static explicit operator byte(OracleDecimal val);
```

Parameters

- *val*

  An OracleDecimal structure.

Return Value

A byte.

Exceptions

OracleNullValueException - OracleDecimal has a null value.

OverFlowException- The byte cannot represent the supplied OracleDecimal structure.

14.4.10.2 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.

14.4.10.3 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.

14.4.10.4 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.

14.4.10.5 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.

14.4.10.6 explicit operator long

This method returns the `long` representation of the `OracleDecimal` value.

Declaration

```
// C#
public static explicit operator long(OracleDecimal val);
```

Parameters

- *`val`*

  An `OracleDecimal` structure.

Return Value

A `long`.

Exceptions

`OracleNullValueException` - The `OracleDecimal` has a null value.

`OverFlowException` - The `long` cannot represent the supplied `OracleDecimal` structure.

14.4.10.7 explicit operator float

This method returns the `float` representation of the `OracleDecimal` value.

Declaration

```
// C#
public static explicit operator float(OracleDecimal val);
```

Parameters

- *`val`*

  An `OracleDecimal` structure.

Return Value

A `float`.

Exceptions

`OracleNullValueException` - The `OracleDecimal` has a null value.

`OverFlowException` - The `float` cannot represent the supplied `OracleDecimal` structure.

14.4.11 OracleDecimal Properties

The `OracleDecimal` properties are listed in Table 14-59.

Table 14-59 OracleDecimal Properties

Properties	Description
BinData	Returns a byte array that represents the Oracle `NUMBER` in Oracle internal format
Format	Specifies the format for `ToString()`
IsInt	Indicates whether or not the current instance is an integer
IsNull	Indicates whether or not the current instance has a null value
IsPositive	Indicates whether or not the current instance is greater than `0`
IsZero	Indicates whether or not the current instance has a `zero` value
Value	Returns a `decimal` value

14.4.11.1 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.

14.4.11.2 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`

14.4.11.3 IsInt

This property indicates whether or not the current instance is an integer value.

Declaration

```
// C#
public bool IsInt {get;}
```

Property Value

A `bool` value that returns `true` if the current instance is an integer value; otherwise, returns `false`.

Exceptions

`OracleNullValueException` - The current instance has a null value.

14.4.11.4 IsNull

This property indicates whether or not the current instance has a null value.

Declaration

```
// C#
public bool IsNull {get;}
```

Property Value

A `bool` value that returns `true` if the current instance has a null value; otherwise, returns `false`.

14.4.11.5 IsPositive

This property indicates whether or not the value of the current instance is greater than `0`.

Declaration

```
// C#
public bool IsPositive {get;}
```

Property Value

A `bool` value that returns `true` if the current instance is greater than `0`; otherwise, returns `false`.

Exceptions

`OracleNullValueException` - The current instance has a null value.

14.4.11.6 IsZero

This property indicates whether or not the current instance has a zero value.

Declaration

```
// C#
public bool IsZero{get;}
```

Property Value

A `bool` value that returns `true` if the current instance has a zero value; otherwise, returns `false`.

Exceptions

`OracleNullValueException` - The current instance has a null value.

14.4.11.7 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" for further information.

14.4.12 OracleDecimal Instance Methods

The `OracleDecimal` instance methods are listed in Table 14-60.

Table 14-60 OracleDecimal Instance Methods

Method	Description
CompareTo	Compares the current instance to the supplied object and returns an integer that represents their relative values
Equals	Determines whether 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)
GetHashCode	Returns a hash code for the current instance
GetType	Inherited from `System.Object`
ToByte	Returns the `byte` representation of the current instance
ToDouble	Returns the `double` representation of the current instance
ToInt16	Returns the `Int16` representation of the current instance
ToInt32	Returns the `Int32` representation of the current instance
ToInt64	Returns the `Int64` representation of the current instance
ToSingle	Returns the `Single` representation of the current instance
ToString	Overloads `Object.ToString()` Returns the `string` representation of the current instance

14.4.12.1 CompareTo

This method compares the current instance to the supplied object and returns an integer that represents their relative values.

Declaration

```
// C#
public int CompareTo(object obj);
```

Parameters

- *`obj`*

  The supplied instance.

Return Value

The method returns a number:

- Less than zero: if the value of the current instance is less than *`obj`*.
- Zero: if the value of the current instance is equal to *`obj`*.
- Greater than zero: if the value of the current instance is greater than *`obj`*.

Implements

`IComparable`

Exceptions

`ArgumentException` - The parameter is not of type `OracleDecimal`.

Remarks

The following rules apply to the behavior of this method.

- The comparison must be between `OracleDecimal`s. 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 `OracleDecimal`s that contain a null value are equal.

14.4.12.2 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 `OracleDecimal`s that contain a null value are equal.

14.4.12.3 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.

14.4.12.4 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.

14.4.12.5 ToDouble

This method returns the `double` representation of the current instance.

Declaration

```
// C#
public double ToDouble();
```

Return Value

A `double`.

Exceptions

`OverFlowException` - The `double` cannot represent the current instance.

`OracleNullValueException` - The current instance has a null value.

14.4.12.6 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.

14.4.12.7 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.

14.4.12.8 ToInt64

This method returns the `Int64` representation of the current instance.

Declaration

```
// C#
public long ToInt64();
```

Return Value

A `long`.

Exceptions

`OverFlowException` - The `long` cannot represent the current instance.

`OracleNullValueException` - The current instance has a null value.

14.4.12.9 ToSingle

This method returns the `Single` representation of the current instance.

Declaration

```
// C#
public float ToSingle();
```

Return Value

A `float`.

Exceptions

`OverFlowException` - The `float` cannot represent the current instance.

`OracleNullValueException` - The current instance has a null value.

14.4.12.10 ToString

Overrides `Object`

This method returns the `string` representation of the current instance.

Declaration

```
// C#
public override string ToString();
```

Return Value

Returns the number in a string returns and a period (`.`) as a numeric separator.

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.

14.5 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

```
// C#
public struct OracleIntervalDS : IComparable, INullable, IXmlSerializable
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver
Assembly	`Oracle.DataAccess.dll`	`Oracle.ManagedDataAccess.dll`
Namespace	`Oracle.DataAccess.Types`	`Oracle.ManagedDataAccess.Types`
.NET Framework	3.5, 4.5, 4.6	4.5, 4.6

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());
  }
}
```

See Also:

- "Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"
- OracleIntervalDS Members
- OracleIntervalDS Constructors
- OracleIntervalDS Static Fields
- OracleIntervalDS Static Methods
- OracleIntervalDS Static Operators
- OracleIntervalDS Type Conversions
- OracleIntervalDS Properties
- OracleIntervalDS Methods

14.5.1 OracleIntervalDS Members

`OracleIntervalDS` members are listed in the following tables:

OracleIntervalDS Constructors

`OracleIntervalDS` constructors are listed in Table 14-61

Table 14-61 OracleIntervalDS Constructors

Constructor	Description
OracleIntervalDS Constructors	Instantiates a new instance of `OracleIntervalDS` structure (Overloaded)

OracleIntervalDS Static Fields

The `OracleIntervalDS` static fields are listed in Table 14-62.

Table 14-62 OracleIntervalDS Static Fields

Field	Description
MaxValue	Represents the maximum valid time interval for an `OracleIntervalDS` structure
MinValue	Represents the minimum valid time interval for an `OracleIntervalDS` structure
Null	Represents a null value that can be assigned to an `OracleIntervalDS` instance
Zero	Represents a zero value for an `OracleIntervalDS` structure

OracleIntervalDS Static Methods

The `OracleIntervalDS` static methods are listed in Table 14-63.

Table 14-63 OracleIntervalDS Static Methods

Methods	Description
Equals	Determines whether or not two `OracleIntervalDS` values are equal (Overloaded)
GreaterThan	Determines whether or not one `OracleIntervalDS` value is greater than another
GreaterThanOrEqual	Determines whether or not one `OracleIntervalDS` value is greater than or equal to another
LessThan	Determines whether or not one `OracleIntervalDS` value is less than another
LessThanOrEqual	Determines whether or not one `OracleIntervalDS` value is less than or equal to another
NotEquals	Determines whether or not two `OracleIntervalDS` values are not equal
Parse	Returns an `OracleIntervalDS` structure and sets its value for time interval using a string
SetPrecision	Returns a new instance of an `OracleIntervalDS` with the specified day precision and fractional second precision

OracleIntervalDS Static Operators

The `OracleIntervalDS` static operators are listed in Table 14-64.

Table 14-64 OracleIntervalDS Static Operators

Operator	Description
operator +	Adds two `OracleIntervalDS` values
operator ==	Determines whether or not two `OracleIntervalDS` values are equal
operator >	Determines whether or not one `OracleIntervalDS` value is greater than another
operator >=	Determines whether or not one `OracleIntervalDS` value is greater than or equal to another
operator !=	Determines whether or not two `OracleIntervalDS` values are not equal
operator <	Determines whether or not one `OracleIntervalDS` value is less than another
operator <=	Determines whether or not one `OracleIntervalDS` value is less than or equal to another
operator -	Subtracts one `OracleIntervalDS` value from another
operator -	Negates an `OracleIntervalDS` structure

Table 14-64 (Cont.) OracleIntervalDS Static Operators

Operator	Description
operator *	Multiplies an `OracleIntervalDS` value by a number
operator /	Divides an `OracleIntervalDS` value by a number

OracleIntervalDS Type Conversions

The `OracleIntervalDS` type conversions are listed in Table 14-65.

Table 14-65 OracleIntervalDS Type Conversions

Operator	Description
explicit operator TimeSpan	Converts an `OracleIntervalDS` structure to a `TimeSpan` structure
explicit operator OracleIntervalDS	Converts a string to an `OracleIntervalDS` structure
implicit operator OracleIntervalDS	Converts a `TimeSpan` structure to an `OracleIntervalDS` structure

OracleIntervalDS Properties

The `OracleIntervalDS` properties are listed in Table 14-66.

Table 14-66 OracleIntervalDS Properties

Properties	Description
BinData	Returns an array of bytes that represents the Oracle `INTERVAL DAY TO SECOND` in Oracle internal format
Days	Gets the days component of an `OracleIntervalDS`
Hours	Gets the hours component of an `OracleIntervalDS`
IsNull	Indicates whether or not the current instance has a null value
Milliseconds	Gets the milliseconds component of an `OracleIntervalDS`
Minutes	Gets the minutes component of an `OracleIntervalDS`
Nanoseconds	Gets the nanoseconds component of an `OracleIntervalDS`
Seconds	Gets the seconds component of an `OracleIntervalDS`
TotalDays	Returns the total number, in days, that represent the time period in the `OracleIntervalDS` structure
Value	Specifies the time interval that is stored in the `OracleIntervalDS` structure

OracleIntervalDS Methods

The `OracleIntervalDS` methods are listed in Table 14-67.

Table 14-67 OracleIntervalDS Methods

Methods	Description
CompareTo	Compares the current `OracleIntervalDS` instance to an object, and returns an integer that represents their relative values
Equals	Determines whether or not the specified `object` has the same time interval as the current instance (Overloaded)
GetHashCode	Returns a hash code for the `OracleIntervalDS` instance
GetType	Inherited from `System.Object`
ToString	Converts the current `OracleIntervalDS` structure to a string

14.5.2 OracleIntervalDS Constructors

`OracleIntervalDS` constructors create a new instance of the `OracleIntervalDS` structure.

Overload List:

- OracleIntervalDS(TimeSpan)

  This constructor creates a new instance of the `OracleIntervalDS` structure and sets its value using a `TimeSpan` structure.
- OracleIntervalDS(string)

  This constructor creates a new instance of the `OracleIntervalDS` structure and sets its value using a string that indicates a period of time.
- OracleIntervalDS(double)

  This constructor creates a new instance of the `OracleIntervalDS` structure and sets its value using the total number of days.
- OracleIntervalDS(int, int, int, int, double)

  This constructor creates a new instance of the `OracleIntervalDS` structure and sets its value using the supplied days, hours, minutes, seconds and milliseconds.
- OracleIntervalDS(int, int, int, int, int)

  This constructor creates a new instance of the `OracleIntervalDS` structure and sets its value using the supplied days, hours, minutes, seconds, and nanoseconds.
- OracleIntervalDS(byte[ ])

  This constructor creates a new instance of the `OracleIntervalDS` structure and sets its value to the provided byte array, which is in an internal Oracle `INTERVAL DAY TO SECOND` format.

14.5.2.1 OracleIntervalDS(TimeSpan)

This constructor creates a new instance of the `OracleIntervalDS` structure and sets its value using a `TimeSpan` structure.

Declaration

```
// C#
public OracleIntervalDS(TimeSpan ts);
```

Parameters

- *`ts`*

  A `TimeSpan` structure.

14.5.2.2 OracleIntervalDS(string)

This constructor creates a new instance of the `OracleIntervalDS` structure and sets its value using a string that indicates a period of time.

Declaration

```
// C#
public OracleIntervalDS(string intervalStr);
```

Parameters

- *`intervalStr`*

  A string representing the Oracle `INTERVAL DAY TO SECOND`.

Exceptions

`ArgumentException` - The *`intervalStr`* parameter is not in the valid format or has an invalid value.

`ArgumentNullException` - The *`intervalStr`* parameter is null.

Remarks

The value specified in the supplied *`intervalStr`* must be in Day HH:MI:SSxFF format.

Example

"1 2:3:4.99" means 1 day, 2 hours, 3 minutes, 4 seconds, and 990 milliseconds or 1 day, 2 hours, 3 minutes, 4 seconds, and 990000000 nanoseconds.

14.5.2.3 OracleIntervalDS(double)

This constructor creates a new instance of the `OracleIntervalDS` structure and sets its value using the total number of days.

Declaration

```
// C#
public OracleIntervalDS(double totalDays);
```

Parameters

- *`totalDays`*

  The supplied total number of days for a time interval. Range of days is -1000,000,000 < *`totalDays`* < 1000,000,000.

Exceptions

`ArgumentOutOfRangeException` - The argument value for one or more of the parameters is out of the specified range.

`ArgumentException` - The argument values of the parameters cannot be used to construct a valid `OracleIntervalDS`.

14.5.2.4 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.

14.5.2.5 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.

14.5.2.6 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.

14.5.3 OracleIntervalDS Static Fields

The `OracleIntervalDS` static fields are listed in Table 14-68.

Table 14-68 OracleIntervalDS Static Fields

Field	Description
MaxValue	Represents the maximum valid time interval for an `OracleIntervalDS` structure
MinValue	Represents the minimum valid time interval for an `OracleIntervalDS` structure
Null	Represents a null value that can be assigned to an `OracleIntervalDS` instance
Zero	Represents a zero value for an `OracleIntervalDS` structure

14.5.3.1 MaxValue

This static field represents the maximum value for an `OracleIntervalDS` structure.

Declaration

```
// C#
public static readonly OracleIntervalDS MaxValue;
```

Remarks

Maximum values:

- Day: 999999999
- hour: 23
- minute is 59
- second: 59
- nanosecond: 999999999

14.5.3.2 MinValue

This static field represents the minimum value for an `OracleIntervalDS` structure.

Declaration

```
// C#
public static readonly OracleIntervalDS MinValue;
```

Remarks

Minimum values:

- Day: -999999999
- hour: -23
- minute: -59
- second: -59
- nanosecond: -999999999

14.5.3.3 Null

This static field represents a null value that can be assigned to an `OracleIntervalDS` instance.

Declaration

```
// C#
public static readonly OracleIntervalDS Null;
```

14.5.3.4 Zero

This static field represents a zero value for an `OracleIntervalDS` structure.

Declaration

```
// C#
public static readonly OracleIntervalDS Zero;
```

14.5.4 OracleIntervalDS Static Methods

The `OracleIntervalDS` static methods are listed in Table 14-69.

Table 14-69 OracleIntervalDS Static Methods

Methods	Description
Equals	Determines whether or not two `OracleIntervalDS` values are equal (Overloaded)
GreaterThan	Determines whether or not one `OracleIntervalDS` value is greater than another
GreaterThanOrEqual	Determines whether or not one `OracleIntervalDS` value is greater than or equal to another
LessThan	Determines whether or not one `OracleIntervalDS` value is less than another

Table 14-69 (Cont.) OracleIntervalDS Static Methods

Methods	Description
LessThanOrEqual	Determines whether or not one `OracleIntervalDS` value is less than or equal to another
NotEquals	Determines whether or not two `OracleIntervalDS` values are not equal
Parse	Returns an `OracleIntervalDS` structure and sets its value for time interval using a string
SetPrecision	Returns a new instance of an `OracleIntervalDS` with the specified day precision and fractional second precision

14.5.4.1 Equals

This static method determines whether or not two `OracleIntervalDS` values are equal.

Declaration

```
// 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 `OracleIntervalDS`s that contain a null value are equal.

14.5.4.2 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 `OracleIntervalDS`s that contain a null value are equal.

14.5.4.3 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 `OracleIntervalDS`s that contain a null value are equal.

14.5.4.4 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.

- Any `OracleIntervalDS` that has a value compares greater than an `OracleIntervalDS` that has a null value.
- Two `OracleIntervalDS`s that contain a null value are equal.

14.5.4.5 LessThanOrEqual

This static method determines whether or not the first of two `OracleIntervalDS` values is less than or equal to the second.

Declaration

```
// C#
public static bool LessThanOrEqual(OracleIntervalDS val1, OracleIntervalDS val2);
```

Parameters

- *`val1`*

  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 `OracleIntervalDS`s that contain a null value are equal.

14.5.4.6 NotEquals

This static method determines whether or not two `OracleIntervalDS` values are not equal.

Declaration

```
// C#
public static bool NotEquals(OracleIntervalDS val1, OracleIntervalDS val2);
```

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`s that contain a null value are equal.

14.5.4.7 Parse

This static method returns an `OracleIntervalDS` instance and sets its value for time interval using a string.

Declaration

```
// C#
public static OracleIntervalDS Parse(string intervalStr);
```

Parameters

- *`intervalStr`*

  A string representing the Oracle `INTERVAL DAY TO SECOND`.

Return Value

Returns an `OracleIntervalDS` instance representing the time interval from the supplied string.

Exceptions

`ArgumentException` - The *`intervalStr`* parameter is not in the valid format or *`intervalStr`* has an invalid value.

`ArgumentNullException` - The *`intervalStr`* parameter is null.

Remarks

The value specified in *`intervalStr`* must be in Day HH:MI:SSxFF format.

Example

"`1 2:3:4.99`" means 1 day, 2 hours, 3 minutes, 4 seconds, and 990 milliseconds or 1 day, 2 hours, 3 minutes, 4 seconds, and 990000000 nanoseconds.

14.5.4.8 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`.

14.5.5 OracleIntervalDS Static Operators

The `OracleIntervalDS` static operators are listed in Table 14-70.

Table 14-70 OracleIntervalDS Static Operators

Operator	Description
operator +	Adds two `OracleIntervalDS` values
operator ==	Determines whether or not two `OracleIntervalDS` values are equal
operator >	Determines whether or not one `OracleIntervalDS` value is greater than another
operator >=	Determines whether or not one `OracleIntervalDS` value is greater than or equal to another
operator !=	Determines whether or not two `OracleIntervalDS` values are not equal
operator <	Determines whether or not one `OracleIntervalDS` value is less than another
operator <=	Determines whether or not one `OracleIntervalDS` value is less than or equal to another
operator -	Subtracts one `OracleIntervalDS` value from another
operator -	Negates an `OracleIntervalDS` structure
operator *	Multiplies an `OracleIntervalDS` value by a number
operator /	Divides an `OracleIntervalDS` value by a number

14.5.5.1 operator +

This static operator adds two `OracleIntervalDS` values.

Declaration

```
// 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.

14.5.5.2 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 `OracleIntervalDS`s that contain a null value are equal.

14.5.5.3 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 `OracleIntervalDS`s that contain a null value are equal.

14.5.5.4 operator >=

This static operator determines if the first of two `OracleIntervalDS` values is greater than or equal to the second.

Declaration

```
// C#
public static bool operator >= (OracleIntervalDS val1,
  OracleIntervalDS val2);
```

Parameters

- *`val1`*

  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 `OracleIntervalDS`s that contain a null value are equal.

14.5.5.5 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 `OracleIntervalDS`s that contain a null value are equal.

14.5.5.6 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 `OracleIntervalDS`s that contain a null value are equal.

14.5.5.7 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 `OracleIntervalDS`s that contain a null value are equal.

14.5.5.8 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.

14.5.5.9 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.

14.5.5.10 operator *

This static operator multiplies an `OracleIntervalDS` value by a number.

Declaration

```
// 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.

14.5.5.11 operator /

This static operator divides an `OracleIntervalDS` value by a number.

Declaration

```
// 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.

14.5.6 OracleIntervalDS Type Conversions

The `OracleIntervalDS` type conversions are listed in Table 14-71.

Table 14-71 OracleIntervalDS Type Conversions

Operator	Description
explicit operator TimeSpan	Converts an `OracleIntervalDS` structure to a `TimeSpan` structure
explicit operator OracleIntervalDS	Converts a string to an `OracleIntervalDS` structure
implicit operator OracleIntervalDS	Converts a `TimeSpan` structure to an `OracleIntervalDS` structure

14.5.6.1 explicit operator TimeSpan

This type conversion operator converts an `OracleIntervalDS` structure to a `TimeSpan` structure.

Declaration

```
// C#
public static explicit operator TimeSpan(OracleIntervalDS val);
```

Parameters

- *`val`*

  An `OracleIntervalDS` instance.

Return Value

A `TimeSpan` structure.

Exceptions

`OracleNullValueException` - The `OracleIntervalDS` structure has a null value.

Remarks

14.5.6.2 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.

14.5.6.3 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`.

14.5.7 OracleIntervalDS Properties

The `OracleIntervalDS` properties are listed in Table 14-72.

Table 14-72 OracleIntervalDS Properties

Properties	Description
BinData	Returns an array of bytes that represents the Oracle `INTERVAL DAY TO SECOND` in Oracle internal format
Days	Gets the days component of an `OracleIntervalDS`
Hours	Gets the hours component of an `OracleIntervalDS`
IsNull	Indicates whether or not the current instance has a null value
Milliseconds	Gets the milliseconds component of an `OracleIntervalDS`
Minutes	Gets the minutes component of an `OracleIntervalDS`
Nanoseconds	Gets the nanoseconds component of an `OracleIntervalDS`
Seconds	Gets the seconds component of an `OracleIntervalDS`
TotalDays	Returns the total number, in days, that represent the time period in the `OracleIntervalDS` structure
Value	Specifies the time interval that is stored in the `OracleIntervalDS` structure

14.5.7.1 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

14.5.7.2 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.

14.5.7.3 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.

14.5.7.4 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`.

14.5.7.5 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.

14.5.7.6 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.

14.5.7.7 Nanoseconds

This property gets the nanoseconds component of an `OracleIntervalDS`.

Declaration

```
// C#
public int Nanoseconds {get;}
```

Property Value

An `int` that represents nanoseconds component.

Exceptions

`OracleNullValueException` - The current instance has a null value.

14.5.7.8 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.

14.5.7.9 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.

14.5.7.10 Value

This property specifies the time interval that is stored in the `OracleIntervalDS` structure.

Declaration

```
// C#
public TimeSpan Value {get;}
```

Property Value

A time interval.

Exceptions

`OracleNullValueException` - The current instance has a null value.

14.5.8 OracleIntervalDS Methods

The `OracleIntervalDS` methods are listed in Table 14-73.

Table 14-73 OracleIntervalDS Methods

Methods	Description
CompareTo	Compares the current `OracleIntervalDS` instance to an object, and returns an integer that represents their relative values
Equals	Determines whether or not the specified `object` has the same time interval as the current instance (Overloaded)
GetHashCode	Returns a hash code for the `OracleIntervalDS` instance
GetType	Inherited from `System.Object`
ToString	Converts the current `OracleIntervalDS` structure to a string

14.5.8.1 CompareTo

This method compares the current `OracleIntervalDS` instance to an object, and returns an integer that represents their relative values.

Declaration

```
// C#
public int CompareTo(object obj);
```

Parameters

- *`obj`*

  The object being compared to.

Return Value

The method returns:

- Less than zero: if the current `OracleIntervalDS` represents a shorter time interval than *`obj`*.
- Zero: if the current `OracleIntervalDS` and *`obj`* represent the same time interval.
- Greater than zero: if the current `OracleIntervalDS` represents a longer time interval than *`obj`*.

Implements

`IComparable`

Exceptions

`ArgumentException` - The *`obj`* parameter is not of type `OracleIntervalDS`.

Remarks

The following rules apply to the behavior of this method.

- The comparison must be between `OracleIntervalDS`s. For example, comparing an `OracleIntervalDS` instance with an `OracleBinary` instance is not allowed. When an `OracleIntervalDS` is compared with a different type, an `ArgumentException` is thrown.

- Any `OracleIntervalDS` that has a value compares greater than an `OracleIntervalDS` that has a null value.
- Two `OracleIntervalDS`s that contain a null value are equal.

14.5.8.2 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 `OracleIntervalDS`s that contain a null value are equal.

14.5.8.3 GetHashCode

Overrides `Object`

This method returns a hash code for the `OracleIntervalDS` instance.

Declaration

```
// C#
public override int GetHashCode();
```

14.5.8.4 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".

14.6 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
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver
Assembly	`Oracle.DataAccess.dll`	`Oracle.ManagedDataAccess.dll`
Namespace	`Oracle.DataAccess.Types`	`Oracle.ManagedDataAccess.Types`
.NET Framework	3.5, 4.5, 4.6	4.5, 4.6

Thread Safety

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

Example

```
// C#

using System;
using Oracle.DataAccess.Types;

class OracleIntervalYMSample
{
  static void Main()
  {
    OracleIntervalYM iYMMax = OracleIntervalYM.MaxValue;
    double totalYears = iYMMax.TotalYears;

    totalYears -= 1;
    OracleIntervalYM iYMMax_1 = new OracleIntervalYM(totalYears);
    // Calculate the difference
    OracleIntervalYM iYMDiff = iYMMax - iYMMax_1;

    // Prints "iYMDiff.ToString() = +000000001-00"
    Console.WriteLine("iYMDiff.ToString() = " + iYMDiff.ToString());
  }
}
```

14.6.1 OracleIntervalYM Members

`OracleIntervalYM` members are listed in the following tables:

OracleIntervalYM Constructors

`OracleIntervalYM` constructors are listed in Table 14-74

Table 14-74 OracleIntervalYM Constructors

Constructor	Description
OracleIntervalYM Constructors	Instantiates a new instance of `OracleIntervalYM` structure (Overloaded)

OracleIntervalYM Static Fields

The `OracleIntervalYM` static fields are listed in Table 14-75.

Table 14-75 OracleIntervalYM Static Fields

Field	Description
MaxValue	Represents the maximum value for an `OracleIntervalYM` structure
MinValue	Represents the minimum value for an `OracleIntervalYM` structure
Null	Represents a null value that can be assigned to an `OracleIntervalYM` instance
Zero	Represents a zero value for an `OracleIntervalYM` structure

OracleIntervalYM Static Methods

The `OracleIntervalYM` static methods are listed in Table 14-76.

Table 14-76 OracleIntervalYM Static Methods

Methods	Description
Equals	Determines whether or not two `OracleIntervalYM` values are equal (Overloaded)
GreaterThan	Determines whether or not one `OracleIntervalYM` value is greater than another

Table 14-76 (Cont.) OracleIntervalYM Static Methods

Methods	Description
GreaterThanOrEqual	Determines whether or not one `OracleIntervalYM` value is greater than or equal to another
LessThan	Determines whether or not one `OracleIntervalYM` value is less than another
LessThanOrEqual	Determines whether or not one `OracleIntervalYM` value is less than or equal to another
NotEquals	Determines whether two `OracleIntervalYM` values are not equal
Parse	Returns an `OracleIntervalYM` structure and sets its value for time interval using a string
SetPrecision	Returns a new instance of an `OracleIntervalYM` with the specified year precision.

OracleIntervalYM Static Operators

The `OracleIntervalYM` static operators are listed in Table 14-77.

Table 14-77 OracleIntervalYM Static Operators

Operator	Description
operator +	Adds two `OracleIntervalYM` values
operator ==	Determines whether or not two `OracleIntervalYM` values are equal
operator >	Determines whether or not one `OracleIntervalYM` value is greater than another
operator >=	Determines whether or not one `OracleIntervalYM` value is greater than or equal to another
operator !=	Determines whether two `OracleIntervalYM` values are not equal
operator <	Determines whether or not one `OracleIntervalYM` value is less than another
operator <=	Determines whether or not one `OracleIntervalYM` value is less than or equal to another
operator -	Subtracts one `OracleIntervalYM` value from another
operator -	Negates an `OracleIntervalYM` structure
operator *	Multiplies an `OracleIntervalYM` value by a number
operator /	Divides an `OracleIntervalYM` value by a number

OracleIntervalYM Type Conversions

The `OracleIntervalYM` conversions are listed in Table 14-78.

Table 14-78 OracleIntervalYM Type Conversions

Operator	Description
explicit operator long	Converts an `OracleIntervalYM` structure to a number
explicit operator OracleIntervalYM	Converts a string to an `OracleIntervalYM` structure
implicit operator OracleIntervalYM	Converts the number of months to an `OracleIntervalYM` structure

OracleIntervalYM Properties

The `OracleIntervalYM` properties are listed in Table 14-79.

Table 14-79 OracleIntervalYM Properties

Properties	Description
BinData	Returns an array of bytes that represents the Oracle `INTERVAL YEAR TO MONTH` in an Oracle internal format
IsNull	Indicates whether or not the current instance has a null value
Months	Gets the months component of an `OracleIntervalYM`
TotalYears	Returns the total number, in years, that represents the period of time in the current `OracleIntervalYM` structure
Value	Specifies the total number of months that is stored in the `OracleIntervalYM` structure
Years	Gets the years component of an `OracleIntervalYM`

OracleIntervalYM Methods

The `OracleIntervalYM` methods are listed in Table 14-80.

Table 14-80 OracleIntervalYM Methods

Methods	Description
CompareTo	Compares the current `OracleIntervalYM` instance to the supplied object, and returns an integer that represents their relative values
Equals	Determines whether or not the specified `object` has the same time interval as the current instance (Overloaded)
GetHashCode	Returns a hash code for the `OracleIntervalYM` instance
GetType	Inherited from `System.Object`

Table 14-80 (Cont.) OracleIntervalYM Methods

Methods	Description
ToString	Converts the current `OracleIntervalYM` structure to a string

14.6.2 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.

14.6.2.1 OracleIntervalYM(long)

This method creates a new instance of the `OracleIntervalYM` structure using the supplied total number of months for a period of time.

Declaration

```
// C#
public OracleIntervalYM (long totalMonths);
```

Parameters

- *`totalMonths`*

  The number of total months for a time interval. Range is -12,000,000,000 < *`totalMonths`* < 12,000,000,000.

Exceptions

`ArgumentOutOfRangeException` - The *`totalMonths`* parameter is out of the specified range.

14.6.2.2 OracleIntervalYM(string)

This method creates a new instance of the `OracleIntervalYM` structure and sets its value using the supplied string.

Declaration

```
// C#
public OracleIntervalYM (string intervalStr);
```

Parameters

- *`intervalStr`*

  A string representing the Oracle `INTERVAL YEAR TO MONTH`.

Remarks

The value specified in the supplied *`intervalStr`* must be in Year-Month format.

Exceptions

`ArgumentException` - The *`intervalStr`* parameter is not in the valid format or *`intervalStr`* has an invalid value.

`ArgumentNullException` - The *`intervalStr`* parameter is null.

Example

"1-2" means 1 year and 2 months.

14.6.2.3 OracleIntervalYM(double)

This method creates a new instance of the `OracleIntervalYM` structure and sets its value using the total number of years.

Declaration

```
// C#
public OracleIntervalYM (double totalYears);
```

Parameters

- *`totalYears`*

  Number of total years. Range is -1,000,000,000 < *`totalYears`* > 1,000,000,000.

Exceptions

`ArgumentOutOfRangeException` - The *`totalYears`* parameter is out of the specified range.

`ArgumentException` - The *`totalYears`* parameter cannot be used to construct a valid `OracleIntervalYM`.

14.6.2.4 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`.

14.6.2.5 OracleIntervalYM(byte[ ])

This method creates a new instance of the `OracleIntervalYM` structure and sets its value to the provided byte array, which is in an internal Oracle `INTERVAL DAY TO SECOND` format.

Declaration

```
// C#
public OracleIntervalYM (byte[] bytes);
```

Parameters

- *`bytes`*

  A byte array that is in an internal Oracle `INTERVAL YEAR TO MONTH` format.

Exceptions

`ArgumentException` - The supplied byte array is not in an internal Oracle `INTERVAL YEAR TO MONTH` format or the supplied byte array has an invalid value.

`ArgumentNullException` - *`bytes`* is null.

Remarks

The supplied byte array must be in an internal Oracle `INTERVAL YEAR TO MONTH` format.

14.6.3 OracleIntervalYM Static Fields

The `OracleIntervalYM` static fields are listed in Table 14-81.

Table 14-81 OracleIntervalYM Static Fields

Field	Description
MaxValue	Represents the maximum value for an `OracleIntervalYM` structure
MinValue	Represents the minimum value for an `OracleIntervalYM` structure
Null	Represents a null value that can be assigned to an `OracleIntervalYM` instance
Zero	Represents a zero value for an `OracleIntervalYM` structure

14.6.3.1 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.

14.6.3.2 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.

14.6.3.3 Null

This static field represents a null value that can be assigned to an `OracleIntervalYM` instance.

Declaration

```
// C#
public static readonly OracleIntervalYM Null;
```

14.6.3.4 Zero

This static field represents a zero value for an `OracleIntervalYM` structure.

Declaration

```
// C#
public static readonly OracleIntervalDS Zero;
```

14.6.4 OracleIntervalYM Static Methods

The `OracleIntervalYM` static methods are listed in Table 14-82.

Table 14-82 OracleIntervalYM Static Methods

Methods	Description
Equals	Determines whether or not two `OracleIntervalYM` values are equal (Overloaded)
GreaterThan	Determines whether or not one `OracleIntervalYM` value is greater than another
GreaterThanOrEqual	Determines whether or not one `OracleIntervalYM` value is greater than or equal to another
LessThan	Determines whether or not one `OracleIntervalYM` value is less than another
LessThanOrEqual	Determines whether or not one `OracleIntervalYM` value is less than or equal to another
NotEquals	Determines whether two `OracleIntervalYM` values are not equal
Parse	Returns an `OracleIntervalYM` structure and sets its value for time interval using a string
SetPrecision	Returns a new instance of an `OracleIntervalYM` with the specified year precision.

14.6.4.1 Equals

This static method determines whether or not two `OracleIntervalYM` values are equal.

Declaration

```
// C#
public static bool Equals(OracleIntervalYM val1, OracleIntervalYM val2);
```

Parameters

- *`val1`*

  An `OracleIntervalYM` structure.

- *`val2`*

  An `OracleIntervalYM` structure.

Return Value

Returns `true` if two `OracleIntervalYM` values represent the same time interval, otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleIntervalYM` that has a value compares greater than an `OracleIntervalYM` that has a null value.
- Two `OracleIntervalYM`s that contain a null value are equal.

14.6.4.2 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 `OracleIntervalYM`s that contain a null value are equal.

14.6.4.3 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.

14.6.4.4 LessThan

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`s that contain a null value are equal.

14.6.4.5 LessThanOrEqual

This static method determines whether or not the first of two `OracleIntervalYM` values is less than or equal to the second.

Declaration

```
// C#
public static bool LessThanOrEqual(OracleIntervalYM val1, OracleIntervalYM val2);
```

Parameters

- *`val1`*

  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`s that contain a null value are equal.

14.6.4.6 NotEquals

This static method determines whether two `OracleIntervalYM` values are not equal.

Declaration

```
// C#
public static bool NotEquals(OracleIntervalYM val1, OracleIntervalYM val2);
```

Parameters

- *`val1`*

  The first `OracleIntervalYM`.

- *`val2`*

  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.

14.6.4.7 Parse

This static method returns an `OracleIntervalYM` structure and sets its value for time interval using a string.

Declaration

```
// C#
public static OracleIntervalYM Parse (string intervalStr);
```

Parameters

- *`intervalStr`*

  A string representing the Oracle `INTERVAL YEAR TO MONTH`.

Return Value

Returns an `OracleIntervalYM` structure.

Exceptions

`ArgumentException` - The *`intervalStr`* parameter is not in the valid format or *`intervalStr`* has an invalid value.

`ArgumentNullException` - The *`intervalStr`* parameter is null.

Remarks

The value specified in the supplied *`intervalStr`* must be in the Year-Month format.

Example

"1-2" means 1 year and 2 months.

14.6.4.8 SetPrecision

This static method returns a new instance of an `OracleIntervalYM` with the specified year precision.

Declaration

```
// C#
public static OracleIntervalYM SetPrecision(OracleIntervalYM value1,
   int yearPrecision);
```

Parameters

- *`value1`*

  An `OracleIntervalYM` structure.

- *`yearPrecision`*

The year precision provided. Range of year precision is (0 to 9).

Return Value

An `OracleIntervalDS` instance.

Exceptions

`ArgumentOutOfRangeException` - *`yearPrecision`* is out of the specified range.

Remarks

Depending on the value specified in the supplied `yearPrecision`, `0` or more leading zeros are displayed in the string returned by `ToString()`.

Example

An `OracleIntervalYM` with a value of "`1-2`" results in the string "`001-2`" when `SetPrecision()` is called with the year precision set to `3`.

14.6.5 OracleIntervalYM Static Operators

The `OracleIntervalYM` static operators are listed in Table 14-83.

Table 14-83 OracleIntervalYM Static Operators

Operator	Description
operator +	Adds two `OracleIntervalYM` values
operator ==	Determines whether or not two `OracleIntervalYM` values are equal
operator >	Determines whether or not one `OracleIntervalYM` value is greater than another
operator >=	Determines whether or not one `OracleIntervalYM` value is greater than or equal to another
operator !=	Determines whether two `OracleIntervalYM` values are not equal
operator <	Determines whether or not one `OracleIntervalYM` value is less than another
operator <=	Determines whether or not one `OracleIntervalYM` value is less than or equal to another
operator -	Subtracts one `OracleIntervalYM` value from another
operator -	Negates an `OracleIntervalYM` structure
operator *	Multiplies an `OracleIntervalYM` value by a number
operator /	Divides an `OracleIntervalYM` value by a number

14.6.5.1 operator +

This static operator adds two `OracleIntervalYM` values.

Declaration

```
// 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.

14.6.5.2 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 `OracleIntervalYM`s that contain a null value are equal.

14.6.5.3 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`.

- *`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 `OracleIntervalYM`s that contain a null value are equal.

14.6.5.4 operator >=

This static operator determines if the first of two `OracleIntervalYM` values is greater than or equal to the second.

Declaration

```
// C#
public static bool operator >= (OracleIntervalYM val1, OracleIntervalYM val2);
```

Parameters

- *`val1`*

  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 `OracleIntervalYM`s that contain a null value are equal.

14.6.5.5 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 `OracleIntervalYM`s that contain a null value are equal.

14.6.5.6 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 `OracleIntervalYM`s that contain a null value are equal.

14.6.5.7 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 `OracleIntervalYM`s that contain a null value are equal.

14.6.5.8 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.

14.6.5.9 operator -

This static operator negates an `OracleIntervalYM` structure.

Declaration

```
// C#
public static OracleIntervalYM operator - (OracleIntervalYM val);
```

Parameters

- *val*

  An `OracleIntervalYM`.

Return Value

An `OracleIntervalYM` structure.

Remarks

If the supplied `OracleIntervalYM` structure has a null value, the returned `OracleIntervalYM` structure has a null value.

14.6.5.10 operator *

This static operator multiplies an `OracleIntervalYM` value by a number.

Declaration

```
// 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.

14.6.5.11 operator /

This static operator divides an `OracleIntervalYM` value by a number.

Declaration

```
// 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.

14.6.6 OracleIntervalYM Type Conversions

The `OracleIntervalYM` conversions are listed in Table 14-84.

Table 14-84 OracleIntervalYM Type Conversions

Operator	Description
explicit operator long	Converts an `OracleIntervalYM` structure to a number
explicit operator OracleIntervalYM	Converts a string to an `OracleIntervalYM` structure
implicit operator OracleIntervalYM	Converts the number of months to an `OracleIntervalYM` structure

14.6.6.1 explicit operator long

This type conversion operator converts an `OracleIntervalYM` to a number that represents the number of months in the time interval.

Declaration

```
// C#
public static explicit operator long (OracleIntervalYM val);
```

Parameters

- *`val`*

  An `OracleIntervalYM` structure.

Return Value

A `long` number in months.

Exceptions

`OracleNullValueException` - The `OracleIntervalYM` structure has a null value.

14.6.6.2 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.

14.6.6.3 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.

14.6.7 OracleIntervalYM Properties

The `OracleIntervalYM` properties are listed in Table 14-85.

Table 14-85 OracleIntervalYM Properties

Properties	Description
BinData	Returns an array of bytes that represents the Oracle `INTERVAL YEAR TO MONTH` in an Oracle internal format
IsNull	Indicates whether or not the current instance has a null value
Months	Gets the months component of an `OracleIntervalYM`
TotalYears	Returns the total number, in years, that represents the period of time in the current `OracleIntervalYM` structure
Value	Specifies the total number of months that is stored in the `OracleIntervalYM` structure
Years	Gets the years component of an `OracleIntervalYM`

14.6.7.1 BinData

This property returns an array of bytes that represents the Oracle `INTERVAL YEAR TO MONTH` in Oracle internal format.

Declaration

```
// C#
public byte[] BinData {get;}
```

Property Value

A byte array that represents an Oracle `INTERVAL YEAR TO MONTH` in Oracle internal format.

Exceptions

`OracleNullValueException` - The current instance has a null value.

14.6.7.2 IsNull

This property indicates whether or not the value has a null value.

Declaration

```
// C#
public bool IsNull {get;}
```

Property Value

Returns `true` if value has a null value; otherwise, returns `false`.

14.6.7.3 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.

14.6.7.4 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.

14.6.7.5 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.

14.6.7.6 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.

14.6.8 OracleIntervalYM Methods

The `OracleIntervalYM` methods are listed in Table 14-86.

Table 14-86 OracleIntervalYM Methods

Methods	Description
CompareTo	Compares the current `OracleIntervalYM` instance to the supplied object, and returns an integer that represents their relative values
Equals	Determines whether or not the specified `object` has the same time interval as the current instance (Overloaded)
GetHashCode	Returns a hash code for the `OracleIntervalYM` instance
GetType	Inherited from `System.Object`
ToString	Converts the current `OracleIntervalYM` structure to a string

14.6.8.1 CompareTo

This method compares the current `OracleIntervalYM` instance to the supplied object, and returns an integer that represents their relative values.

Declaration

```
// C#
public int CompareTo(object obj);
```

Parameters

- *`obj`*

  The supplied object.

Return Value

The method returns a number:

Less than zero: if the current `OracleIntervalYM` represents a shorter time interval than *`obj`*.

Zero: if the current `OracleIntervalYM` and *`obj`* represent the same time interval.

Greater than zero: if the current `OracleIntervalYM` represents a longer time interval than *`obj`*.

Implements

`IComparable`

Exceptions

`ArgumentException` - The *`obj`* parameter is not of type `OracleIntervalYM`.

Remarks

The following rules apply to the behavior of this method.

- The comparison must be between `OracleIntervalYM`s. For example, comparing an `OracleIntervalYM` instance with an `OracleBinary` instance is not allowed. When an `OracleIntervalYM` is compared with a different type, an `ArgumentException` is thrown.
- Any `OracleIntervalYM` that has a value compares greater than an `OracleIntervalYM` that has a null value.
- Two `OracleIntervalYM`s that contain a null value are equal.

14.6.8.2 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 `OracleIntervalYM`s that contain a null value are equal.

14.6.8.3 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.

14.6.8.4 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".

14.7 OracleString Structure

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

```
// C#
public struct OracleString : IComparable, INullable, IXmlSerializable
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver
Assembly	`Oracle.DataAccess.dll`	`Oracle.ManagedDataAccess.dll`
Namespace	`Oracle.DataAccess.Types`	`Oracle.ManagedDataAccess.Types`
.NET Framework	3.5, 4.5, 4.6	4.5, 4.6

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);
  }
}
```

See Also:

- "Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"
- OracleString Members
- OracleString Constructors
- OracleString Static Fields
- OracleString Static Methods
- OracleString Static Operators
- OracleString Type Conversions
- OracleString Properties
- OracleString Methods

14.7.1 OracleString Members

`OracleString` members are listed in the following tables:

OracleString Constructors

`OracleString` constructors are listed in Table 14-87

Table 14-87 OracleString Constructors

Constructor	Description
OracleString Constructors	Instantiates a new instance of `OracleString` structure (Overloaded)

OracleString Static Fields

The `OracleString` static fields are listed in Table 14-88.

Table 14-88 OracleString Static Fields

Field	Description
Null	Represents a null value that can be assigned to an instance of the `OracleString` structure

OracleString Static Methods

The `OracleString` static methods are listed in Table 14-89.

Table 14-89 OracleString Static Methods

Methods	Description
Concat	Concatenates two `OracleString` instances and returns a new `OracleString` instance that represents the result
Equals	Determines if two `OracleString` values are equal (Overloaded)
GreaterThan	Determines whether or not the first of two `OracleString` values is greater than the second
GreaterThanOrEqual	Determines whether or not the first of two `OracleString` values is greater than or equal to the second
LessThan	Determines whether or not the first of two `OracleString` values is less than the second
LessThanOrEqual	Determines whether or not the first of two `OracleString` values is less than or equal to the second
NotEquals	Determines whether two `OracleString` values are not equal

OracleString Static Operators

The `OracleString` static operators are listed in Table 14-90.

Table 14-90 OracleString Static Operators

Operator	Description
operator +	Concatenates two `OracleString` values
operator ==	Determines if two `OracleString` values are equal
operator >	Determines if the first of two `OracleString` values is greater than the second
operator >=	Determines if the first of two `OracleString` values is greater than or equal to the second
operator !=	Determines if the two `OracleString` values are not equal
operator <	Determines if the first of two `OracleString` values is less than the second
operator <=	Determines if two `OracleString` values are not equal

OracleString Type Conversions

The `OracleString` type conversions are listed in Table 14-91.

Table 14-91   OracleString Type Conversions

Operator	Description
explicit operator string	Converts the supplied `OracleString` to a `string` instance
implicit operator OracleString	Converts the supplied `string` to an `OracleString` instance

OracleString Properties

The `OracleString` properties are listed in Table 14-92.

Table 14-92   OracleString Properties

Properties	Description
IsCaseIgnored	Indicates whether or not case should be ignored when performing string comparison
IsNull	Indicates whether or not the current instance has a null value
Item	Obtains the particular character in an `OracleString` using an index.
Length	Returns the length of the `OracleString`
Value	Returns the string data that is stored in the `OracleString` structure.

OracleString Methods

The `OracleString` methods are listed in Table 14-93.

Table 14-93   OracleString Methods

Methods	Description
Clone	Returns a copy of the current `OracleString` instance
CompareTo	Compares the current `OracleString` instance to the supplied object, and returns an integer that represents their relative values
Equals	Determines whether or not an object has the same string value as the current `OracleString` structure (Overloaded)
GetHashCode	Returns a hash code for the `OracleString` instance
GetNonUnicodeBytes	Returns an array of bytes, containing the contents of the `OracleString`, in the client character set format
GetType	Inherited from `System.Object`
GetUnicodeBytes	Returns an array of bytes, containing the contents of the `OracleString`, in Unicode format

Table 14-93 (Cont.) OracleString Methods

Methods	Description
ToString	Converts the current `OracleString` instance to a `string`

14.7.2 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.

14.7.2.1 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.

14.7.2.2 OracleString(string, bool)

This constructor creates a new instance of the `OracleString` structure and sets its value using a string and specifies if case is ignored in comparison.

Declaration

```
// C#
public OracleString(string data, bool isCaseIgnored);
```

Parameters

- *`data`*

  A string value.

- *`isCaseIgnored`*

  Specifies if case is ignored in comparison. Specifies `true` if case is to be ignored; otherwise, specifies `false`.

14.7.2.3 OracleString(byte [ ], bool)

This constructor creates a new instance of the `OracleString` structure and sets its value using a byte array and specifies if the supplied byte array is Unicode encoded.

Declaration

```
// C#
public OracleString(byte[] data, bool fUnicode);
```

Parameters

- *`data`*

  Byte array data for the new `OracleString`.

- *`fUnicode`*

  Specifies if the supplied `data` is Unicode encoded. Specifies `true` if Unicode encoded; otherwise, `false`.

Exceptions

`ArgumentNullException` - The *`data`* parameter is null.

14.7.2.4 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.

14.7.2.5 OracleString(byte [ ], int, int, bool)

This constructor creates a new instance of the `OracleString` structure and sets its value using a byte array, and specifies the following: the starting index in the byte array, the number of bytes to copy from the byte array, and if the supplied byte array is Unicode encoded.

Declaration

```
// C#
public OracleString(byte[] data, int index, int count, bool fUnicode);
```

Parameters

- *`data`*

  Byte array data for the new `OracleString`.
- *`index`*

  The starting index to copy from *`data`*.
- *`count`*

  The number of bytes to copy.
- *`fUnicode`*

  Specifies if the supplied `data` is Unicode encoded. Specifies `true` if Unicode encoded; otherwise, `false`.

Exceptions

`ArgumentNullException` - The *`data`* parameter is null.

`ArgumentOutOfRangeException` - The *`count`* parameter is less than zero.

`IndexOutOfRangeException` - The *`index`* parameter is greater than or equal to the length of *`data`* or less than zero.

14.7.2.6 OracleString(byte [ ], int, int, bool, bool)

This constructor creates a new instance of the `OracleString` structure and sets its value using a byte array, and specifies the following: the starting index in the byte array, the number of bytes to copy from the byte array, if the supplied byte array is Unicode encoded, and if case is ignored in comparison.

Declaration

```
// C#
public OracleString(byte[] data, int index, int count, bool fUnicode,
 bool isCaseIgnored);
```

Parameters

- *`data`*

  Byte array data for the new `OracleString`.
- *`index`*

  The starting index to copy from *`data`*.
- *`count`*

  The number of bytes to copy.
- *`fUnicode`*

  Specifies if the supplied `data` is Unicode encoded. Specifies `true` if Unicode encoded; otherwise, `false`.
- *`isCaseIgnored`*

  Specifies if case is ignored in comparison. Specifies `true` if case is to be ignored; otherwise, specifies `false`.

Exceptions

`ArgumentNullException` - The *`data`* parameter is null.

`ArgumentOutOfRangeException` - The *`count`* parameter is less than zero.

`IndexOutOfRangeException` - The *`index`* parameter is greater than or equal to the length of *`data`* or less than zero.

14.7.3 OracleString Static Fields

The `OracleString` static fields are listed in Table 14-94.

Table 14-94 OracleString Static Fields

Field	Description
Null	Represents a null value that can be assigned to an instance of the `OracleString` structure

14.7.3.1 Null

This static field represents a null value that can be assigned to an instance of the `OracleString` structure.

Declaration

```
// C#
public static readonly OracleString Null;
```

14.7.4 OracleString Static Methods

The `OracleString` static methods are listed in Table 14-95.

Table 14-95 OracleString Static Methods

Methods	Description
Concat	Concatenates two `OracleString` instances and returns a new `OracleString` instance that represents the result
Equals	Determines if two `OracleString` values are equal (Overloaded)
GreaterThan	Determines whether or not the first of two `OracleString` values is greater than the second
GreaterThanOrEqual	Determines whether or not the first of two `OracleString` values is greater than or equal to the second
LessThan	Determines whether or not the first of two `OracleString` values is less than the second
LessThanOrEqual	Determines whether or not the first of two `OracleString` values is less than or equal to the second
NotEquals	Determines whether two `OracleString` values are not equal

14.7.4.1 Concat

This static method concatenates two `OracleString` instances and returns a new `OracleString` instance that represents the result.

Declaration

```
// C#
public static OracleString Concat(OracleString str1, OracleString str2);
```

Parameters

- *`str1`*

  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.

14.7.4.2 Equals

Overloads `Object`

This static method determines whether or not the two `OracleString`s being compared are equal.

Declaration

```
// 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 `OracleString`s being compared are equal; returns `false` otherwise.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleString` that has a value is greater than an `OracleString` that has a null value.
- Two `OracleString`s that contain a null value are equal.

14.7.4.3 GreaterThan

This static method determines whether or not the first of two `OracleString` values is greater than the second.

Declaration

```
// 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 `OracleString`s is greater than the second; otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleString` that has a value is greater than an `OracleString` that has a null value.
- Two `OracleString`s that contain a null value are equal.

14.7.4.4 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 `OracleString`s 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 `OracleString`s that contain a null value are equal.

14.7.4.5 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 `OracleString`s that contain a null value are equal.

14.7.4.6 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 `OracleString`s that contain a null value are equal.

14.7.4.7 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 `OracleString`s that contain a null value are equal.

14.7.5 OracleString Static Operators

The `OracleString` static operators are listed in Table 14-96.

Table 14-96 OracleString Static Operators

Operator	Description
operator +	Concatenates two `OracleString` values
operator ==	Determines if two `OracleString` values are equal
operator >	Determines if the first of two `OracleString` values is greater than the second
operator >=	Determines if the first of two `OracleString` values is greater than or equal to the second
operator !=	Determines if the two `OracleString` values are not equal
operator <	Determines if the first of two `OracleString` values is less than the second
operator <=	Determines if two `OracleString` values are not equal

14.7.5.1 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.

14.7.5.2 operator ==

This static operator determines if two `OracleString` values are equal.

Declaration

```
// 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 `OracleString`s that contain a null value are equal.

14.7.5.3 operator >

This static operator determines if the first of two `OracleString` values is greater than the second.

Declaration

```
// C#
public static bool operator > (OracleString value1, OracleString value2);
```

Parameters

- *`value1`*

  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 `OracleString`s that contain a null value are equal.

14.7.5.4 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 `OracleString`s that contain a null value are equal.

14.7.5.5 operator !=

This static operator determines if two `OracleString` values are not equal.

Declaration

```
// 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.

14.7.5.6 operator <

This static operator determines if the first of two `OracleString`s is less than 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`s is less than the second; otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleString` that has a value is greater than an `OracleString` has a null value.
- Two `OracleString`s that contain a null value are equal.

14.7.5.7 operator <=

This static operator determines if the first of two `OracleString` values is less than or equal to the second.

Declaration

```
// C#
public static bool operator <= (OracleString value1, OracleString value1);
```

Parameters

- *`value1`*

  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 `OracleString`s that contain a null value are equal.

14.7.6 OracleString Type Conversions

The `OracleString` type conversions are listed in Table 14-97.

Table 14-97 OracleString Type Conversions

Operator	Description
explicit operator string	Converts the supplied `OracleString` to a `string` instance

Table 14-97 (Cont.) OracleString Type Conversions

Operator	Description
implicit operator OracleString	Converts the supplied `string` to an `OracleString` instance

14.7.6.1 explicit operator string

This type conversion operator converts the supplied `OracleString` to a `string`.

Declaration

```
//C#
public static explicit operator string (OracleString value1);
```

Parameters

- *`value1`*

  The supplied `OracleString`.

Return Value

`string`

Exceptions

`OracleNullValueException` - The `OracleString` structure has a null value.

14.7.6.2 implicit operator OracleString

This type conversion operator converts the supplied `string` to an `OracleString`.

Declaration

```
// C#
public static implicit operator OracleString (string value1);
```

Parameters

- *`value1`*

  The supplied string.

Return Value

An `OracleString`.

14.7.7 OracleString Properties

The `OracleString` properties are listed in Table 14-98.

Table 14-98   OracleString Properties

Properties	Description
IsCaseIgnored	Indicates whether or not case should be ignored when performing string comparison
IsNull	Indicates whether or not the current instance has a null value
Item	Obtains the particular character in an `OracleString` using an index.
Length	Returns the length of the `OracleString`
Value	Returns the string data that is stored in the `OracleString` structure.

14.7.7.1 IsCaseIgnored

This property indicates whether or not case should be ignored when performing string comparison.

Declaration

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

Property Value

Returns `true` if string comparison must ignore case; otherwise `false`.

Remarks

Default value is `true`.

Example

```
// C#

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));
  }
}
```

14.7.7.2 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`.

14.7.7.3 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.

14.7.7.4 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.

14.7.7.5 Value

This property returns the string data that is stored in the `OracleString`.

Declaration

```
// C#
public string Value {get;}
```

Property Value

The stored string value

Exceptions

`OracleNullValueException` - The current instance has a null value.

14.7.8 OracleString Methods

The `OracleString` methods are listed in Table 14-99.

Table 14-99 OracleString Methods

Methods	Description
Clone	Returns a copy of the current `OracleString` instance
CompareTo	Compares the current `OracleString` instance to the supplied object, and returns an integer that represents their relative values
Equals	Determines whether or not an object has the same string value as the current `OracleString` structure (Overloaded)
GetHashCode	Returns a hash code for the `OracleString` instance
GetNonUnicodeBytes	Returns an array of bytes, containing the contents of the `OracleString`, in the client character set format
GetType	Inherited from `System.Object`
GetUnicodeBytes	Returns an array of bytes, containing the contents of the `OracleString`, in Unicode format
ToString	Converts the current `OracleString` instance to a `string`

14.7.8.1 Clone

This method creates a copy of an `OracleString` instance.

Declaration

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

Return Value

An `OracleString` structure.

Remarks

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

Example

```
// C#

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));
  }
}
```

14.7.8.2 CompareTo

This method compares the current `OracleString` instance to the supplied object, and returns an integer that represents their relative values.

Declaration

```
// C#
public int CompareTo(object obj);
```

Parameters

- *`obj`*

  The object being compared to the current instance.

Return Value

The method returns a number that is:

- Less than zero: if the current `OracleString` value is less than *`obj`*.
- Zero: if the current `OracleString` value is equal to *`obj`*.
- Greater than zero: if the current `OracleString` value is greater than *`obj`*.

Implements

`IComparable`

Exceptions

`ArgumentException` - The *`obj`* parameter is not of type `OracleString`.

Remarks

The following rules apply to the behavior of this method.

- The comparison must be between `OracleString`s. For example, comparing an `OracleString` instance with an `OracleBinary` instance is not allowed. When an `OracleString` is compared with a different type, an `ArgumentException` is thrown.
- Any `OracleString` that has a value is greater than an `OracleString` that has a null value.
- Two `OracleString`s that contain a null value are equal.

14.7.8.3 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 `OracleString`s that contain a null value are equal.

14.7.8.4 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.

14.7.8.5 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.

14.7.8.6 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.

14.7.8.7 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".

14.8 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

```
 // C#public struct OracleTimeStamp : IComparable, INullable, IXmlSerializable
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver
Assembly	`Oracle.DataAccess.dll`	`Oracle.ManagedDataAccess.dll`
Namespace	`Oracle.DataAccess.Types`	`Oracle.ManagedDataAccess.Types`
.NET Framework	3.5, 4.5, 4.6	4.5, 4.6

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);
  }
}
```

14.8.1 OracleTimeStamp Members

`OracleTimeStamp` members are listed in the following tables:

OracleTimeStamp Constructors

`OracleTimeStamp` constructors are listed in Table 14-100

Table 14-100 OracleTimeStamp Constructors

Constructor	Description
OracleTimeStamp Constructors	Instantiates a new instance of `OracleTimeStamp` structure (Overloaded)

OracleTimeStamp Static Fields

The `OracleTimeStamp` static fields are listed in Table 14-101.

Table 14-101 OracleTimeStamp Static Fields

Field	Description
MaxValue	Represents the maximum valid date for an `OracleTimeStamp` structure, which is December 31, 9999 23:59:59.999999999
MinValue	Represents the minimum valid date for an `OracleTimeStamp` structure, which is January 1, -4712 0:0:0
Null	Represents a null value that can be assigned to an instance of the `OracleTimeStamp` structure

OracleTimeStamp Static Methods

The `OracleTimeStamp` static methods are listed in Table 14-102.

Table 14-102 OracleTimeStamp Static Methods

Methods	Description
Equals	Determines if two `OracleTimeStamp` values are equal (Overloaded)
GreaterThan	Determines if the first of two `OracleTimeStamp` values is greater than the second
GreaterThanOrEqual	Determines if the first of two `OracleTimeStamp` values is greater than or equal to the second
LessThan	Determines if the first of two `OracleTimeStamp` values is less than the second
LessThanOrEqual	Determines if the first of two `OracleTimeStamp` values is less than or equal to the second

Table 14-102 (Cont.) OracleTimeStamp Static Methods

Methods	Description
NotEquals	Determines if two `OracleTimeStamp` values are not equal
GetSysDate	Gets an `OracleTimeStamp` structure that represents the current date and time
Parse	Gets an `OracleTimeStamp` structure and sets its value using the supplied string
SetPrecision	Returns a new instance of an `OracleTimeStamp` with the specified fractional second precision

OracleTimeStamp Static Operators

The `OracleTimeStamp` static operators are listed in Table 14-103.

Table 14-103 OracleTimeStamp Static Operators

Operator	Description
operator +	Adds the supplied instance value to the supplied `OracleTimeStamp` and returns a new `OracleTimeStamp` structure (Overloaded)
operator ==	Determines if two `OracleTimeStamp` values are equal
operator >	Determines if the first of two `OracleTimeStamp` values is greater than the second
operator >=	Determines if the first of two `OracleTimeStamp` values is greater than or equal to the second
operator !=	Determines if the two `OracleTimeStamp` values are not equal
operator <	Determines if the first of two `OracleTimeStamp` values is less than the second
operator <=	Determines if the first of two `OracleTimeStamp` values is less than or equal to the second
operator -	Subtracts the supplied instance value from the supplied `OracleTimeStamp` and returns a new `OracleTimeStamp` structure (Overloaded)

OracleTimeStamp Static Type Conversions

The `OracleTimeStamp` static type conversions are listed in Table 14-104.

Table 14-104 OracleTimeStamp Static Type Conversions

Operator	Description
explicit operator OracleTimeStamp	Converts an instance value to an `OracleTimeStamp` structure (Overloaded)

Table 14-104 (Cont.) OracleTimeStamp Static Type Conversions

Operator	Description
implicit operator OracleTimeStamp	Converts an instance value to an `OracleTimeStamp` structure (Overloaded)
explicit operator DateTime	Converts an `OracleTimeStamp` value to a `DateTime` structure

OracleTimeStamp Properties

The `OracleTimeStamp` properties are listed in Table 14-105.

Table 14-105 OracleTimeStamp Properties

Properties	Description
BinData	Returns an array of bytes that represents an Oracle `TIMESTAMP` in Oracle internal format
Day	Specifies the day component of an `OracleTimeStamp`
IsNull	Indicates whether or not the `OracleTimeStamp` instance has a null value
Hour	Specifies the hour component of an `OracleTimeStamp`
Millisecond	Specifies the millisecond component of an `OracleTimeStamp`
Minute	Specifies the minute component of an `OracleTimeStamp`
Month	Specifies the month component of an `OracleTimeStamp`
Nanosecond	Specifies the nanosecond component of an `OracleTimeStamp`
Second	Specifies the second component of an `OracleTimeStamp`
Value	Specifies the date and time that is stored in the `OracleTimeStamp` structure
Year	Specifies the year component of an `OracleTimeStamp`

OracleTimeStamp Methods

The `OracleTimeStamp` methods are listed in Table 14-106.

Table 14-106 OracleTimeStamp Methods

Methods	Description
AddDays	Adds the supplied number of days to the current instance

Table 14-106 (Cont.) OracleTimeStamp Methods

Methods	Description
AddHours	Adds the supplied number of hours to the current instance
AddMilliseconds	Adds the supplied number of milliseconds to the current instance
AddMinutes	Adds the supplied number of minutes to the current instance
AddMonths	Adds the supplied number of months to the current instance
AddNanoseconds	Adds the supplied number of nanoseconds to the current instance
AddSeconds	Adds the supplied number of seconds to the current instance
AddYears	Adds the supplied number of years to the current instance
CompareTo	Compares the current `OracleTimeStamp` instance to an object, and returns an integer that represents their relative values
Equals	Determines whether or not an object has the same date and time as the current `OracleTimeStamp` instance (Overloaded)
GetHashCode	Returns a hash code for the `OracleTimeStamp` instance
GetDaysBetween	Subtracts an `OracleTimeStamp` value from the current instance and returns an `OracleIntervalDS` that represents the time difference between the supplied `OracleTimeStamp` and the current instance
GetYearsBetween	Subtracts `value1` from the current instance and returns an `OracleIntervalYM` that represents the difference between `value1` and the current instance using `OracleIntervalYM`
GetType	Inherited from `System.Object`
ToOracleDate	Converts the current `OracleTimeStamp` structure to an `OracleDate` structure
ToOracleTimeStampLTZ	Converts the current `OracleTimeStamp` structure to an `OracleTimeStampLTZ` structure
ToOracleTimeStampTZ	Converts the current `OracleTimeStamp` structure to an `OracleTimeStampTZ` structure
ToString	Converts the current `OracleTimeStamp` structure to a string

14.8.2 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.

14.8.2.1 OracleTimeStamp(DateTime)

This constructor creates a new instance of the `OracleTimeStamp` structure and sets its value for date and time using the supplied `DateTime` value.

Declaration

```
// C#
public OracleTimeStamp (DateTime dt);
```

Parameters

- *`dt`*

  The supplied `DateTime` value.

Exceptions

`ArgumentException` - The *`dt`* parameter cannot be used to construct a valid `OracleTimeStamp`.

14.8.2.2 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());
  }
}
```

14.8.2.3 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).

14.8.2.4 OracleTimeStamp(int, int, int, int, int, int)

This constructor creates a new instance of the `OracleTimeStamp` structure and sets its value for date and time using year, month, day, hour, minute, and second.

Declaration

```
// C#
public OracleTimeStamp (int year, int month, int day, int hour,
  int minute, int second);
```

Parameters

- *`year`*

  The year provided. Range of *`year`* is (-4712 to 9999).

- *`month`*

The month provided. Range of *month* is (1 to 12).

- *day*

  The day provided. Range of *day* is (1 to 31).
- *hour*

  The hour provided. Range of *hour* is (0 to 23).
- *minute*

  The minute provided. Range of *minute* is (0 to 59).
- *second*

  The second provided. Range of *second* is (0 to 59).

Exceptions

`ArgumentOutOfRangeException` - The argument value for one or more of the parameters is out of the specified range.

`ArgumentException` - The argument values of the parameters cannot be used to construct a valid `OracleTimeStamp` (that is, the day is out of range for the month).

14.8.2.5 OracleTimeStamp(int, int, int, int, int, int, double)

This constructor creates a new instance of the `OracleTimeStamp` structure and sets its value for date and time using year, month, day, hour, minute, second, and millisecond.

Declaration

```
// C#
public OracleTimeStamp(int year, int month, int day, int hour,
   int minute,  int second, double millisecond);
```

Parameters

- *year*

  The year provided. Range of `year` is (-4712 to 9999).
- *month*

  The month provided. Range of `month` is (1 to 12).
- *day*

  The day provided. Range of `day` is (1 to 31).
- *hour*

  The hour provided. Range of `hour` is (0 to 23).
- *minute*

  The minute provided. Range of `minute` is (0 to 59).
- *second*

  The second provided. Range of `second` is (0 to 59).
- *milliSeconds*

The milliseconds provided. Range of `millisecond` is (0 to 999.999999).

Exceptions

`ArgumentOutOfRangeException` - The argument value for one or more of the parameters is out of the specified range.

`ArgumentException` - The argument values of the parameters cannot be used to construct a valid `OracleTimeStamp` (that is, the day is out of range for the month).

14.8.2.6 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).

14.8.2.7 OracleTimeStamp(byte [ ])

This constructor creates a new instance of the `OracleTimeStamp` structure and sets its value to the provided byte array, which is in the internal Oracle `TIMESTAMP` format.

Declaration

```
// C#
public OracleTimeStamp (byte[] bytes);
```

Parameters

- *`bytes`*

  A byte array that represents an Oracle `TIMESTAMP` in Oracle internal format.

Exceptions

`ArgumentException` - *`bytes`* is not in an internal Oracle `TIMESTAMP` format or *`bytes`* is not a valid Oracle `TIMESTAMP`.

`ArgumentNullException` - *`bytes`* is null.

14.8.3 OracleTimeStamp Static Fields

The `OracleTimeStamp` static fields are listed in Table 14-107.

Table 14-107 OracleTimeStamp Static Fields

Field	Description
MaxValue	Represents the maximum valid date for an `OracleTimeStamp` structure, which is December 31, 9999 23:59:59.999999999
MinValue	Represents the minimum valid date for an `OracleTimeStamp` structure, which is January 1, -4712 0:0:0
Null	Represents a null value that can be assigned to an instance of the `OracleTimeStamp` structure

14.8.3.1 MaxValue

This static field represents the maximum valid date and time for an `OracleTimeStamp` structure, which is December 31, 9999 23:59:59.999999999.

Declaration

```
// C#
public static readonly OraTimestamp MaxValue;
```

14.8.3.2 MinValue

This static field represents the minimum valid date and time for an `OracleTimeStamp` structure, which is January 1, -4712 0:0:0.

Declaration

```
// C#
public static readonly OracleTimeStamp MinValue;
```

14.8.3.3 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;
```

14.8.4 OracleTimeStamp Static Methods

The `OracleTimeStamp` static methods are listed in NOT_SUPPORTED.

NOT_SUPPORTED

Methods	Description
Equals	Determines if two `OracleTimeStamp` values are equal (Overloaded)
GreaterThan	Determines if the first of two `OracleTimeStamp` values is greater than the second
GreaterThanOrEqual	Determines if the first of two `OracleTimeStamp` values is greater than or equal to the second
LessThan	Determines if the first of two `OracleTimeStamp` values is less than the second
LessThanOrEqual	Determines if the first of two `OracleTimeStamp` values is less than or equal to the second
NotEquals	Determines if two `OracleTimeStamp` values are not equal
GetSysDate	Gets an `OracleTimeStamp` structure that represents the current date and time
Parse	Gets an `OracleTimeStamp` structure and sets its value using the supplied string
SetPrecision	Returns a new instance of an `OracleTimeStamp` with the specified fractional second precision

14.8.4.1 Equals

This static method determines if two `OracleTimeStamp` values are equal.

Declaration

```
// 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 OracleTimeStamps that contain a null value are equal.

14.8.4.2 GreaterThan

This static method determines if the first of two OracleTimeStamp values is greater than the second.

Declaration

```
// C#
public static bool GreaterThan(OracleTimeStamp value1,
   OracleTimeStamp value2);
```

Parameters

- *value1*

  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 OracleTimeStamps that contain a null value are equal.

14.8.4.3 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 `OracleTimeStamp`s that contain a null value are equal.

14.8.4.4 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.

14.8.4.5 LessThanOrEqual

This static method determines if the first of two `OracleTimeStamp` values is less than or equal to the second.

Declaration

```
// C#
public static bool LessThanOrEqual(OracleTimeStamp value1,
     OracleTimeStamp value2);
```

Parameters

- *`value1`*

  The first `OracleTimeStamp`.
- *`value2`*

  The second `OracleTimeStamp`.

Return Value

Returns `true` if the first of two `OracleTimeStamp` values is less than or equal to the second. Returns `false` otherwise.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleTimeStamp` that has a value is greater than an `OracleTimeStamp` that has a null value.
- Two `OracleTimeStamps` that contain a null value are equal.

14.8.4.6 NotEquals

This static method determines if two `OracleTimeStamp` values are not equal.

Declaration

```
// C#
public static bool NotEquals(OracleTimeStamp value1,
   OracleTimeStamp value2);
```

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.

14.8.4.7 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.

14.8.4.8 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

```
// 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());
  }
}
```

14.8.4.9 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`.

14.8.5 OracleTimeStamp Static Operators

The `OracleTimeStamp` static operators are listed in Table 14-109.

Table 14-109 OracleTimeStamp Static Operators

Operator	Description
operator +	Adds the supplied instance value to the supplied `OracleTimeStamp` and returns a new `OracleTimeStamp` structure (Overloaded)
operator ==	Determines if two `OracleTimeStamp` values are equal
operator >	Determines if the first of two `OracleTimeStamp` values is greater than the second
operator >=	Determines if the first of two `OracleTimeStamp` values is greater than or equal to the second
operator !=	Determines if the two `OracleTimeStamp` values are not equal
operator <	Determines if the first of two `OracleTimeStamp` values is less than the second
operator <=	Determines if the first of two `OracleTimeStamp` values is less than or equal to the second
operator -	Subtracts the supplied instance value from the supplied `OracleTimeStamp` and returns a new `OracleTimeStamp` structure (Overloaded)

14.8.5.1 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.

14.8.5.2 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.

14.8.5.3 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.

14.8.5.4 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.

14.8.5.5 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.

14.8.5.6 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.

14.8.5.7 operator >=

This static operator determines if the first of two `OracleTimeStamp` values is greater than or equal to the second.

Declaration

```
// C#
public static bool operator >= (OracleTimeStamp value1,
  OracleTimeStamp value2);
```

Parameters

- *value1*

  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 `OracleTimeStamps` that contain a null value are equal.

14.8.5.8 operator !=

This static operator determines if two `OracleTimeStamp` values are not 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 two `OracleTimeStamp` values are not equal; otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleTimeStamp` that has a value is greater than an `OracleTimeStamp` that has a null value.
- Two `OracleTimeStamps` that contain a null value are equal.

14.8.5.9 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 `OracleTimeStamp`s that contain a null value are equal.

14.8.5.10 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 `OracleTimeStamp` that has a value is greater than an `OracleTimeStamp` that has a null value.
- Two `OracleTimeStamps` that contain a null value are equal.

14.8.5.11 operator -

`operator-` subtracts the supplied value, from the supplied `OracleTimeStamp` value, and returns a new `OracleTimeStamp` structure.

Overload List:

- operator - (OracleTimeStamp, OracleIntervalDS)

  This static operator subtracts the supplied `OracleIntervalDS` value, from the supplied `OracleTimeStamp` value, and return a new `OracleTimeStamp` structure.
- operator - (OracleTimeStamp, OracleIntervalYM)

  This static operator subtracts the supplied `OracleIntervalYM` value, from the supplied `OracleTimeStamp` value, and returns a new `OracleTimeStamp` structure.
- operator - (OracleTimeStamp, TimeSpan)

  This static operator subtracts the supplied `TimeSpan` value, from the supplied `OracleTimeStamp` value, and returns a new `OracleTimeStamp` structure.

14.8.5.12 operator - (OracleTimeStamp, OracleIntervalDS)

This static operator subtracts the supplied `OracleIntervalDS` value, from the supplied `OracleTimeStamp` value, and return a new `OracleTimeStamp` structure.

Declaration

```
// C#
public static operator - (OracleTimeStamp value1, OracleIntervalDS value2);
```

Parameters

- *`value1`*

  An `OracleTimeStamp`.
- *`value2`*

  An `OracleIntervalDS` instance.

Return Value

An `OracleTimeStamp` structure.

Remarks

If either parameter has a null value, the returned `OracleTimeStamp` has a null value.

14.8.5.13 operator - (OracleTimeStamp, OracleIntervalYM)

This static operator subtracts the supplied `OracleIntervalYM` value, from the supplied `OracleTimeStamp` value, and returns a new `OracleTimeStamp` structure.

Declaration

```
// C#
public static operator - (OracleTimeStamp value1, OracleIntervalYM value2);
```

Parameters

- *`value1`*

  An `OracleTimeStamp`.

- *`value2`*

  An `OracleIntervalYM` instance.

Return Value

An `OracleTimeStamp` structure.

Remarks

If either parameter has a null value, the returned `OracleTimeStamp` has a null value.

14.8.5.14 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.

14.8.6 OracleTimeStamp Static Type Conversions

The `OracleTimeStamp` static type conversions are listed in Table 14-110.

Table 14-110 OracleTimeStamp Static Type Conversions

Operator	Description
explicit operator OracleTimeStamp	Converts an instance value to an `OracleTimeStamp` structure (Overloaded)
implicit operator OracleTimeStamp	Converts an instance value to an `OracleTimeStamp` structure (Overloaded)
explicit operator DateTime	Converts an `OracleTimeStamp` value to a `DateTime` structure

14.8.6.1 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.

14.8.6.2 explicit operator OracleTimeStamp(OracleTimeStampLTZ)

This static type conversion operator converts an `OracleTimeStampLTZ` value to an `OracleTimeStamp` structure.

Declaration

```
// C#
public static explicit operator OracleTimeStamp(OracleTimeStampLTZ value1);
```

Parameters

- *`value1`*

  An `OracleTimeStampLTZ` instance.

Return Value

The returned `OracleTimeStamp` contains the date and time of the `OracleTimeStampLTZ` structure.

Remarks

If the `OracleTimeStampLTZ` structure has a null value, the returned `OracleTimeStamp` structure also has a null value.

14.8.6.3 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.

14.8.6.4 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());
  }
}
```

14.8.6.5 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.

14.8.6.6 implicit operator OracleTimeStamp(OracleDate)

This static type conversion operator converts an `OracleDate` value to an `OracleTimeStamp` structure.

Declaration

```
// C#
public static implicit operator OracleTimeStamp (OracleDate value1);
```

Parameters

- *`value1`*

  An `OracleDate` instance.

Return Value

An `OracleTimeStamp` structure that contains the date and time of the `OracleDate` structure, *`value1`*.

Remarks

If the `OracleDate` structure has a null value, the returned `OracleTimeStamp` structure also has a null value.

14.8.6.7 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.

14.8.6.8 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.

14.8.7 OracleTimeStamp Properties

The OracleTimeStamp properties are listed in Table 14-111.

Table 14-111 OracleTimeStamp Properties

Properties	Description
BinData	Returns an array of bytes that represents an Oracle TIMESTAMP in Oracle internal format
Day	Specifies the day component of an OracleTimeStamp
IsNull	Indicates whether or not the OracleTimeStamp instance has a null value
Hour	Specifies the hour component of an OracleTimeStamp
Millisecond	Specifies the millisecond component of an OracleTimeStamp
Minute	Specifies the minute component of an OracleTimeStamp
Month	Specifies the month component of an OracleTimeStamp
Nanosecond	Specifies the nanosecond component of an OracleTimeStamp
Second	Specifies the second component of an OracleTimeStamp
Value	Specifies the date and time that is stored in the OracleTimeStamp structure
Year	Specifies the year component of an OracleTimeStamp

14.8.7.1 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.

14.8.7.2 Day

This property specifies the day component of an `OracleTimeStamp`.

Declaration

```
// C#
public int Day{get;}
```

Property Value

A number that represents the day. Range of `Day` is (1 to 31).

Exceptions

`OracleNullValueException` - The current instance has a null value.

14.8.7.3 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`.

14.8.7.4 Hour

This property specifies the hour component of an `OracleTimeStamp`.

Declaration

```
// C#
public int Hour{get;}
```

Property Value

A number that represents the hour. Range of `hour` is (0 to 23).

Exceptions

`OracleNullValueException` - The current instance has a null value.

14.8.7.5 Millisecond

This property gets the millisecond component of an `OracleTimeStamp`.

Declaration

```
// C#
public double Millisecond{get;}
```

Property Value

A number that represents a millisecond. Range of `Millisecond` is (0 to 999.999999).

Exceptions

`OracleNullValueException` - The current instance has a null value.

14.8.7.6 Minute

This property gets the minute component of an `OracleTimeStamp`.

Declaration

```
// C#
public int Minute{get;}
```

Property Value

A number that represent a minute. Range of `Minute` is (0 to 59).

Exceptions

`OracleNullValueException` - The current instance has a null value.

14.8.7.7 Month

This property gets the month component of an `OracleTimeStamp`.

Declaration

```
// C#
public int Month{get;}
```

Property Value

A number that represents a month. Range of `Month` is (1 to 12).

Exceptions

`OracleNullValueException` - The current instance has a null value.

14.8.7.8 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.

14.8.7.9 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.

14.8.7.10 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.

14.8.7.11 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.

14.8.8 OracleTimeStamp Methods

The `OracleTimeStamp` methods are listed in Table 14-112.

Table 14-112 OracleTimeStamp Methods

Methods	Description
AddDays	Adds the supplied number of days to the current instance
AddHours	Adds the supplied number of hours to the current instance
AddMilliseconds	Adds the supplied number of milliseconds to the current instance
AddMinutes	Adds the supplied number of minutes to the current instance
AddMonths	Adds the supplied number of months to the current instance
AddNanoseconds	Adds the supplied number of nanoseconds to the current instance
AddSeconds	Adds the supplied number of seconds to the current instance
AddYears	Adds the supplied number of years to the current instance
CompareTo	Compares the current `OracleTimeStamp` instance to an object, and returns an integer that represents their relative values
Equals	Determines whether or not an object has the same date and time as the current `OracleTimeStamp` instance (Overloaded)
GetHashCode	Returns a hash code for the `OracleTimeStamp` instance
GetDaysBetween	Subtracts an `OracleTimeStamp` value from the current instance and returns an `OracleIntervalDS` that represents the time difference between the supplied `OracleTimeStamp` and the current instance
GetYearsBetween	Subtracts `value1` from the current instance and returns an `OracleIntervalYM` that represents the difference between `value1` and the current instance using `OracleIntervalYM`
GetType	Inherited from `System.Object`
ToOracleDate	Converts the current `OracleTimeStamp` structure to an `OracleDate` structure
ToOracleTimeStampLTZ	Converts the current `OracleTimeStamp` structure to an `OracleTimeStampLTZ` structure

Table 14-112 (Cont.) OracleTimeStamp Methods

Methods	Description
ToOracleTimeStampTZ	Converts the current `OracleTimeStamp` structure to an `OracleTimeStampTZ` structure
ToString	Converts the current `OracleTimeStamp` structure to a string

14.8.8.1 AddDays

This method adds the supplied number of days to the current instance.

Declaration

```
// C#
public OracleTimeStamp AddDays(double days);
```

Parameters

- *`days`*

  The supplied number of days. Range is (-1,000,000,000 < *`days`* < 1,000,000,000)

Return Value

An `OracleTimeStamp`.

Exceptions

`ArgumentOutofRangeException` - The argument value is out of the specified range.

`OracleNullValueException` - The current instance has a null value.

14.8.8.2 AddHours

This method adds the supplied number of hours to the current instance.

Declaration

```
// C#
public OracleTimeStamp AddHours(double hours);
```

Parameters

- *`hours`*

  The supplied number of hours. Range is (-24,000,000,000 < *`hours`* < 24,000,000,000).

Return Value

An `OracleTimeStamp`.

Exceptions

`ArgumentOutofRangeException` - The argument value is out of the specified range.

`OracleNullValueException` - The current instance has a null value.

14.8.8.3 AddMilliseconds

This method adds the supplied number of milliseconds to the current instance.

Declaration

```
// C#
public OracleTimeStamp AddMilliseconds(double milliseconds);
```

Parameters

- *`milliseconds`*

  The supplied number of milliseconds. Range is (-8.64 * 1016< `milliseconds` < 8.64 * 1016).

Return Value

An `OracleTimeStamp`.

Exceptions

`ArgumentOutofRangeException` - The argument value is out of the specified range.

`OracleNullValueException` - The current instance has a null value.

14.8.8.4 AddMinutes

This method adds the supplied number of minutes to the current instance.

Declaration

```
// C#
public OracleTimeStamp AddMinutes(double minutes);
```

Parameters

- *`minutes`*

  The supplied number of minutes. Range is (-1,440,000,000,000 < *`minutes`* < 1,440,000,000,000).

Return Value

An `OracleTimeStamp`.

Exceptions

`ArgumentOutofRangeException` - The argument value is out of the specified range.

`OracleNullValueException` - The current instance has a null value.

14.8.8.5 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.

14.8.8.6 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.

14.8.8.7 AddSeconds

This method adds the supplied number of seconds to the current instance.

Declaration

```
// C#
public OracleTimeStamp AddSeconds(double seconds);
```

Parameters

- *`seconds`*

The supplied number of seconds. Range is (-8.64 * 1013< `seconds` < 8.64 * 1013).

Return Value

An `OracleTimeStamp`.

Exceptions

`ArgumentOutofRangeException` - The argument value is out of the specified range.

`OracleNullValueException` - The current instance has a null value.

14.8.8.8 AddYears

This method adds the supplied number of years to the current instance.

Declaration

```
// C#
public OracleTimeStamp AddYears(int years);
```

Parameters

- *`years`*

  The supplied number of years. Range is (-999,999,999 <= *`years`* < = 999,999,999)

Return Value

An `OracleTimeStamp`.

Exceptions

`ArgumentOutofRangeException` - The argument value is out of the specified range.

`OracleNullValueException` - The current instance has a null value.

14.8.8.9 CompareTo

This method compares the current `OracleTimeStamp` instance to an object, and returns an integer that represents their relative values.

Declaration

```
// C#
public int CompareTo(object obj);
```

Parameters

- *`obj`*

  The object being compared to the current `OracleTimeStamp` instance.

Return Value

The method returns a number that is:

Less than zero: if the current `OracleTimeStamp` instance value is less than that of *`obj`*.

Zero: if the current `OracleTimeStamp` instance and *`obj`* values are equal.

Greater than zero: if the current `OracleTimeStamp` instance value is greater than that of *`obj`*.

Implements

`IComparable`

Exceptions

`ArgumentException` - The *`obj`* parameter is not of type `OracleTimeStamp`.

Remarks

The following rules apply to the behavior of this method.

- The comparison must be between `OracleTimeStamps`. For example, comparing an `OracleTimeStamp` instance with an `OracleBinary` instance is not allowed. When an `OracleTimeStamp` is compared with a different type, an `ArgumentException` is thrown.
- Any `OracleTimeStamp` that has a value is greater than an `OracleTimeStamp` that has a null value.
- Two `OracleTimeStamps` that contain a null value are equal.

14.8.8.10 Equals

Overrides `Object`

This method determines whether or not an object has the same date and time as the current `OracleTimeStamp` instance.

Declaration

```
// C#
public override bool Equals(object obj);
```

Parameters

- *`obj`*

  The object being compared to the current `OracleTimeStamp` instance.

Return Value

Returns `true` if the *`obj`* is of type `OracleTimeStamp` and represents the same date and time; otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleTimeStamp` that has a value is greater than an `OracleTimeStamp` that has a null value.
- Two `OracleTimeStamps` that contain a null value are equal.

14.8.8.11 GetHashCode

Overrides `Object`

This method returns a hash code for the `OracleTimeStamp` instance.

Declaration

```
// C#
public override int GetHashCode();
```

Return Value

A number that represents the hash code.

14.8.8.12 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.

14.8.8.13 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.

14.8.8.14 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.

14.8.8.15 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.

14.8.8.16 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.

14.8.8.17 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());
  }
}
```

14.9 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

```
// C#
public struct OracleTimeStampLTZ : IComparable, INullable, IXmlSerializable
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver
Assembly	`Oracle.DataAccess.dll`	`Oracle.ManagedDataAccess.dll`
Namespace	`Oracle.DataAccess.Types`	`Oracle.ManagedDataAccess.Types`
.NET Framework	3.5, 4.5, 4.6	4.5, 4.6

Thread Safety

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

Example

```
// C#

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

class OracleTimeStampLTZSample
{
  static void Main()
  {
    // Illustrates usage of OracleTimeStampLTZ
    // Display Local Time Zone Name
    Console.WriteLine("Local Time Zone Name = " +
      OracleTimeStampLTZ.GetLocalTimeZoneName());
    OracleTimeStampLTZ tsLocal1 = OracleTimeStampLTZ.GetSysDate();
    OracleTimeStampLTZ tsLocal2 = DateTime.Now;

    // Calculate the difference between tsLocal1 and tsLocal2
    OracleIntervalDS idsDiff = tsLocal2.GetDaysBetween(tsLocal1);

    // Calculate the difference using AddNanoseconds()
    int nanoDiff = 0;
    while (tsLocal2 > tsLocal1)
    {
      nanoDiff += 10;
      tsLocal1 = tsLocal1.AddNanoseconds(10);
    }
    Console.WriteLine("idsDiff.Nanoseconds = " + idsDiff.Nanoseconds);
    Console.WriteLine("nanoDiff = " + nanoDiff);
  }
}
```

14.9.1 OracleTimeStampLTZ Members

`OracleTimeStampLTZ` members are listed in the following tables:

OracleTimeStampLTZ Constructors

`OracleTimeStampLTZ` constructors are listed in Table 14-113

Table 14-113   OracleTimeStampLTZConstructors

Constructor	Description
OracleTimeStampLTZ Constructors	Instantiates a new instance of `OracleTimeStampLTZ` structure (Overloaded)

OracleTimeStampLTZ Static Fields

The `OracleTimeStampLTZ` static fields are listed in Table 14-114.

Table 14-114   OracleTimeStampLTZ Static Fields

Field	Description
MaxValue	Represents the maximum valid date for an `OracleTimeStampLTZ` structure, which is December 31, 9999 23:59:59.999999999
MinValue	Represents the minimum valid date for an `OracleTimeStampLTZ` structure, which is January 1, -4712 0:0:0
Null	Represents a null value that can be assigned to an instance of the `OracleTimeStampLTZ` structure

OracleTimeStampLTZ Static Methods

The `OracleTimeStampLTZ` static methods are listed in Table 14-115.

Table 14-115 OracleTimeStampLTZ Static Methods

Methods	Description
Equals	Determines if two `OracleTimeStampLTZ` values are equal (Overloaded)
GetLocalTimeZoneName	Gets the client's local time zone name
GetLocalTimeZoneOffset	Gets the client's local time zone offset relative to UTC
GetSysDate	Gets an `OracleTimeStampLTZ` structure that represents the current date and time
GreaterThan	Determines if the first of two `OracleTimeStampLTZ` values is greater than the second
GreaterThanOrEqual	Determines if the first of two `OracleTimeStampLTZ` values is greater than or equal to the second
LessThan	Determines if the first of two `OracleTimeStampLTZ` values is less than the second
LessThanOrEqual	Determines if the first of two `OracleTimeStampLTZ` values is less than or equal to the second
NotEquals	Determines if two `OracleTimeStampLTZ` values are not equal
Parse	Gets an `OracleTimeStampLTZ` structure and sets its value for date and time using the supplied string
SetPrecision	Returns a new instance of an `OracleTimeStampLTZ` with the specified fractional second precision

OracleTimeStampLTZ Static Operators

The `OracleTimeStampLTZ` static operators are listed in Table 14-116.

Table 14-116 OracleTimeStampLTZ Static Operators

Operator	Description
operator +	Adds the supplied instance value to the supplied `OracleTimeStampLTZ` and returns a new `OracleTimeStampLTZ` structure (Overloaded)
operator ==	Determines if two `OracleTimeStampLTZ` values are equal
operator >	Determines if the first of two `OracleTimeStampLTZ` values is greater than the second
operator >=	Determines if the first of two `OracleTimeStampLTZ` values is greater than or equal to the second
operator !=	Determines if two `OracleTimeStampLTZ` values are not equal

Table 14-116 (Cont.) OracleTimeStampLTZ Static Operators

Operator	Description
operator <	Determines if the first of two `OracleTimeStampLTZ` values is less than the second
operator <=	Determines if the first of two `OracleTimeStampLTZ` values is less than or equal to the second
operator -	Subtracts the supplied instance value from the supplied `OracleTimeStampLTZ` and returns a new `OracleTimeStampLTZ` structure (Overloaded)

OracleTimeStampLTZ Static Type Conversions

The `OracleTimeStampLTZ` static type conversions are listed in Table 14-117.

Table 14-117 OracleTimeStampLTZ Static Type Conversions

Operator	Description
explicit operator OracleTimeStampLTZ	Converts an instance value to an `OracleTimeStampLTZ` structure (Overloaded)
implicit operator OracleTimeStampLTZ	Converts an instance value to an `OracleTimeStampLTZ` structure (Overloaded)
explicit operator DateTime	Converts an `OracleTimeStampLTZ` value to a `DateTime` structure

OracleTimeStampLTZ Properties

The `OracleTimeStampLTZ` properties are listed in Table 14-118.

Table 14-118 OracleTimeStampLTZ Properties

Properties	Description
BinData	Returns an array of bytes that represents an `Oracle TIMESTAMP WITH LOCAL TIME ZONE` in Oracle internal format
Day	Specifies the `day` component of an `OracleTimeStampLTZ`
IsNull	Indicates whether or not the `OracleTimeStampLTZ` instance has a null value
Hour	Specifies the hour component of an `OracleTimeStampLTZ`
Millisecond	Specifies the millisecond component of an `OracleTimeStampLTZ`
Minute	Specifies the minute component of an `OracleTimeStampLTZ`
Month	Specifies the month component of an `OracleTimeStampLTZ`

Table 14-118 (Cont.) OracleTimeStampLTZ Properties

Properties	Description
Nanosecond	Specifies the nanosecond component of an `OracleTimeStampLTZ`
Second	Specifies the second component of an `OracleTimeStampLTZ`
Value	Specifies the date and time that is stored in the `OracleTimeStampLTZ` structure
Year	Specifies the year component of an `OracleTimeStampLTZ`

OracleTimeStampLTZ Methods

The `OracleTimeStampLTZ` methods are listed in Table 14-119.

Table 14-119 OracleTimeStampLTZ Methods

Methods	Description
AddDays	Adds the supplied number of days to the current instance
AddHours	Adds the supplied number of hours to the current instance
AddMilliseconds	Adds the supplied number of milliseconds to the current instance
AddMinutes	Adds the supplied number of minutes to the current instance
AddMonths	Adds the supplied number of months to the current instance
AddNanoseconds	Adds the supplied number of nanoseconds to the current instance
AddSeconds	Adds the supplied number of seconds to the current instance
AddYears	Adds the supplied number of years to the current instance
CompareTo	Compares the current `OracleTimeStampLTZ` instance to an object and returns an integer that represents their relative values
Equals	Determines whether or not an object has the same date and time as the current `OracleTimeStampLTZ` instance (Overloaded)
GetHashCode	Returns a hash code for the `OracleTimeStampLTZ` instance
GetDaysBetween	Subtracts an `OracleTimeStampLTZ` from the current instance and returns an `OracleIntervalDS` that represents the difference

Table 14-119 (Cont.) OracleTimeStampLTZ Methods

Methods	Description
GetYearsBetween	Subtracts an `OracleTimeStampLTZ` from the current instance and returns an `OracleIntervalYM` that represents the difference
GetType	Inherited from `System.Object`
ToOracleDate	Converts the current `OracleTimeStampLTZ` structure to an `OracleDate` structure
ToOracleTimeStamp	Converts the current `OracleTimeStampLTZ` structure to an `OracleTimeStamp` structure
ToOracleTimeStampTZ	Converts the current `OracleTimeStampLTZ` structure to an `OracleTimeStampTZ` structure
ToString	Converts the current `OracleTimeStampLTZ` structure to a string
ToUniversalTime	Converts the current local time to Coordinated Universal Time (UTC)

14.9.2 OracleTimeStampLTZ Constructors

The `OracleTimeStampLTZ` constructors create new instances of the `OracleTimeStampLTZ` structure.

Overload List:

- OracleTimeStampLTZ(DateTime)

  This constructor creates a new instance of the `OracleTimeStampLTZ` structure and sets its value for date and time using the supplied `DateTime` value.
- OracleTimeStampLTZ(string)

  This constructor creates a new instance of the `OracleTimeStampLTZ` structure and sets its value for date and time using the supplied string.
- OracleTimeStampLTZ(int, int, int)

  This constructor creates a new instance of the `OracleTimeStampLTZ` structure and sets its value for date using year, month, and day.
- OracleTimeStampLTZ(int, int, int, int, int, int)

  This constructor creates a new instance of the `OracleTimeStampLTZ` structure and sets its value for date and time using year, month, day, hour, minute, and second.
- OracleTimeStampLTZ(int, int, int, int, int, int, double)

  This constructor creates a new instance of the `OracleTimeStampLTZ` structure and sets its value for date and time using year, month, day, hour, minute, second, and millisecond.
- OracleTimeStampLTZ(int, int, int, int, int, int, int)

This constructor creates a new instance of the `OracleTimeStampLTZ` structure and sets its value for date and time using year, month, day, hour, minute, second, and nanosecond.

- OracleTimeStampLTZ(byte [ ])

  This constructor creates a new instance of the `OracleTimeStampLTZ` structure and sets its value to the provided byte array, which is in the internal Oracle `TIMESTAMP WITH LOCAL TIME ZONE` format.

14.9.2.1 OracleTimeStampLTZ(DateTime)

This constructor creates a new instance of the `OracleTimeStampLTZ` structure and sets its value for date and time using the supplied `DateTime` value.

Declaration

```
// C#
public OracleTimeStampLTZ (DateTime dt);
```

Parameters

- *`dt`*

  The supplied `DateTime` value.

Exceptions

`ArgumentException` - The *`dt`* parameter cannot be used to construct a valid `OracleTimeStampLTZ`.

14.9.2.2 OracleTimeStampLTZ(string)

This constructor creates a new instance of the `OracleTimeStampLTZ` structure and sets its value for date and time using the supplied string.

Declaration

```
// C#
public OracleTimeStampLTZ(string tsStr);
```

Parameters

- *`tsStr`*

  A string that represents an Oracle `TIMESTAMP WITH LOCAL TIME ZONE`.

Exceptions

`ArgumentException` - The *`tsStr`* is an invalid string representation of an Oracle `TIMESTAMP WITH LOCAL TIME ZONE` or the supplied *`tsStr`* is not in the timestamp format specified by the `OracleGlobalization.TimeStampFormat` property of the thread, which represents 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.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());
  }
}
```

14.9.2.3 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).

14.9.2.4 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).

14.9.2.5 OracleTimeStampLTZ(int, int, int, int, int, int, double)

This constructor creates a new instance of the `OracleTimeStampLTZ` structure and sets its value for date and time using year, month, day, hour, minute, second, and millisecond.

Declaration

```
// C#
public OracleTimeStampLTZ(int year, int month, int day, int hour, int minute, int
second, double millisecond);
```

Parameters

- *`year`*

  The year provided. Range of *`year`* is (-4712 to 9999).

- *`month`*

  The month provided. Range of *`month`* is (1 to 12).

- *`day`*

  The day provided. Range of *`day`* is (1 to 31).

- *`hour`*

  The hour provided. Range of *`hour`* is (0 to 23).

- *`minute`*

  The minute provided. Range of *`minute`* is (0 to 59).

- *`second`*

  The second provided. Range of *`second`* is (0 to 59).

- *`milliSeconds`*

  The milliseconds provided. Range of *`millisecond`* is (0 to 999.999999).

Exceptions

`ArgumentOutOfRangeException` - The argument value for one or more of the parameters is out of the specified range.

`ArgumentException` - The argument values of the parameters cannot be used to construct a valid `OracleTimeStampLTZ` (that is, the day is out of range for the month).

14.9.2.6 OracleTimeStampLTZ(int, int, int, int, int, int, int)

This constructor creates a new instance of the `OracleTimeStampLTZ` structure and sets its value for date and time using year, month, day, hour, minute, second, and nanosecond.

Declaration

```
// C#
public OracleTimeStampLTZ (int year, int month, int day, int hour,
  int minute, int second, int nanosecond);
```

Parameters

- *year*

  The year provided. Range of *year* is (-4712 to 9999).
- *month*

  The month provided. Range of *month* is (1 to 12).
- *day*

  The day provided. Range of *day* is (1 to 31).
- *hour*

  The hour provided. Range of *hour* is (0 to 23).
- *minute*

  The minute provided. Range of *minute* is (0 to 59).
- *second*

  The second provided. Range of *second* is (0 to 59).
- *nanosecond*

  The nanosecond provided. Range of *nanosecond* is (0 to 999999999).

Exceptions

ArgumentOutOfRangeException - The argument value for one or more of the parameters is out of the specified range.

ArgumentException - The argument values of the parameters cannot be used to construct a valid OracleTimeStampLTZ (that is, the day is out of range for the month).

14.9.2.7 OracleTimeStampLTZ(byte [ ])

This constructor creates a new instance of the OracleTimeStampLTZ structure and sets its value to the provided byte array, which is in the internal Oracle TIMESTAMP WITH LOCAL TIME ZONE format.

Declaration

```
// C#
public OracleTimeStampLTZ (byte[] bytes);
```

Parameters

- *bytes*

  A byte array that represents an Oracle TIMESTAMP WITH LOCAL TIME ZONE in Oracle internal format.

Exceptions

ArgumentException - *bytes* is not in an internal Oracle TIMESTAMP WITH LOCAL TIME ZONE format or *bytes* is not a valid Oracle TIMESTAMP WITH LOCAL TIME ZONE.

ArgumentNullException - *bytes* is null.

14.9.3 OracleTimeStampLTZ Static Fields

The `OracleTimeStampLTZ` static fields are listed in Table 14-120.

Table 14-120 OracleTimeStampLTZ Static Fields

Field	Description
MaxValue	Represents the maximum valid date for an `OracleTimeStampLTZ` structure, which is December 31, 9999 23:59:59.999999999
MinValue	Represents the minimum valid date for an `OracleTimeStampLTZ` structure, which is January 1, -4712 0:0:0
Null	Represents a null value that can be assigned to an instance of the `OracleTimeStampLTZ` structure

14.9.3.1 MaxValue

This static field represents the maximum valid date for an `OracleTimeStampLTZ` structure, which is December 31, 9999 23:59:59.999999999.

Declaration

```
// C#
public static readonly OracleTimeStampLTZ MaxValue;
```

Remarks

This value is the maximum date and time in the client time zone.

14.9.3.2 MinValue

This static field represents the minimum valid date for an `OracleTimeStampLTZ` structure, which is January 1, -4712 0:0:0.

Declaration

```
// C#
public static readonly OracleTimeStampLTZ MinValue;
```

Remarks

This value is the minimum date and time in the client time zone.

14.9.3.3 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;
```

14.9.4 OracleTimeStampLTZ Static Methods

The `OracleTimeStampLTZ` static methods are listed in Table 14-121.

Table 14-121 OracleTimeStampLTZ Static Methods

Methods	Description
Equals	Determines if two `OracleTimeStampLTZ` values are equal (Overloaded)
GetLocalTimeZoneName	Gets the client's local time zone name
GetLocalTimeZoneOffset	Gets the client's local time zone offset relative to UTC
GetSysDate	Gets an `OracleTimeStampLTZ` structure that represents the current date and time
GreaterThan	Determines if the first of two `OracleTimeStampLTZ` values is greater than the second
GreaterThanOrEqual	Determines if the first of two `OracleTimeStampLTZ` values is greater than or equal to the second
LessThan	Determines if the first of two `OracleTimeStampLTZ` values is less than the second
LessThanOrEqual	Determines if the first of two `OracleTimeStampLTZ` values is less than or equal to the second
NotEquals	Determines if two `OracleTimeStampLTZ` values are not equal
Parse	Gets an `OracleTimeStampLTZ` structure and sets its value for date and time using the supplied string
SetPrecision	Returns a new instance of an `OracleTimeStampLTZ` with the specified fractional second precision

14.9.4.1 Equals

This static method determines if two `OracleTimeStampLTZ` values are equal.

Declaration

```
// C#
public static bool Equals(OracleTimeStampLTZ value1,
  OracleTimeStampLTZ value2);
```

Parameters

- *`value1`*

  The first `OracleTimeStampLTZ`.

- *`value2`*

  The second `OracleTimeStampLTZ`.

Return Value

Returns `true` if two `OracleTimeStampLTZ` values are equal. Returns `false` otherwise.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleTimeStampLTZ` that has a value is greater than an `OracleTimeStampLTZ` that has a null value.
- Two `OracleTimeStampLTZ`s that contain a null value are equal.

14.9.4.2 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.

14.9.4.3 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.

14.9.4.4 GetSysDate

This static method gets an `OracleTimeStampLTZ` structure that represents the current date and time.

Declaration

```
// C#
public static OracleTimeStampLTZ GetSysDate();
```

Return Value

An `OracleTimeStampLTZ` structure that represents the current date and time.

14.9.4.5 GreaterThan

This static method determines if the first of two `OracleTimeStampLTZ` values is greater than the second.

Declaration

```
// C#
public static bool GreaterThan(OracleTimeStampLTZ value1,
   OracleTimeStampLTZ value2);
```

Parameters

- *value1*

  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 `OracleTimeStampLTZ`s that contain a null value are equal.

14.9.4.6 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 `OracleTimeStampLTZ`s that contain a null value are equal.

14.9.4.7 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 `OracleTimeStampLTZ`s that contain a null value are equal.

14.9.4.8 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 `OracleTimeStampLTZ`s that contain a null value are equal.

14.9.4.9 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 `OracleTimeStampLTZ`s that contain a null value are equal.

14.9.4.10 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.

`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 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());
  }
}
```

14.9.4.11 SetPrecision

This static method returns a new instance of an `OracleTimeStampLTZ` with the specified fractional second precision.

Declaration

```
// C#
public static OracleTimeStampLTZ SetPrecision(OracleTimeStampLTZ value1,
     int fracSecPrecision);
```

Parameters

- *`value1`*

  The provided `OracleTimeStampLTZ` object.
- *`fracSecPrecision`*

  The fractional second precision provided. Range of fractional second precision is (0 to 9).

Return Value

An `OracleTimeStampLTZ` structure with the specified fractional second precision

Exceptions

`ArgumentOutOfRangeException` - *`fracSecPrecision`* is out of the specified range.

Remarks

The value specified in the supplied *`fracSecPrecision`* parameter is used to perform a rounding off operation on the supplied `OracleTimeStampLTZ` value. Depending on this value, `0` or more trailing zeros are displayed in the string returned by `ToString()`.

Example

The `OracleTimeStampLTZ` with a value of "`December 31, 9999 23:59:59.99`" results in the string "`December 31, 9999 23:59:59.99000`" when `SetPrecision()` is called with the fractional second precision set to `5`.

14.9.5 OracleTimeStampLTZ Static Operators

The `OracleTimeStampLTZ` static operators are listed in Table 14-122.

Table 14-122 OracleTimeStampLTZ Static Operators

Operator	Description
operator +	Adds the supplied instance value to the supplied `OracleTimeStampLTZ` and returns a new `OracleTimeStampLTZ` structure (Overloaded)
operator ==	Determines if two `OracleTimeStampLTZ` values are equal
operator >	Determines if the first of two `OracleTimeStampLTZ` values is greater than the second

Table 14-122 (Cont.) OracleTimeStampLTZ Static Operators

Operator	Description
operator >=	Determines if the first of two `OracleTimeStampLTZ` values is greater than or equal to the second
operator !=	Determines if two `OracleTimeStampLTZ` values are not equal
operator <	Determines if the first of two `OracleTimeStampLTZ` values is less than the second
operator <=	Determines if the first of two `OracleTimeStampLTZ` values is less than or equal to the second
operator -	Subtracts the supplied instance value from the supplied `OracleTimeStampLTZ` and returns a new `OracleTimeStampLTZ` structure (Overloaded)

14.9.5.1 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.

14.9.5.2 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.

14.9.5.3 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.

14.9.5.4 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.

14.9.5.5 operator ==

This static operator determines if two `OracleTimeStampLTZ` values are equal.

Declaration

```
// C#
public static bool operator == (OracleTimeStampLTZ value1,
  OracleTimeStampLTZ  value2);
```

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 `OracleTimeStampLTZ`s that contain a null value are equal.

14.9.5.6 operator >

This static operator determines if the first of two `OracleTimeStampLTZ` values is greater than the second.

Declaration

```
// C#
public static bool operator > (OracleTimeStampLTZ value1,
     OracleTimeStampLTZ value2);
```

Parameters

- *`value1`*

  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 `OracleTimeStampLTZ`s that contain a null value are equal.

14.9.5.7 operator >=

This static operator determines if the first of two `OracleTimeStampLTZ` values is greater than or equal to the second.

Declaration

```
// C#
public static bool operator >= (OracleTimeStampLTZ value1,
    OracleTimeStampLTZ value2);
```

Parameters

- *value1*

  An `OracleTimeStampLTZ`.
- *value2*

  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.

14.9.5.8 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`.
- *value2*

  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.

14.9.5.9 operator <

This static operator determines if the first of two `OracleTimeStampLTZ` values is less than the second.

Declaration

```
// C#
public static bool operator < (OracleTimeStampLTZ value1,
     OracleTimeStampLTZ value2);
```

Parameters

- *value1*

  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.

14.9.5.10 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`s that contain a null value are equal.

14.9.5.11 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.

14.9.5.12 operator - (OracleTimeStampLTZ, OracleIntervalDS)

This static operator subtracts the supplied `OracleIntervalDS` value, from the supplied `OracleTimeStampLTZ` value, and return a new `OracleTimeStampLTZ` structure.

Declaration

```
// C#
public static operator - (OracleTimeStampLTZ value1,
  OracleIntervalDS value2);
```

Parameters

- *`value1`*

  An `OracleTimeStampLTZ`.
- *`value2`*

  An `OracleIntervalDS` instance.

Return Value

An `OracleTimeStampLTZ` structure.

Remarks

If either parameter has a null value, the returned `OracleTimeStampLTZ` has a null value.

14.9.5.13 operator - (OracleTimeStampLTZ, OracleIntervalYM)

This static operator subtracts the supplied `OracleIntervalYM` value, from the supplied `OracleTimeStampLTZ` value, and returns a new `OracleTimeStampLTZ` structure.

Declaration

```
// C#
public static operator - (OracleTimeStampLTZ value1,
  OracleIntervalYM value2);
```

Parameters

- *`value1`*

  An `OracleTimeStampLTZ`.
- *`value2`*

  An `OracleIntervalYM`.

Return Value

An `OracleTimeStampLTZ` structure.

Remarks

If either parameter has a null value, the returned `OracleTimeStampLTZ` has a null value.

14.9.5.14 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.

14.9.6 OracleTimeStampLTZ Static Type Conversions

The `OracleTimeStampLTZ` static type conversions are listed in Table 14-123.

Table 14-123 OracleTimeStampLTZ Static Type Conversions

Operator	Description
explicit operator OracleTimeStampLTZ	Converts an instance value to an `OracleTimeStampLTZ` structure (Overloaded)
implicit operator OracleTimeStampLTZ	Converts an instance value to an `OracleTimeStampLTZ` structure (Overloaded)
explicit operator DateTime	Converts an `OracleTimeStampLTZ` value to a `DateTime` structure

14.9.6.1 explicit operator OracleTimeStampLTZ

`explicit operator OracleTimeStampLTZ` converts the supplied value to an `OracleTimeStampLTZ` structure.

Overload List:

- explicit operator OracleTimeStampLTZ(OracleTimeStamp)

  This static type conversion operator converts an `OracleTimeStamp` value to an `OracleTimeStampLTZ` structure.

- explicit operator OracleTimeStampLTZ(OracleTimeStampTZ)

  This static type conversion operator converts an `OracleTimeStampTZ` value to an `OracleTimeStampLTZ` structure.
- explicit operator OracleTimeStampLTZ(string)

  This static type conversion operator converts the supplied string to an `OracleTimeStampLTZ` structure.

14.9.6.2 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.

14.9.6.3 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.

14.9.6.4 explicit operator OracleTimeStampLTZ(string)

This static type conversion operator converts the supplied string to an `OracleTimeStampLTZ` structure.

Declaration

```
// C#
public static explicit operator OracleTimeStampLTZ (string tsStr);
```

Parameters

- *`tsStr`*

  A string representation of an Oracle `TIMESTAMP WITH LOCAL TIME ZONE`.

Return Value

A `OracleTimeStampLTZ`.

Exceptions

`ArgumentException` - The*`tsStr`* parameter is an invalid string representation of an Oracle `TIMESTAMP WITH LOCAL TIME ZONE` or the *`tsStr`* is not in the timestamp format specified by the thread's `OracleGlobalization.TimeStampFormat` property, which represents 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());
  }
}
```

14.9.6.5 implicit operator OracleTimeStampLTZ

`implicit operator OracleTimeStampLTZ` converts the supplied structure to an `OracleTimeStampLTZ` structure.

Overload List:

- implicit operator OracleTimeStampLTZ(OracleDate)

  This static type conversion operator converts an `OracleDate` value to an `OracleTimeStampLTZ` structure.

- implicit operator OracleTimeStampLTZ(DateTime)

  This static type conversion operator converts a `DateTime` structure to an `OracleTimeStampLTZ` structure.

14.9.6.6 implicit operator OracleTimeStampLTZ(OracleDate)

This static type conversion operator converts an `OracleDate` value to an `OracleTimeStampLTZ` structure.

Declaration

```
// C#
public static implicit operator OracleTimeStampLTZ(OracleDate value1);
```

Parameters

- *`value1`*

  An `OracleDate`.

Return Value

The returned `OracleTimeStampLTZ` structure contains the date and time in the `OracleDate` structure.

Remarks

If the `OracleDate` structure has a null value, the returned `OracleTimeStampLTZ` structure also has a null value.

14.9.6.7 implicit operator OracleTimeStampLTZ(DateTime)

This static type conversion operator converts a `DateTime` structure to an `OracleTimeStampLTZ` structure.

Declaration

```
// C#
public static implicit operator OracleTimeStampLTZ(DateTime value1);
```

Parameters

- *`value1`*

  A `DateTime` structure.

Return Value

An `OracleTimeStampLTZ` structure.

14.9.6.8 explicit operator DateTime

This static type conversion operator converts an `OracleTimeStampLTZ` value to a `DateTime` structure.

Declaration

```
// C#
public static explicit operator DateTime(OracleTimeStampLTZ value1);
```

Parameters

- *`value1`*

  An `OracleTimeStampLTZ` instance.

Return Value

A `DateTime` that contains the date and time in the current instance.

Exceptions

`OracleNullValueException` - The `OracleTimeStampLTZ` structure has a null value.

Remarks

The precision of the `OracleTimeStampLTZ` value can be lost during the conversion.

14.9.7 OracleTimeStampLTZ Properties

The `OracleTimeStampLTZ` properties are listed in Table 14-124.

Table 14-124 OracleTimeStampLTZ Properties

Properties	Description
BinData	Returns an array of bytes that represents an `Oracle TIMESTAMP WITH LOCAL TIME ZONE` in Oracle internal format
Day	Specifies the `day` component of an `OracleTimeStampLTZ`
IsNull	Indicates whether or not the `OracleTimeStampLTZ` instance has a null value
Hour	Specifies the hour component of an `OracleTimeStampLTZ`
Millisecond	Specifies the millisecond component of an `OracleTimeStampLTZ`
Minute	Specifies the minute component of an `OracleTimeStampLTZ`
Month	Specifies the month component of an `OracleTimeStampLTZ`
Nanosecond	Specifies the nanosecond component of an `OracleTimeStampLTZ`
Second	Specifies the second component of an `OracleTimeStampLTZ`
Value	Specifies the date and time that is stored in the `OracleTimeStampLTZ` structure
Year	Specifies the year component of an `OracleTimeStampLTZ`

14.9.7.1 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.

14.9.7.2 Day

This property specifies the day component of an `OracleTimeStampLTZ`.

Declaration

```
// C#
public int Day{get;}
```

Property Value

A number that represents the day. Range of `Day` is (1 to 31).

Exceptions

`OracleNullValueException` - The current instance has a null value.

14.9.7.3 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 contains a null value; otherwise, returns `false`.

14.9.7.4 Hour

This property specifies the hour component of an `OracleTimeStampLTZ`.

Declaration

```
// C#
public int Hour{get;}
```

Property Value

A number that represents the hour. Range of `Hour` is (0 to 23).

Exceptions

`OracleNullValueException` - The current instance has a null value.

14.9.7.5 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.

14.9.7.6 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.

14.9.7.7 Month

This property gets the month component of an `OracleTimeStampLTZ`.

Declaration

```
// C#
public int Month{get;}
```

Property Value

A number that represents a month. Range of `Month` is (1 to 12).

Exceptions

`OracleNullValueException` - The current instance has a null value.

14.9.7.8 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.

14.9.7.9 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.

14.9.7.10 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.

14.9.7.11 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.

14.9.8 OracleTimeStampLTZ Methods

The `OracleTimeStampLTZ` methods are listed in Table 14-125.

Table 14-125 OracleTimeStampLTZ Methods

Methods	Description
AddDays	Adds the supplied number of days to the current instance
AddHours	Adds the supplied number of hours to the current instance

Table 14-125 (Cont.) OracleTimeStampLTZ Methods

Methods	Description
AddMilliseconds	Adds the supplied number of milliseconds to the current instance
AddMinutes	Adds the supplied number of minutes to the current instance
AddMonths	Adds the supplied number of months to the current instance
AddNanoseconds	Adds the supplied number of nanoseconds to the current instance
AddSeconds	Adds the supplied number of seconds to the current instance
AddYears	Adds the supplied number of years to the current instance
CompareTo	Compares the current `OracleTimeStampLTZ` instance to an object and returns an integer that represents their relative values
Equals	Determines whether or not an object has the same date and time as the current `OracleTimeStampLTZ` instance (Overloaded)
GetHashCode	Returns a hash code for the `OracleTimeStampLTZ` instance
GetDaysBetween	Subtracts an `OracleTimeStampLTZ` from the current instance and returns an `OracleIntervalDS` that represents the difference
GetYearsBetween	Subtracts an `OracleTimeStampLTZ` from the current instance and returns an `OracleIntervalYM` that represents the difference
GetType	Inherited from `System.Object`
ToOracleDate	Converts the current `OracleTimeStampLTZ` structure to an `OracleDate` structure
ToOracleTimeStamp	Converts the current `OracleTimeStampLTZ` structure to an `OracleTimeStamp` structure
ToOracleTimeStampTZ	Converts the current `OracleTimeStampLTZ` structure to an `OracleTimeStampTZ` structure
ToString	Converts the current `OracleTimeStampLTZ` structure to a string
ToUniversalTime	Converts the current local time to Coordinated Universal Time (UTC)

14.9.8.1 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.

14.9.8.2 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.

14.9.8.3 AddMilliseconds

This method adds the supplied number of milliseconds to the current instance.

Declaration

```
// C#
public OracleTimeStampLTZ AddMilliseconds(double milliseconds);
```

Parameters

- *milliseconds*

  The supplied number of milliseconds. Range is (-8.64 * 1016< `milliseconds` < 8.64 * 1016).

Return Value

An `OracleTimeStampLTZ`.

Exceptions

`OracleNullValueException` - The current instance has a null value.

`ArgumentOutofRangeException` - The argument value is out of the specified range.

14.9.8.4 AddMinutes

This method adds the supplied number of minutes to the current instance.

Declaration

```
// C#
public OracleTimeStampLTZ AddMinutes(double minutes);
```

Parameters

- *`minutes`*

  The supplied number of minutes. Range is (-1,440,000,000,000 < *`minutes`* < 1,440,000,000,000).

Return Value

An `OracleTimeStampLTZ`.

Exceptions

`OracleNullValueException` - The current instance has a null value.

`ArgumentOutofRangeException` - The argument value is out of the specified range.

14.9.8.5 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.

14.9.8.6 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.

14.9.8.7 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.

14.9.8.8 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.

14.9.8.9 CompareTo

This method compares the current `OracleTimeStampLTZ` instance to an object, and returns an integer that represents their relative values.

Declaration

```
// C#
public int CompareTo(object obj);
```

Parameters

- *`obj`*

  The object being compared to the current `OracleTimeStampLTZ` instance.

Return Value

The method returns a number that is:

- Less than zero: if the current `OracleTimeStampLTZ` instance value is less than that of *`obj`*.
- Zero: if the current `OracleTimeStampLTZ` instance and *`obj`* values are equal.
- Greater than zero: if the current `OracleTimeStampLTZ` instance value is greater than that of *`obj`*.

Implements

`IComparable`

Exceptions

`ArgumentException` - The *`obj`* parameter is not of type `OracleTimeStampLTZ`.

Remarks

The following rules apply to the behavior of this method.

- The comparison must be between `OracleTimeStampLTZ`s. For example, comparing an `OracleTimeStampLTZ` instance with an `OracleBinary` instance is not allowed. When an `OracleTimeStampLTZ` is compared with a different type, an `ArgumentException` is thrown.

- Any `OracleTimeStampLTZ` that has a value is greater than an `OracleTimeStampLTZ` that has a null value.
- Two `OracleTimeStampLTZ`s that contain a null value are equal.

14.9.8.10 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 `OracleTimeStampLTZ`s that contain a null value are equal.

14.9.8.11 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.

14.9.8.12 GetDaysBetween

This method subtracts an `OracleTimeStampLTZ` value from the current instance and returns an `OracleIntervalDS` that represents the difference.

Declaration

```
// C#
public OracleIntervalDS GetDaysBetween(OracleTimeStampLTZ value1);
```

Parameters

- *`value1`*

  The `OracleTimeStampLTZ` value being subtracted.

Return Value

An `OracleIntervalDS` that represents the interval between two `OracleTimeStampLTZ` values.

Remarks

If either the current instance or the parameter has a null value, the returned `OracleIntervalDS` has a null value.

14.9.8.13 GetYearsBetween

This method subtracts an `OracleTimeStampLTZ` value from the current instance and returns an `OracleIntervalYM` that represents the time interval.

Declaration

```
// C#
public OracleIntervalYM GetYearsBetween(OracleTimeStampLTZ value1);
```

Parameters

- *`value1`*

  The `OracleTimeStampLTZ` value being subtracted.

Return Value

An `OracleIntervalYM` that represents the interval between two `OracleTimeStampLTZ` values.

Remarks

If either the current instance or the parameter has a null value, the returned `OracleIntervalYM` has a null value.

14.9.8.14 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.

14.9.8.15 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.

14.9.8.16 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.

14.9.8.17 ToString

Overrides `Object`

This method converts the current `OracleTimeStampLTZ` structure to a string.

Declaration

```
// C#
public override string ToString();
```

Return Value

A `string` that represents the same date and time as the current `OracleTimeStampLTZ` structure.

Remarks

The returned value is a string representation of the `OracleTimeStampLTZ` in the format specified by the `OracleGlobalization.TimeStampFormat` property of the thread.

The names and abbreviations used for months and days are in the language specified by the `DateLanguage` and `Calendar` properties of the thread's `OracleGlobalization` object. If any of the thread's globalization properties are set to null or an empty string, the client computer's settings are used.

Example

```
// C#

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());
  }
}
```

14.9.8.18 ToUniversalTime

This method converts the current local time to Coordinated Universal Time (UTC).

Declaration

```
// C#
public OracleTimeStampTZ ToUniversalTime();
```

Return Value

An `OracleTimeStampTZ` structure.

Remarks

If the current instance has a null value, the value of the returned `OracleTimeStampTZ` structure has a null value.

14.10 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

```
// C#
public struct OracleTimeStampTZ : IComparable, INullable, IXmlSerializable
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver
Assembly	`Oracle.DataAccess.dll`	`Oracle.ManagedDataAccess.dll`
Namespace	`Oracle.DataAccess.Types`	`Oracle.ManagedDataAccess.Types`
.NET Framework	3.5, 4.5, 4.6	4.5, 4.6

Thread Safety

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

Example

```
// C#
```

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

class OracleTimeStampTZSample
{
  static void Main()
  {
    // Set the nls parameters for the current thread
    OracleGlobalization info = OracleGlobalization.GetClientInfo();
    info.TimeZone = "US/Eastern";
    info.TimeStampFormat = "DD-MON-YYYY HH:MI:SS.FF AM";
    info.TimeStampTZFormat = "DD-MON-YYYY HH:MI:SS.FF AM TZR";
    OracleGlobalization.SetThreadInfo(info);

    // Create an OracleTimeStampTZ in US/Pacific time zone
    OracleTimeStampTZ tstz1=new OracleTimeStampTZ("11-NOV-1999 "+
      "11:02:33.444 AM US/Pacific");

    // Note that ToOracleTimeStampTZ uses the thread's time zone region,
    // "US/Eastern"
    OracleTimeStamp ts = new OracleTimeStamp("11-NOV-1999 11:02:33.444 AM");
    OracleTimeStampTZ tstz2 = ts.ToOracleTimeStampTZ();

    // Calculate the difference between tstz1 and tstz2
    OracleIntervalDS idsDiff = tstz1.GetDaysBetween(tstz2);

    // Display information
    Console.WriteLine("tstz1.TimeZone = " + tstz1.TimeZone);

    // Prints "US/Pacific"
    Console.WriteLine("tstz2.TimeZone = " + tstz2.TimeZone);

    // Prints "US/Eastern"
    Console.WriteLine("idsDiff.Hours = " + idsDiff.Hours); // Prints 3
    Console.WriteLine("idsDiff.Minutes = " + idsDiff.Minutes); // Prints 0
  }
}
```

See Also:

- "Oracle.DataAccess.Types and Oracle.ManagedDataAccess.Types Namespaces"
- OracleTimeStampTZ Members
- OracleTimeStampTZ Constructors
- OracleTimeStampTZ Static Fields
- OracleTimeStampTZ Static Methods
- OracleTimeStampTZ Static Operators
- OracleTimeStampTZ Static Type Conversions
- OracleTimeStampTZ Properties
- OracleTimeStampTZ Methods

14.10.1 OracleTimeStampTZ Members

`OracleTimeStampTZ` members are listed in the following tables:

OracleTimeStampTZ Constructors

`OracleTimeStampTZ` constructors are listed in Table 14-126

Table 14-126 OracleTimeStampTZ Constructors

Constructor	Description
OracleTimeStampTZ Constructors	Instantiates a new instance of `OracleTimeStampTZ` structure (Overloaded)

OracleTimeStampTZ Static Fields

The `OracleTimeStampTZ` static fields are listed in Table 14-127.

Table 14-127 OracleTimeStampTZ Static Fields

Field	Description
MaxValue	Represents the maximum valid date for an `OracleTimeStampTZ` structure in UTC, which is December 31, 999923:59:59.999999999
MinValue	Represents the minimum valid date for an `OracleTimeStampTZ` structure in UTC, which is January 1, -4712 0:0:0
Null	Represents a null value that can be assigned to an instance of the `OracleTimeStampTZ` structure

OracleTimeStampTZ Static Methods

The `OracleTimeStampTZ` static methods are listed in Table 14-128.

Table 14-128 OracleTimeStampTZ Static Methods

Methods	Description
Equals	Determines if two `OracleTimeStampTZ` values are equal (Overloaded)
GetSysDate	Gets an `OracleTimeStampTZ` structure that represents the current date and time
GreaterThan	Determines if the first of two `OracleTimeStampTZ` values is greater than the second
GreaterThanOrEqual	Determines if the first of two `OracleTimeStampTZ` values is greater than or equal to the second
LessThan	Determines if the first of two `OracleTimeStampTZ` values is less than the second
LessThanOrEqual	Determines if the first of two `OracleTimeStampTZ` values is less than or equal to the second

Table 14-128 (Cont.) OracleTimeStampTZ Static Methods

Methods	Description
NotEquals	Determines if two `OracleTimeStampTZ` values are not equal
Parse	Gets an `OracleTimeStampTZ` structure and sets its value for date and time using the supplied string
SetPrecision	Returns a new instance of an `OracleTimeStampTZ` with the specified fractional second precision

OracleTimeStampTZ Static Operators

The `OracleTimeStampTZ` static operators are listed in Table 14-129.

Table 14-129 OracleTimeStampTZ Static Operators

Operator	Description
operator +	Adds the supplied instance value to the supplied `OracleTimeStampTZ` and returns a new `OracleTimeStampTZ` structure (Overloaded)
operator ==	Determines if two `OracleTimeStampTZ` values are equal
operator >	Determines if the first of two `OracleTimeStampTZ` values is greater than the second
operator >=	Determines if the first of two `OracleTimeStampTZ` values is greater than or equal to the second
operator !=	Determines if two `OracleTimeStampTZ` values are not equal
operator <	Determines if the first of two `OracleTimeStampTZ` values is less than the second
operator <=	Determines if the first of two `OracleTimeStampTZ` values is less than or equal to the second
operator -	Subtracts the supplied instance value from the supplied `OracleTimeStampTZ` and returns a new `OracleTimeStampTZ` structure (Overloaded)

OracleTimeStampTZ Static Type Conversions

The `OracleTimeStampTZ` static type conversions are listed in Table 14-130.

Table 14-130 OracleTimeStampTZ Static Type Conversions

Operator	Description
explicit operator OracleTimeStampTZ	Converts an instance value to an `OracleTimeStampTZ` structure (Overloaded)
implicit operator OracleTimeStampTZ	Converts an instance value to an `OracleTimeStampTZ` structure (Overloaded)

Table 14-130 (Cont.) OracleTimeStampTZ Static Type Conversions

Operator	Description
explicit operator DateTime	Converts an `OracleTimeStampTZ` value to a `DateTime` structure

OracleTimeStampTZ Properties

The `OracleTimeStampTZ` properties are listed in Table 14-131.

Table 14-131 OracleTimeStampTZ Properties

Properties	Description
BinData	Returns an array of bytes that represents an Oracle `TIMESTAMP WITH TIME ZONE` in Oracle internal format
Day	Specifies the day component of an `OracleTimeStampTZ` in the current time zone
IsNull	Indicates whether or not the current instance has a null value
Hour	Specifies the hour component of an `OracleTimeStampTZ` in the current time zone
Millisecond	Specifies the millisecond component of an `OracleTimeStampTZ` in the current time zone
Minute	Specifies the minute component of an `OracleTimeStampTZ` in the current time zone
Month	Specifies the month component of an `OracleTimeStampTZ` in the current time zone
Nanosecond	Specifies the nanosecond component of an `OracleTimeStampTZ` in the current time zone
Second	Specifies the second component of an `OracleTimeStampTZ` in the current time zone
TimeZone	Returns the time zone of the `OracleTimeStampTZ` instance
Value	Returns the date and time that is stored in the `OracleTimeStampTZ` structure in the current time zone
Year	Specifies the year component of an `OracleTimeStampTZ`

OracleTimeStampTZ Methods

The `OracleTimeStampTZ` methods are listed in Table 14-132.

Table 14-132 OracleTimeStampTZ Methods

Methods	Description
AddDays	Adds the supplied number of days to the current instance
AddHours	Adds the supplied number of hours to the current instance
AddMilliseconds	Adds the supplied number of milliseconds to the current instance
AddMinutes	Adds the supplied number of minutes to the current instance
AddMonths	Adds the supplied number of months to the current instance
AddNanoseconds	Adds the supplied number of nanoseconds to the current instance
AddSeconds	Adds the supplied number of seconds to the current instance
AddYears	Adds the supplied number of years to the current instance
CompareTo	Compares the current `OracleTimeStampTZ` instance to an object, and returns an integer that represents their relative values
Equals	Determines whether or not an object has the same date and time as the current `OracleTimeStampTZ` instance
GetDaysBetween	Subtracts an `OracleTimeStampTZ` from the current instance and returns an `OracleIntervalDS` that represents the time interval
GetHashCode	Returns a hash code for the `OracleTimeStampTZ` instance
GetTimeZoneOffset	Gets the time zone information in hours and minutes of the current `OracleTimeStampTZ`
GetYearsBetween	Subtracts an `OracleTimeStampTZ` from the current instance and returns an `OracleIntervalYM` that represents the time interval
GetType	Inherited from `System.Object`
ToLocalTime	Converts the current `OracleTimeStampTZ` instance to local time
ToOracleDate	Converts the current `OracleTimeStampTZ` structure to an `OracleDate` structure
ToOracleTimeStampLTZ	Converts the current `OracleTimeStampTZ` structure to an `OracleTimeStampLTZ` structure
ToOracleTimeStamp	Converts the current `OracleTimeStampTZ` structure to an `OracleTimeStamp` structure
ToString	Converts the current `OracleTimeStampTZ` structure to a string

Table 14-132 (Cont.) OracleTimeStampTZ Methods

Methods	Description
ToUniversalTime	Converts the current datetime to Coordinated Universal Time (UTC)

14.10.2 OracleTimeStampTZ Constructors

The `OracleTimeStampTZ` constructors create new instances of the `OracleTimeStampTZ` structure.

Overload List:

- OracleTimeStampTZ(DateTime)

  This constructor creates a new instance of the `OracleTimeStampTZ` structure and sets its value for date and time using the supplied `DateTime` value.

- OracleTimeStampTZ(DateTime, string)

  This constructor creates a new instance of the `OracleTimeStampTZ` structure and sets its value for date and time using the supplied `DateTime` value and the supplied time zone data.

- OracleTimeStampTZ(string)

  This constructor creates a new instance of the `OracleTimeStampTZ` structure and sets its value for date and time using the supplied string.

- OracleTimeStampTZ(int, int, int)

  This constructor creates a new instance of the `OracleTimeStampTZ` structure and sets its value for date and time using year, month, and day.

- OracleTimeStampTZ(int, int, int, string)

  This constructor creates a new instance of the `OracleTimeStampTZ` structure and sets its value for date and time using year, month, day, and time zone data.

- OracleTimeStampTZ(int, int, int, int, int, int)

  This constructor creates a new instance of the `OracleTimeStampTZ` structure and sets its value for date and time using year, month, day, hour, minute, and second.

- OracleTimeStampTZ(int, int, int, int, int, int, string)

  This constructor creates a new instance of the `OracleTimeStampTZ` structure and sets its value for date and time using year, month, day, hour, minute, second, and time zone data.

- OracleTimeStampTZ(int, int, int, int, int, int, double)

  This constructor creates a new instance of the `OracleTimeStampTZ` structure and sets its value for date and time using year, month, day, hour, minute, second, and millisecond.

- OracleTimeStampTZ(int, int, int, int, int, int, double, string)

This constructor creates a new instance of the `OracleTimeStampTZ` structure and sets its value for date and time using year, month, day, hour, minute, second, millisecond, and time zone data.

- OracleTimeStampTZ(int, int, int, int, int, int, int)

  This constructor creates a new instance of the `OracleTimeStampTZ` structure and sets its value for date and time using year, month, day, hour, minute, second, and nanosecond.

- OracleTimeStampTZ(int, int, int, int, int, int, int, string)

  This constructor creates a new instance of the `OracleTimeStampTZ` structure and sets its value for date and time using year, month, day, hour, minute, second, nanosecond, and time zone data.

- OracleTimeStampTZ(byte [ ])

  This constructor creates a new instance of the `OracleTimeStampTZ` structure and sets its value to the provided byte array, that represents the internal Oracle `TIMESTAMP WITH TIME ZONE` format.

14.10.2.1 OracleTimeStampTZ(DateTime)

This constructor creates a new instance of the `OracleTimeStampTZ` structure and sets its value for date and time using the supplied `DateTime` value.

Declaration

```
// C#
public OracleTimeStampTZ (DateTime dt);
```

Parameters

- *`dt`*

  The supplied `DateTime` value.

Remarks

The time zone is set to the `OracleGlobalization.TimeZone` of the thread.

Exceptions

`ArgumentException` - The *`dt`* parameter cannot be used to construct a valid `OracleTimeStampTZ`.

14.10.2.2 OracleTimeStampTZ(DateTime, string)

This constructor creates a new instance of the `OracleTimeStampTZ` structure with the supplied `DateTime` value and the time zone data.

Declaration

```
// C#
public OracleTimeStampTZ (DateTime value1, string timeZone);
```

Parameters

- *`value1`*

  The supplied `DateTime` value.

- *`timeZone`*

  The time zone data provided.

Exceptions

`ArgumentException` - The argument values of the parameters cannot be used to construct a valid `OracleTimeStampTZ`.

Remarks

`timeZone` can be either an hour offset, for example, 7:00, or a valid time zone region name that is provided in `V$TIMEZONE_NAMES`, such as US/Pacific. Time zone abbreviations are not supported.

If time zone is null, the `OracleGlobalization.TimeZone` of the thread is used.

> **Note:**
>
> PST is a time zone region name as well as a time zone abbreviation; therefore it is accepted by `OracleTimeStampTZ`.

14.10.2.3 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());
  }
}
```

14.10.2.4 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.

14.10.2.5 OracleTimeStampTZ(int, int, int, string)

This constructor creates a new instance of the `OracleTimeStampTZ` structure and sets its value for date and time using year, month, day, and time zone data.

Declaration

```
// C#
public OracleTimeStampTZ(int year, int month, int day,
  string timeZone);
```

Parameters

- *`year`*

  The year provided. Range of *`year`* is (-4712 to 9999).
- *`month`*

  The month provided. Range of *`month`* is (1 to 12).
- *`day`*

  The day provided. Range of *`day`* is (1 to 31).
- *`timeZone`*

  The time zone data provided.

Exceptions

`ArgumentOutOfRangeException` - The argument value for one or more of the parameters is out of the specified range.

`ArgumentException` - The argument values of the parameters cannot be used to construct a valid `OracleTimeStampTZ` (that is, the day is out of range for the month or the time zone is invalid).

Remarks

`timeZone` can be either an hour offset, for example, 7:00, or a valid time zone region name that is provided in `V$TIMEZONE_NAMES`, such as US/Pacific. Time zone abbreviations are not supported.

If time zone is null, the `OracleGlobalization.TimeZone` of the thread is used.

Note:

PST is a time zone region name as well as a time zone abbreviation; therefore it is accepted by `OracleTimeStampTZ`.

14.10.2.6 OracleTimeStampTZ(int, int, int, int, int, int)

This constructor creates a new instance of the `OracleTimeStampTZ` structure and sets its value for date and time using year, month, day, hour, minute, and second.

Declaration

```
// C#
public OracleTimeStampTZ(int year, int month, int day, int hour,
  int minute,  int second);
```

Parameters

- *`year`*

  The year provided. Range of *`year`* is (-4712 to 9999).
- *`month`*

  The month provided. Range of *`month`* is (1 to 12).
- *`day`*

  The day provided. Range of *`day`* is (1 to 31).
- *`hour`*

  The hour provided. Range of *`hour`* is (0 to 23).
- *`minute`*

  The minute provided. Range of *`minute`* is (0 to 59).
- *`second`*

  The second provided. Range of *`second`* is (0 to 59).

Exceptions

`ArgumentOutOfRangeException` - The argument value for one or more of the parameters is out of the specified range.

`ArgumentException` - The argument values of the parameters cannot be used to construct a valid `OracleTimeStampTZ` (that is, the day is out of range for the month).

Remarks

The time zone is set to the `OracleGlobalization.TimeZone` of the thread.

14.10.2.7 OracleTimeStampTZ(int, int, int, int, int, int, string)

This constructor creates a new instance of the `OracleTimeStampTZ` structure and sets its value for date and time using year, month, day, hour, minute, second, and time zone data.

Declaration

```
// C#
public OracleTimeStampTZ (int year, int month, int day, int hour,
  int minute, int second, string timeZone);
```

Parameters

- *`year`*

  The year provided. Range of *`year`* is (-4712 to 9999).
- *`month`*

  The month provided. Range of *`month`* is (1 to 12).
- *`day`*

  The day provided. Range of *`day`* is (1 to 31).
- *`hour`*

  The hour provided. Range of *`hour`* is (0 to 23).
- *`minute`*

  The minute provided. Range of *`minute`* is (0 to 59).
- *`second`*

  The second provided. Range of *`second`* is (0 to 59).
- *`timeZone`*

  The time zone data provided.

Exceptions

`ArgumentOutOfRangeException` - The argument value for one or more of the parameters is out of the specified range.

`ArgumentException` - The argument values of the parameters cannot be used to construct a valid `OracleTimeStampTZ` (that is, the day is out of range of the month or the time zone is invalid).

Remarks

`timeZone` can be either an hour offset, for example, 7:00, or a valid time zone region name that is provided in `V$TIMEZONE_NAMES`, such as US/Pacific. Time zone abbreviations are not supported.

If time zone is null, the `OracleGlobalization.TimeZone` of the thread is used.

Note:

PST is a time zone region name as well as a time zone abbreviation; therefore it is accepted by `OracleTimeStampTZ`.

14.10.2.8 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.

14.10.2.9 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`.

14.10.2.10 OracleTimeStampTZ(int, int, int, int, int, int, int)

This constructor creates a new instance of the `OracleTimeStampTZ` structure and sets its value for date and time using year, month, day, hour, minute, second, and nanosecond.

Declaration

```
// C#
public OracleTimeStampTZ(int year, int month, int day, int hour,
  int minute, int second, int nanosecond);
```

Parameters

- *`year`*

  The year provided. Range of *`year`* is (-4712 to 9999).
- *`month`*

  The month provided. Range of *`month`* is (1 to 12).
- *`day`*

  The day provided. Range of *`day`* is (1 to 31).
- *`hour`*

  The hour provided. Range of *`hour`* is (0 to 23).
- *`minute`*

  The minute provided. Range of *`minute`* is (0 to 59).
- *`second`*

  The second provided. Range of *`second`* is (0 to 59).
- *`nanosecond`*

  The nanosecond provided. Range of *`nanosecond`* is (0 to 999999999).

Exceptions

`ArgumentOutOfRangeException` - The argument value for one or more of the parameters is out of the specified range.

`ArgumentException` - The argument values of the parameters cannot be used to construct a valid `OracleTimeStampTZ` (that is, the day is out of range for the month).

Remarks

The time zone is set to the `OracleGlobalization.TimeZone` of the thread.

14.10.2.11 OracleTimeStampTZ(int, int, int, int, int, int, int, string)

This constructor creates a new instance of the `OracleTimeStampTZ` structure and sets its value for date and time using year, month, day, hour, minute, second, nanosecond, and time zone data.

Declaration

```
// C#
public OracleTimeStampTZ(int year, int month, int day, int hour,
  int minute, int second, int nanosecond, string timeZone);
```

Parameters

- *`year`*

  The year provided. Range of *`year`* is (-4712 to 9999).
- *`month`*

  The month provided. Range of *`month`* is (1 to 12).
- *`day`*

  The day provided. Range of *`day`* is (1 to 31).
- *`hour`*

  The hour provided. Range of *`hour`* is (0 to 23).
- *`minute`*

  The minute provided. Range of *`minute`* is (0 to 59).
- *`second`*

  The second provided. Range of *`second`* is (0 to 59).
- *`nanosecond`*

  The nanosecond provided. Range of *`nanosecond`* is (0 to 999999999).
- *`timeZone`*

  The time zone data provided.

Exceptions

`ArgumentOutOfRangeException` - The argument value for one or more of the parameters is out of the specified range.

`ArgumentException` - The argument values of the parameters cannot be used to construct a valid `OracleTimeStampTZ` (that is, the day is out of range for the month or the time zone is invalid).

Remarks

`timeZone` can be either an hour offset, for example, 7:00, or a valid time zone region name that is provided in `V$TIMEZONE_NAMES`, such as US/Pacific. Time zone abbreviations are not supported.

If time zone is null, the `OracleGlobalization.TimeZone` of the thread is used.

Note:

PST is a time zone region name as well as a time zone abbreviation; therefore it is accepted by `OracleTimeStampTZ`.

14.10.2.12 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.

14.10.3 OracleTimeStampTZ Static Fields

The `OracleTimeStampTZ` static fields are listed in Table 14-133.

Table 14-133 OracleTimeStampTZ Static Fields

Field	Description
MaxValue	Represents the maximum valid date for an `OracleTimeStampTZ` structure in UTC, which is December 31, 999923:59:59.999999999
MinValue	Represents the minimum valid date for an `OracleTimeStampTZ` structure in UTC, which is January 1, -4712 0:0:0
Null	Represents a null value that can be assigned to an instance of the `OracleTimeStampTZ` structure

14.10.3.1 MaxValue

This static field represents the maximum valid datetime time for an `OracleTimeStampTZ` structure in UTC, which is December 31, 999923:59:59.999999999.

Declaration

```
// C#
public static readonly OracleTimeStampTZ MaxValue;
```

14.10.3.2 MinValue

This static field represents the minimum valid datetime for an `OracleTimeStampTZ` structure in UTC, which is January 1, -4712 0:0:0.

Declaration

```
// C#
public static readonly OracleTimeStampTZ MinValue;
```

14.10.3.3 Null

This static field represents a null value that can be assigned to an instance of the `OracleTimeStampTZ` structure.

Declaration

```
// C#
public static readonly OracleTimeStampTZ Null;
```

14.10.4 OracleTimeStampTZ Static Methods

The `OracleTimeStampTZ` static methods are listed in Table 14-134.

Table 14-134 OracleTimeStampTZ Static Methods

Methods	Description
Equals	Determines if two `OracleTimeStampTZ` values are equal (Overloaded)
GetSysDate	Gets an `OracleTimeStampTZ` structure that represents the current date and time
GreaterThan	Determines if the first of two `OracleTimeStampTZ` values is greater than the second
GreaterThanOrEqual	Determines if the first of two `OracleTimeStampTZ` values is greater than or equal to the second
LessThan	Determines if the first of two `OracleTimeStampTZ` values is less than the second
LessThanOrEqual	Determines if the first of two `OracleTimeStampTZ` values is less than or equal to the second
NotEquals	Determines if two `OracleTimeStampTZ` values are not equal
Parse	Gets an `OracleTimeStampTZ` structure and sets its value for date and time using the supplied string

Table 14-134 (Cont.) OracleTimeStampTZ Static Methods

Methods	Description
SetPrecision	Returns a new instance of an `OracleTimeStampTZ` with the specified fractional second precision

14.10.4.1 Equals

This static method determines if two `OracleTimeStampTZ` values are equal.

Declaration

```
// 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.

14.10.4.2 GetSysDate

This static method gets an `OracleTimeStampTZ` structure that represents the current date and time.

Declaration

```
// C#
public static OracleTimeStampTZ GetSysDate();
```

Return Value

An `OracleTimeStampTZ` structure that represents the current date and time.

14.10.4.3 GreaterThan

This static method determines if the first of two `OracleTimeStampTZ` values is greater than the second.

Declaration

```
// C#
public static bool GreaterThan(OracleTimeStampTZ value1,
    OracleTimeStampTZ value2);
```

Parameters

- *`value1`*

  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 `OracleTimeStampTZ`s that contain a null value are equal.

14.10.4.4 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 `OracleTimeStampTZ`s that contain a null value are equal.

14.10.4.5 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 `OracleTimeStampTZ`s that contain a null value are equal.

14.10.4.6 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 `OracleTimeStampTZ`s that contain a null value are equal.

14.10.4.7 NotEquals

This static method determines if two `OracleTimeStampTZ` values are not equal.

Declaration

```
// 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 `OracleTimeStampTZ`s that contain a null value are equal.

14.10.4.8 Parse

This static method returns an `OracleTimeStampTZ` structure and sets its value for date and time using the supplied string.

Declaration

```
// C#
public static OracleTimeStampTZ Parse(string tsStr);
```

Parameters

- *tsStr*

  A string that represents an Oracle TIMESTAMP WITH TIME ZONE.

Return Value

An OracleTimeStampTZ structure.

Exceptions

ArgumentException - The *tsStr* is an invalid string representation of an Oracle TIMESTAMP WITH TIME ZONE or the *tsStr* is not in the timestamp format specified by the OracleGlobalization.TimeStampTZFormat property of the thread, which represents 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

```
// 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());
  }
}
```

14.10.4.9 SetPrecision

This static method returns a new instance of an `OracleTimeStampTZ` with the specified fractional second precision.

Declaration

```
// C#
public static OracleTimeStampTZ SetPrecision(OracleTimeStampTZ value1,
    int fracSecPrecision);
```

Parameters

- *`value1`*

  The provided `OracleTimeStampTZ` object.

- *`fracSecPrecision`*

  The fractional second precision provided. Range of fractional second precision is (0 to 9).

Return Value

An `OracleTimeStampTZ` structure with the specified fractional second precision

Exceptions

`ArgumentOutOfRangeException` - *`fracSecPrecision`* is out of the specified range.

Remarks

The value specified in the supplied *`fracSecPrecision`* is used to perform a rounding off operation on the supplied `OracleTimeStampTZ` value. Depending on this value, `0` or more trailing zeros are displayed in the string returned by `ToString()`.

Example

The `OracleTimeStampTZ` with a value of "`December 31, 9999 23:59:59.99 US/Pacific`" results in the string "`December 31, 9999 23:59:59.99000 US/Pacific`" when `SetPrecision()` is called with the fractional second precision set to `5`.

14.10.5 OracleTimeStampTZ Static Operators

The `OracleTimeStampTZ` static operators are listed in Table 14-135.

Table 14-135 OracleTimeStampTZ Static Operators

Operator	Description
operator +	Adds the supplied instance value to the supplied `OracleTimeStampTZ` and returns a new `OracleTimeStampTZ` structure (Overloaded)
operator ==	Determines if two `OracleTimeStampTZ` values are equal
operator >	Determines if the first of two `OracleTimeStampTZ` values is greater than the second

Table 14-135 (Cont.) OracleTimeStampTZ Static Operators

Operator	Description
operator >=	Determines if the first of two `OracleTimeStampTZ` values is greater than or equal to the second
operator !=	Determines if two `OracleTimeStampTZ` values are not equal
operator <	Determines if the first of two `OracleTimeStampTZ` values is less than the second
operator <=	Determines if the first of two `OracleTimeStampTZ` values is less than or equal to the second
operator -	Subtracts the supplied instance value from the supplied `OracleTimeStampTZ` and returns a new `OracleTimeStampTZ` structure (Overloaded)

14.10.5.1 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.

14.10.5.2 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.

14.10.5.3 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.

14.10.5.4 operator +(OracleTimeStampTZ, TimeSpan)

This static operator adds the supplied `TimeSpan` to the supplied `OracleTimeStampTZ` and returns a new `OracleTimeStampTZ` structure.

Declaration

```
// C#
public static operator +(OracleTimeStampTZ value1, TimeSpan value2);
```

Parameters

- *`value1`*

  An `OracleTimeStampTZ`.
- *`value2`*

  A `TimeSpan`.

Return Value

An `OracleTimeStampTZ`.

Remarks

If the `OracleTimeStampTZ` instance has a null value, the returned `OracleTimeStampTZ` has a null value.

14.10.5.5 operator ==

This static operator determines if two `OracleTimeStampTZ` values are equal.

Declaration

```
// C#
public static bool operator == (OracleTimeStampTZ value1,
     OracleTimeStampTZ value2);
```

Parameters

- *`value1`*

  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.

14.10.5.6 operator >

This static operator determines if the first of two `OracleTimeStampTZ` values is greater than the second.

Declaration

```
// C#
public static bool operator > (OracleTimeStampTZ value1,
    OracleTimeStampTZ value2);
```

Parameters

- *`value1`*

  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.

14.10.5.7 operator >=

This static operator determines if the first of two `OracleTimeStampTZ` values is greater than or equal to the second.

Declaration

```
// C#
public static bool operator >= (OracleTimeStampTZ value1,
    OracleTimeStampTZ value2);
```

Parameters

- *value1*

  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 `OracleTimeStampTZ`s that contain a null value are equal.

14.10.5.8 operator !=

This static operator determines if two `OracleTimeStampTZ` values are not equal.

Declaration

```
// 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.

14.10.5.9 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 `OracleTimeStampTZ`s that contain a null value are equal.

14.10.5.10 operator <=

This static operator determines if the first of two `OracleTimeStampTZ` values is less than or equal to the second.

Declaration

```
// C#
public static bool operator <= (OracleTimeStampTZ value1,
 OracleTimeStampTZ value2);
```

Parameters

- *`value1`*

  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.

14.10.5.11 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.

14.10.5.12 operator - (OracleTimeStampTZ, OracleIntervalDS)

This static operator subtracts the supplied `OracleIntervalDS` value, from the supplied `OracleTimeStampTZ` value, and return a new `OracleTimeStampTZ` structure.

Declaration

```
// C#
public static operator - (OracleTimeStampTZ value1,
  OracleIntervalDS value2);
```

Parameters

- *`value1`*

  An `OracleTimeStampTZ`.
- *`value2`*

  An `OracleIntervalDS`.

Return Value

An `OracleTimeStampTZ` structure.

Remarks

If either parameter has a null value, the returned `OracleTimeStampTZ` has a null value.

14.10.5.13 operator - (OracleTimeStampTZ, OracleIntervalYM)

This static operator subtracts the supplied `OracleIntervalYM` value, from the supplied `OracleTimeStampTZ` value, and returns a new `OracleTimeStampTZ` structure.

Declaration

```
// C#
public static operator - (OracleTimeStampTZ value1,
  OracleIntervalYM value2);
```

Parameters

- *`value1`*

  An `OracleTimeStampTZ`.
- *`value2`*

  An `OracleIntervalYM`.

Return Value

An `OracleTimeStampTZ` structure.

Remarks

If either parameter has a null value, the returned `OracleTimeStampTZ` has a null value.

14.10.5.14 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.

14.10.6 OracleTimeStampTZ Static Type Conversions

The `OracleTimeStampTZ` static type conversions are listed in Table 14-136.

Table 14-136 OracleTimeStampTZ Static Type Conversions

Operator	Description
explicit operator OracleTimeStampTZ	Converts an instance value to an `OracleTimeStampTZ` structure (Overloaded)
implicit operator OracleTimeStampTZ	Converts an instance value to an `OracleTimeStampTZ` structure (Overloaded)
explicit operator DateTime	Converts an `OracleTimeStampTZ` value to a `DateTime` structure in the current time zone

14.10.6.1 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.

14.10.6.2 explicit operator OracleTimeStampTZ(OracleTimeStamp)

This static type conversion operator converts an `OracleTimeStamp` value to an `OracleTimeStampTZ` structure.

Declaration

```
// C#
public static explicit operator OracleTimeStampTZ(OracleTimeStamp value1);
```

Parameters

- *`value1`*

  An `OracleTimeStamp`.

Return Value

The returned `OracleTimeStampTZ` contains the date and time from the `OracleTimeStamp` and the time zone from the `OracleGlobalization.TimeZone` of the thread.

Remarks

The `OracleGlobalization.TimeZone` of the thread is used to convert from an `OracleTimeStamp` structure to an `OracleTimeStampTZ` structure.

If the `OracleTimeStamp` structure has a null value, the returned `OracleTimeStampTZ` structure also has a null value.

14.10.6.3 explicit operator OracleTimeStampTZ(OracleTimeStampLTZ)

This static type conversion operator converts an `OracleTimeStampLTZ` value to an `OracleTimeStampTZ` structure.

Declaration

```
// C#
public static explicit operator OracleTimeStampTZ(OracleTimeStampLTZ value1);
```

Parameters

- *`value1`*

  An `OracleTimeStampLTZ`.

Return Value

The returned `OracleTimeStampTZ` contains the date and time from the `OracleTimeStampLTZ` and the time zone from the `OracleGlobalization.TimeZone` of the thread.

Remarks

If the `OracleTimeStampLTZ` structure has a null value, the returned `OracleTimeStampTZ` structure also has a null value.

14.10.6.4 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());
  }
}
```

14.10.6.5 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.

14.10.6.6 implicit operator OracleTimeStampTZ(OracleDate)

This static type conversion operator converts an `OracleDate` value to an `OracleTimeStampTZ` structure.

Declaration

```
// C#
public static implicit operator OracleTimeStampTZ(OracleDate value1);
```

Parameters

- *`value1`*

  An `OracleDate`.

Return Value

The returned `OracleTimeStampTZ` contains the date and time from the `OracleDate` and the time zone from the `OracleGlobalization.TimeZone` of the thread.

Remarks

The `OracleGlobalization.TimeZone` of the thread is used to convert from an `OracleDate` to an `OracleTimeStampTZ` structure. If the `OracleDate` structure has a null value, the returned `OracleTimeStampTZ` structure also has a null value.

14.10.6.7 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.

14.10.6.8 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

14.10.7 OracleTimeStampTZ Properties

The `OracleTimeStampTZ` properties are listed in Table 14-137.

Table 14-137 OracleTimeStampTZ Properties

Properties	Description
BinData	Returns an array of bytes that represents an Oracle `TIMESTAMP WITH TIME ZONE` in Oracle internal format
Day	Specifies the day component of an `OracleTimeStampTZ` in the current time zone
IsNull	Indicates whether or not the current instance has a null value
Hour	Specifies the hour component of an `OracleTimeStampTZ` in the current time zone
Millisecond	Specifies the millisecond component of an `OracleTimeStampTZ` in the current time zone
Minute	Specifies the minute component of an `OracleTimeStampTZ` in the current time zone
Month	Specifies the month component of an `OracleTimeStampTZ` in the current time zone
Nanosecond	Specifies the nanosecond component of an `OracleTimeStampTZ` in the current time zone
Second	Specifies the second component of an `OracleTimeStampTZ` in the current time zone
TimeZone	Returns the time zone of the `OracleTimeStampTZ` instance
Value	Returns the date and time that is stored in the `OracleTimeStampTZ` structure in the current time zone
Year	Specifies the year component of an `OracleTimeStampTZ`

14.10.7.1 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.

14.10.7.2 Day

This property specifies the day component of an `OracleTimeStampTZ` in the current time zone.

Declaration

```
// C#
public int Day{get;}
```

Property Value

A number that represents the day. Range of `Day` is (1 to 31).

Exceptions

`OracleNullValueException` - The current instance has a null value.

14.10.7.3 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`.

14.10.7.4 Hour

This property specifies the hour component of an `OracleTimeStampTZ` in the current time zone.

Declaration

```
// C#
public int Hour{get;}
```

Property Value

A number that represents the hour. Range of `Hour` is (0 to 23).

Exceptions

`OracleNullValueException` - The current instance has a null value.

14.10.7.5 Millisecond

This property gets the millisecond component of an `OracleTimeStampTZ` in the current time zone.

Declaration

```
// C#
public double Millisecond{get;}
```

Property Value

A number that represents a millisecond. Range of `Millisecond` is (0 to 999.999999)

Exceptions

`OracleNullValueException` - The current instance has a null value.

14.10.7.6 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.

14.10.7.7 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.

14.10.7.8 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.

14.10.7.9 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.

14.10.7.10 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

14.10.7.11 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.

14.10.7.12 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.

14.10.8 OracleTimeStampTZ Methods

The `OracleTimeStampTZ` methods are listed in Table 14-138.

Table 14-138 OracleTimeStampTZ Methods

Methods	Description
AddDays	Adds the supplied number of days to the current instance
AddHours	Adds the supplied number of hours to the current instance
AddMilliseconds	Adds the supplied number of milliseconds to the current instance
AddMinutes	Adds the supplied number of minutes to the current instance
AddMonths	Adds the supplied number of months to the current instance
AddNanoseconds	Adds the supplied number of nanoseconds to the current instance

Table 14-138 (Cont.) OracleTimeStampTZ Methods

Methods	Description
AddSeconds	Adds the supplied number of seconds to the current instance
AddYears	Adds the supplied number of years to the current instance
CompareTo	Compares the current `OracleTimeStampTZ` instance to an object, and returns an integer that represents their relative values
Equals	Determines whether or not an object has the same date and time as the current `OracleTimeStampTZ` instance (Overloaded)
GetDaysBetween	Subtracts an `OracleTimeStampTZ` from the current instance and returns an `OracleIntervalDS` that represents the time interval
GetHashCode	Returns a hash code for the `OracleTimeStampTZ` instance
GetTimeZoneOffset	Gets the time zone information in hours and minutes of the current `OracleTimeStampTZ`
GetYearsBetween	Subtracts an `OracleTimeStampTZ` from the current instance and returns an `OracleIntervalYM` that represents the time interval
GetType	Inherited from `System.Object`
ToLocalTime	Converts the current `OracleTimeStampTZ` instance to local time
ToOracleDate	Converts the current `OracleTimeStampTZ` structure to an `OracleDate` structure
ToOracleTimeStampLTZ	Converts the current `OracleTimeStampTZ` structure to an `OracleTimeStampLTZ` structure
ToOracleTimeStamp	Converts the current `OracleTimeStampTZ` structure to an `OracleTimeStamp` structure
ToString	Converts the current `OracleTimeStampTZ` structure to a string
ToUniversalTime	Converts the current datetime to Coordinated Universal Time (UTC)

14.10.8.1 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.

14.10.8.2 AddHours

This method adds the supplied number of hours to the current instance.

Declaration

```
// C#
public OracleTimeStampTZ AddHours(double hours);
```

Parameters

- *`hours`*

  The supplied number of hours. Range is (-24,000,000,000 < *`hours`* < 24,000,000,000).

Return Value

An `OracleTimeStampTZ`.

Exceptions

`OracleNullValueException` - The current instance has a null value.

`ArgumentOutofRangeException` - The argument value is out of the specified range.

14.10.8.3 AddMilliseconds

This method adds the supplied number of milliseconds to the current instance.

Declaration

```
// C#
public OracleTimeStampTZ AddMilliseconds(double milliseconds);
```

Parameters

- *`milliseconds`*

  The supplied number of milliseconds. Range is (-8.64 * 1016< `milliseconds` < 8.64 * 1016).

Return Value

An `OracleTimeStampTZ`.

Exceptions

`OracleNullValueException` - The current instance has a null value.

`ArgumentOutofRangeException` - The argument value is out of the specified range.

14.10.8.4 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.

14.10.8.5 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.

14.10.8.6 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.

14.10.8.7 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.

14.10.8.8 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.

14.10.8.9 CompareTo

This method compares the current `OracleTimeStampTZ` instance to an object, and returns an integer that represents their relative values.

Declaration

```
// C#
public int CompareTo(object obj);
```

Parameters

- *`obj`*

  The object being compared to the current `OracleTimeStampTZ` instance.

Return Value

The method returns a number that is:

Less than zero: if the current `OracleTimeStampTZ` instance value is less than that of *`obj`*.

Zero: if the current `OracleTimeStampTZ` instance and *`obj`* values are equal.

Greater than zero: if the current `OracleTimeStampTZ` instance value is greater than that of *`obj`*.

Implements

`IComparable`

Exceptions

`ArgumentException` - The *`obj`* is not of type `OracleTimeStampTZ`.

Remarks

The following rules apply to the behavior of this method.

- The comparison must be between `OracleTimeStampTZ`s. For example, comparing an `OracleTimeStampTZ` instance with an `OracleBinary` instance is not allowed. When an `OracleTimeStampTZ` is compared with a different type, an `ArgumentException` is thrown.
- Any `OracleTimeStampTZ` that has a value is greater than an `OracleTimeStampTZ` that has a null value.
- Two `OracleTimeStampTZ`s that contain a null value are equal.

14.10.8.10 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 `OracleTimeStampTZ`s that contain a null value are equal.

14.10.8.11 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.

14.10.8.12 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.

14.10.8.13 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.

14.10.8.14 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.

14.10.8.15 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.

14.10.8.16 ToOracleDate

This method converts the current `OracleTimeStampTZ` structure to an `OracleDate` structure.

Declaration

```
// C#
public OracleDate ToOracleDate();
```

Return Value

The returned `OracleDate` contains the date and time in the current instance, but the time zone information in the current instance is truncated

Remarks

The precision of the `OracleTimeStampTZ` value can be lost during the conversion, and the time zone information in the current instance is truncated.

If the current instance has a null value, the value of the returned `OracleDate` structure has a null value.

14.10.8.17 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.

14.10.8.18 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.

14.10.8.19 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);
  }
}
```

14.10.8.20 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.

14.11 INullable Interface

The `INullable` interface is used to determine whether or not an ODP.NET type has a NULL value.

Declaration

```
// C#
public interface INullable
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver
Assembly	`Oracle.DataAccess.dll`	`Oracle.ManagedDataAccess.dll`
Namespace	`Oracle.DataAccess.Types`	`Oracle.ManagedDataAccess.Types`
.NET Framework	3.5, 4.5, 4.6	4.5, 4.6

Thread Safety

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

14.11.1 INullable Interface Members

INullable members are listed in the following tables.

INullable Interface Properties

`INullable` interface properties are listed in Table 14-139.

Table 14-139 INullable Interface Properties

Public Property	Description
IsNull	Indicates whether or not the ODP.NET type has a `NULL` value

14.11.2 INullable Interface Properties

`INullable` interface properties are listed in Table 14-139.

Table 14-140 INullable Interface Properties

Public Property	Description
IsNull	Indicates whether or not the ODP.NET type has a `NULL` value

14.11.2.1 IsNull

This property indicates whether or not the ODP.NET type has a `NULL` value.

Declaration

```
// C#
bool IsNull {get;}
```

Property Value

Returns true if the ODP.NET type has a `NULL value`; otherwise, returns false.

15 Oracle Data Provider for .NET Types Exceptions

This section covers the ODP.NET Types exceptions.

This chapter contains these topics:

15.1 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
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver
Assembly	`Oracle.DataAccess.dll`	`Oracle.ManagedDataAccess.dll`
Namespace	`Oracle.DataAccess.Types`	`Oracle.ManagedDataAccess.Types`
.NET Framework	3.5, 4.5, 4.6	4.5, 4.6

Thread Safety

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

15.1.1 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

Constructor	Description
OracleTypeException Constructors	Creates a new instance of the `OracleTypeException` class (Overloaded)

OracleTypeException Static Methods

The `OracleTypeException` static methods are listed in Table 15-2.

Table 15-2 OracleTypeException Static Methods

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

OracleTypeException Properties

The `OracleTypeException` properties are listed in Table 15-3.

Table 15-3 OracleTypeException Properties

Properties	Description
`HelpLink`	Inherited from `System.SystemException.Exception`
`InnerException`	Inherited from `System.SystemException.Exception`
Message	Specifies the error messages that occur in the exception
Number	Specifies the error number that occurs in the exception
Source	Specifies the name of the data provider that generates the error
`StackTrace`	Inherited from `System.SystemException.Exception`
`TargetSite`	Inherited from `System.SystemException.Exception`

OracleTypeException Methods

The `OracleTypeException` methods are listed in Table 15-4.

Table 15-4 OracleTypeException Methods

Methods	Description
`Equals`	Inherited from `System.Object` (Overloaded)
`GetBaseException`	Inherited from `System.SystemException.Exception`
`GetHashCode`	Inherited from `System.Object`

Table 15-4 (Cont.) OracleTypeException Methods

Methods	Description
`GetObjectData`	Inherited from `System.SystemException.Exception`
`GetType`	Inherited from `System.Object`
ToString	Returns the fully qualified name of this exception

15.1.2 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`.

15.1.2.1 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.

15.1.2.2 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.

- *sc*

  The specified streaming context.

15.1.3 OracleTypeException Static Methods

The `OracleTypeException` static methods are listed in Table 15-5.

Table 15-5 OracleTypeException Static Methods

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

15.1.4 OracleTypeException Properties

The `OracleTypeException` properties are listed in Table 15-6.

Table 15-6 OracleTypeException Properties

Properties	Description
`HelpLink`	Inherited from `System.SystemException.Exception`
`InnerException`	Inherited from `System.SystemException.Exception`
Message	Specifies the error messages that occur in the exception
Number	Specifies the error number that occurs in the exception
Source	Specifies the name of the data provider that generates the error
`StackTrace`	Inherited from `System.SystemException.Exception`
`TargetSite`	Inherited from `System.SystemException.Exception`

15.1.4.1 Message

Overrides `Exception`

This property specifies the error messages that occur in the exception.

Declaration

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

Property Value

An error message.

15.1.4.2 Number

Overrides `Exception`

This property specifies the error number that occurs in the exception

Declaration

```
// C#
public override int Number {get;}
```

Property Value

An error number

15.1.4.3 Source

Overrides `Exception`

This property specifies the name of the data provider that generates the error.

Declaration

```
// C#
public override string Source {get;}
```

Property Value

Oracle Data Provider for .NET.

15.1.5 OracleTypeException Methods

The `OracleTypeException` methods are listed in Table 15-7.

Table 15-7 OracleTypeException Methods

Methods	Description
`Equals`	Inherited from `System.Object` (Overloaded)
`GetBaseException`	Inherited from `System.SystemException.Exception`
`GetHashCode`	Inherited from `System.Object`
`GetObjectData`	Inherited from `System.SystemException.Exception`
`GetType`	Inherited from `System.Object`
ToString	Returns the fully qualified name of this exception

15.1.5.1 ToString

Overrides `Exception`

This method returns the fully qualified name of this exception, the error message in the Message property, the `InnerException.ToString()` message, and the stack trace.

Declaration

```
// C#
public override string ToString();
```

Return Value

The fully qualified name of this exception.

15.2 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
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver
Assembly	`Oracle.DataAccess.dll`	`Oracle.ManagedDataAccess.dll`
Namespace	`Oracle.DataAccess.Types`	`Oracle.ManagedDataAccess.Types`
.NET Framework	3.5, 4.5, 4.6	4.5, 4.6

Thread Safety

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

15.2.1 OracleNullValueException Members

`OracleNullValueException` members are listed in the following tables.

OracleNullValueException Constructors

The `OracleNullValueException` constructors are listed in Table 15-8.

Table 15-8 OracleNullValueException Constructors

Constructor	Description
OracleNullValueException Constructors	Creates a new instance of the `OracleNullValueException` class (Overloaded)

OracleNullValueException Static Methods

The `OracleNullValueException` static methods are listed in Table 15-9.

Table 15-9 OracleNullValueException Static Methods

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

OracleNullValueException Properties

The `OracleNullValueException` properties are listed in Table 15-10.

Table 15-10 OracleNullValueException Properties

Properties	Description
`HelpLink`	Inherited from `System.SystemException.Exception`
`InnerException`	Inherited from `System.SystemException.Exception`
`Message`	Inherited from `OracleTypeException`
`Source`	Inherited from `OracleTypeException`
`StackTrace`	Inherited from `System.SystemException.Exception`
`TargetSite`	Inherited from `System.SystemException.Exception`

OracleNullValueException Methods

The `OracleNullValueException` methods are listed in Table 15-11.

Table 15-11 OracleNullValueException Methods

Methods	Description
`Equals`	Inherited from `System.Object` (Overloaded)
`GetBaseException`	Inherited from `System.SystemException.Exception`
`GetHashCode`	Inherited from `System.Object`
`GetObjectData`	Inherited from `System.SystemException.Exception`
`GetType`	Inherited from `System.Object`
`ToString`	Inherited from `OracleTypeException`

15.2.2 OracleNullValueException Constructors

The `OracleNullValueException` constructors create new instances of the `OracleNullValueException` class.

Overload List:

- OracleNullValueException()

  This constructor creates a new instance of the `OracleNullValueException` class with its default properties.

- OracleNullValueException(string)

  This constructor creates a new instance of the `OracleNullValueException` class with the specified error message, `errMessage`.

15.2.2.1 OracleNullValueException()

This constructor creates a new instance of the `OracleNullValueException` class with its default properties.

Declaration

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

15.2.2.2 OracleNullValueException(string)

This constructor creates a new instance of the `OracleNullValueException` class with the specified error message, `errMessage`.

Declaration

```
// C#
public OracleNullValueException (string errMessage);
```

Parameters

- *`errMessage`*

  The specified error message.

15.2.3 OracleNullValueException Static Methods

The `OracleNullValueException` static methods are listed in Table 15-12.

Table 15-12 OracleNullValueException Static Methods

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

15.2.4 OracleNullValueException Properties

The `OracleNullValueException` properties are listed in Table 15-13.

Table 15-13 OracleNullValueException Properties

Properties	Description
`HelpLink`	Inherited from `System.SystemException.Exception`
`InnerException`	Inherited from `System.SystemException.Exception`
`Message`	Inherited from `OracleTypeException`
`Source`	Inherited from `OracleTypeException`
`StackTrace`	Inherited from `System.SystemException.Exception`
`TargetSite`	Inherited from `System.SystemException.Exception`

15.2.5 OracleNullValueException Methods

The `OracleNullValueException` methods are listed in Table 15-14.

Table 15-14 OracleNullValueException Methods

Methods	Description
`Equals`	Inherited from `System.Object` (Overloaded)
`GetBaseException`	Inherited from `System.SystemException.Exception`
`GetHashCode`	Inherited from `System.Object`
`GetObjectData`	Inherited from `System.SystemException.Exception`
`GetType`	Inherited from `System.Object`
`ToString`	Inherited from `OracleTypeException`

15.3 OracleTruncateException Class

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
```

Requirements

Provider	ODP.NET, Unmanaged Driver	ODP.NET, Managed Driver
Assembly	`Oracle.DataAccess.dll`	`Oracle.ManagedDataAccess.dll`
Namespace	`Oracle.DataAccess.Types`	`Oracle.ManagedDataAccess.Types`
.NET Framework	3.5, 4.5, 4.6	4.5, 4.6

Thread Safety

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

15.3.1 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

Constructor	Description
OracleTruncateException Constructors	Creates a new instance of the `OracleTruncateException` class (Overloaded)

OracleTruncateException Static Methods

The `OracleTruncateException` static methods are listed in Table 15-16.

Table 15-16 OracleTruncateException Static Methods

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

OracleTruncateException Properties

The `OracleTruncateException` properties are listed in Table 15-17.

Table 15-17 OracleTruncateException Properties

Properties	Description
`HelpLink`	Inherited from `System.SystemException.Exception`
`InnerException`	Inherited from `System.SystemException.Exception`

Table 15-17 (Cont.) OracleTruncateException Properties

Properties	Description
`Message`	Inherited from `OracleTypeException`
`Source`	Inherited from `OracleTypeException`
`StackTrace`	Inherited from `System.SystemException.Exception`
`TargetSite`	Inherited from `System.SystemException.Exception`

OracleTruncateException Methods

The `OracleTruncateException` methods are listed in Table 15-18.

Table 15-18 OracleTruncateException Methods

Methods	Description
`Equals`	Inherited from `System.Object` (Overloaded)
`GetBaseException`	Inherited from `System.SystemException.Exception`
`GetHashCode`	Inherited from `System.Object`
`GetObjectData`	Inherited from `System.SystemException.Exception`
`GetType`	Inherited from `System.Object`
`ToString`	Inherited from `OracleTypeException`

15.3.2 OracleTruncateException Constructors

The `OracleTruncateException` constructors create new instances of the `OracleTruncateException` class

Overload List:

- OracleTruncateException()

  This constructor creates a new instance of the `OracleTruncateException` class with its default properties.

- OracleTruncateException(string)

  This constructor creates a new instance of the `OracleTruncateException` class with the specified error message, `errMessage`.

15.3.2.1 OracleTruncateException()

This constructor creates a new instance of the `OracleTruncateException` class with its default properties.

Declaration

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

15.3.2.2 OracleTruncateException(string)

This constructor creates a new instance of the `OracleTruncateException` class with the specified error message, `errMessage`.

Declaration

```
// C#
public OracleTruncateException (string errMessage);
```

Parameters

- *`errMessage`*

  The specified error message.

15.3.3 OracleTruncateException Static Methods

The `OracleTruncateException` static methods are listed in Table 15-19.

Table 15-19 OracleTruncateException Static Methods

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

15.3.4 OracleTruncateException Properties

The `OracleTruncateException` properties are listed in Table 15-20.

Table 15-20 OracleTruncateException Properties

Properties	Description
`HelpLink`	Inherited from `System.SystemException.Exception`
`InnerException`	Inherited from `System.SystemException.Exception`
`Message`	Inherited from `OracleTypeException`
`Source`	Inherited from `OracleTypeException`
`StackTrace`	Inherited from `System.SystemException.Exception`
`TargetSite`	Inherited from `System.SystemException.Exception`

15.3.5 OracleTruncateException Methods

The `OracleTruncateException` methods are listed in Table 15-21.

Table 15-21 OracleTruncateException Methods

Methods	Description
`Equals`	Inherited from `System.Object` (Overloaded)
`GetBaseException`	Inherited from `System.SystemException.Exception`
`GetHashCode`	Inherited from `System.Object`
`GetObjectData`	Inherited from `System.SystemException.Exception`
`GetType`	Inherited from `System.Object`
`ToString`	Inherited from `OracleTypeException`

16 Oracle Data Provider for .NET UDT-Related Classes

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.

> **See Also:**
>
> "Oracle User-Defined Types (UDTs) and .NET Custom Types"

- OracleCustomTypeMappingAttribute Class
- OracleObjectMappingAttribute Class
- OracleArrayMappingAttribute Class
- IOracleCustomType Interface
- IOracleCustomTypeFactory Interface
- IOracleArrayTypeFactory Interface
- OracleUdt Class
- OracleRef Class
- OracleUdtFetchOption Enumeration
- OracleUdtStatus Enumeration

16.1 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
```

Requirements

Provider	ODP.NET, Unmanaged Driver
Assembly	`Oracle.DataAccess.dll`
Namespace	`Oracle.DataAccess.Types`
.NET Framework	3.5, 4.5, 4.6

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 `NUMBER`s.

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.

16.1.1 OracleCustomTypeMappingAttribute Members

`OracleCustomTypeMappingAttribute` members are listed in the following tables.

OracleCustomTypeMappingAttribute Constructors

`OracleCustomTypeMappingAttribute` constructors are listed in Table 16-1.

Table 16-1 OracleCustomTypeMappingAttribute Constructors

Constructor	Description
OracleCustomTypeMappingAttribute Constructors	Instantiates a new instance of `OracleCustomTypeMappingAttribute` class

OracleCustomTypeMappingAttribute Static Methods

`OracleCustomTypeMappingAttribute` static methods are listed in Table 16-2.

Table 16-2 OracleCustomTypeMappingAttribute Static Methods

Method	Description
`Equals`	Inherited from `System.Attribute`
`GetCustomAttribute`	Inherited from `System.Attribute`
`GetCustomAttributes`	Inherited from `System.Attribute`
`IsDefined`	Inherited from `System.Attribute`
`ReferenceEquals`	Inherited from `System.Attribute`

OracleCustomTypeMappingAttribute Properties

`OracleCustomTypeMappingAttribute` properties are listed in Table 16-3.

Table 16-3 OracleCustomTypeMappingAttribute Properties

Property	Description
UdtTypeName	Specifies the Oracle user-defined type name that the custom class maps to
`TypeId`	Inherited from `System.Attribute`

OracleCustomTypeMappingAttribute Methods

`OracleCustomTypeMappingAttribute` methods are listed in Table 16-4.

Table 16-4 OracleCustomTypeMappingAttribute Methods

Method	Description
`Equals`	Inherited from `System.Attribute`
`GetHashCode`	Inherited from `System.Attribute`
`GetType`	Inherited from `System.Attribute`
`IsDefaultAttribute`	Inherited from `System.Attribute`
`Match`	Inherited from `System.Attribute`
`ToString`	Inherited from `System.Attribute`

16.1.2 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.

16.1.2.1 OracleCustomTypeMappingAttribute(string)

This constructor creates and initializes an `OracleCustomTypeMappingAttribute` using the specified Oracle user-defined type name.

Declaration

```
// 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`*.

16.1.3 OracleCustomTypeMappingAttribute Static Methods

`OracleCustomTypeMappingAttribute` static methods are listed in Table 16-5.

Table 16-5 OracleCustomTypeMappingAttribute Static Methods

Method	Description
`Equals`	Inherited from `System.Attribute`
`GetCustomAttribute`	Inherited from `System.Attribute`
`GetCustomAttributes`	Inherited from `System.Attribute`
`IsDefined`	Inherited from `System.Attribute`
`ReferenceEquals`	Inherited from `System.Attribute`

16.1.4 OracleCustomTypeMappingAttribute Properties

`OracleCustomTypeMappingAttribute` properties are listed in Table 16-6.

Table 16-6 OracleCustomTypeMappingAttribute Properties

Property	Description
UdtTypeName	Specifies the Oracle user-defined type name that the custom class maps to
`TypeId`	Inherited from `System.Attribute`

16.1.4.1 UdtTypeName

This property specifies the Oracle user-defined type name that the custom class maps to.

Declaration

```
// C#
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`*.

16.1.5 OracleCustomTypeMappingAttribute Methods

`OracleCustomTypeMappingAttribute` methods are listed in Table 16-7.

Table 16-7 OracleCustomTypeMappingAttribute Methods

Method	Description
`Equals`	Inherited from `System.Attribute`
`GetHashCode`	Inherited from `System.Attribute`
`GetType`	Inherited from `System.Attribute`
`IsDefaultAttribute`	Inherited from `System.Attribute`
`Match`	Inherited from `System.Attribute`
`ToString`	Inherited from `System.Attribute`

16.2 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
```

Requirements

Provider	ODP.NET, Unmanaged Driver
Assembly	`Oracle.DataAccess.dll`
Namespace	`Oracle.DataAccess.Types`
.NET Framework	3.5, 4.5, 4.6

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.

16.2.1 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

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

OracleObjectMappingAttribute Static Methods

`OracleObjectMappingAttribute` static methods are listed in Table 16-9.

Table 16-9 OracleObjectMappingAttribute Static Methods

Method	Description
`Equals`	Inherited from `System.Attribute`
`GetCustomAttribute`	Inherited from `System.Attribute`
`GetCustomAttributes`	Inherited from `System.Attribute`
`IsDefined`	Inherited from `System.Attribute`
`ReferenceEquals`	Inherited from `System.Attribute`

OracleObjectMappingAttribute Properties

`OracleObjectMappingAttribute` properties are listed in Table 16-10.

Table 16-10 OracleObjectMappingAttribute Properties

Property	Description
AttributeIndex	Specifies the index of the Oracle Object attribute that must be retrieved
AttributeName	Specifies the name of Oracle Object attribute that must be retrieved
`TypeId`	Inherited from `System.Attribute`

OracleObjectMappingAttribute Methods

`OracleObjectMappingAttribute` methods are listed in Table 16-11.

Table 16-11 OracleObjectMappingAttribute Methods

Method	Description
`Equals`	Inherited from `System.Attribute`
`GetHashCode`	Inherited from `System.Attribute`
`GetType`	Inherited from `System.Attribute`
`IsDefaultAttribute`	Inherited from `System.Attribute`
`Match`	Inherited from `System.Attribute`
`ToString`	Inherited from `System.Attribute`

16.2.2 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.

16.2.2.1 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.

16.2.2.2 OracleObjectMappingAttribute(int)

This constructor creates and initializes an `OracleObjectMappingAttribute` object with the specified Oracle Object attribute index.

Declaration

```
// C#
public OracleObjectMappingAttribute(int attrIndex);
```

Parameters

- *`attrIndex`*

  The zero-based index of the Oracle Object attribute to map to.

16.2.3 OracleObjectMappingAttribute Static Methods

`OracleObjectMappingAttribute` static methods are listed in Table 16-12.

Table 16-12 OracleObjectMappingAttribute Static Method

Method	Description
`Equals`	Inherited from `System.Attribute`
`GetCustomAttribute`	Inherited from `System.Attribute`
`GetCustomAttributes`	Inherited from `System.Attribute`
`IsDefined`	Inherited from `System.Attribute`
`ReferenceEquals`	Inherited from `System.Attribute`

16.2.4 OracleObjectMappingAttribute Properties

`OracleObjectMappingAttribute` properties are listed in Table 16-13.

Table 16-13 OracleObjectMappingAttribute Properties

Property	Description
AttributeIndex	Specifies the index of the Oracle Object attribute that must be retrieved
AttributeName	Specifies the name of the Oracle Object attribute that must be retrieved
`TypeId`	Inherited from `System.Attribute`

16.2.4.1 AttributeIndex

This property specifies the index of the Oracle Object attribute that must be retrieved.

Declaration

```
// 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.

16.2.4.2 AttributeName

This property specifies the name of the Oracle Object attribute that must be retrieved.

Declaration

```
// 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.

16.2.5 OracleObjectMappingAttribute Methods

`OracleObjectMappingAttribute` methods are listed in Table 16-14.

Table 16-14 OracleObjectMappingAttribute Methods

Method	Description
`Equals`	Inherited from `System.Attribute`
`GetHashCode`	Inherited from `System.Attribute`
`GetType`	Inherited from `System.Attribute`
`IsDefaultAttribute`	Inherited from `System.Attribute`
`Match`	Inherited from `System.Attribute`
`ToString`	Inherited from `System.Attribute`

16.3 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
```

Requirements

Provider	ODP.NET, Unmanaged Driver
Assembly	`Oracle.DataAccess.dll`
Namespace	`Oracle.DataAccess.Types`
.NET Framework	3.5, 4.5, 4.6

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.

16.3.1 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

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

OracleArrayMappingAttribute Static Methods

`OracleArrayMappingAttribute` static methods are listed in Table 16-16.

Table 16-16 OracleArrayMappingAttribute Static Methods

Method	Description
`Equals`	Inherited from `System.Attribute`
`GetCustomAttribute`	Inherited from `System.Attribute`
`GetCustomAttributes`	Inherited from `System.Attribute`
`IsDefined`	Inherited from `System.Attribute`
`ReferenceEquals`	Inherited from `System.Attribute`

OracleArrayMappingAttribute Properties

`OracleArrayMappingAttribute` properties are listed in Table 16-17.

Table 16-17 OracleArrayMappingAttribute Properties

Property	Description
`TypeId`	Inherited from `System.Attribute`

OracleArrayMappingAttribute Methods

`OracleArrayMappingAttribute` methods are listed in Table 16-18.

Table 16-18 OracleArrayMappingAttribute Methods

Method	Description
`Equals`	Inherited from `System.Attribute`
`GetHashCode`	Inherited from `System.Attribute`
`GetType`	Inherited from `System.Attribute`
`IsDefaultAttribute`	Inherited from `System.Attribute`

Table 16-18 (Cont.) OracleArrayMappingAttribute Methods

Method	Description
`Match`	Inherited from `System.Attribute`
`ToString`	Inherited from `System.Attribute`

16.3.2 OracleArrayMappingAttribute Constructors

`OracleArrayMappingAttribute` constructors create new instances of the `OracleArrayMappingAttribute` class.

Overload List:

- OracleArrayMappingAttribute()

  This constructor creates and initializes an `OracleArrayMappingAttribute` object.

16.3.2.1 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.

16.3.3 OracleArrayMappingAttribute Static Methods

`OracleArrayMappingAttribute` static methods are listed in Table 16-19.

Table 16-19 OracleArrayMappingAttribute Static Methods

Method	Description
`Equals`	Inherited from `System.Attribute`
`GetCustomAttribute`	Inherited from `System.Attribute`
`GetCustomAttributes`	Inherited from `System.Attribute`
`IsDefined`	Inherited from `System.Attribute`
`ReferenceEquals`	Inherited from `System.Attribute`

16.3.4 OracleArrayMappingAttribute Properties

`OracleArrayMappingAttribute` properties are listed in Table 16-20.

Table 16-20 OracleArrayMappingAttribute Properties

Property	Description
`TypeId`	Inherited from `System.Attribute`

16.3.5 OracleArrayMappingAttribute Methods

`OracleArrayMappingAttribute` methods are listed in Table 16-21.

Table 16-21 OracleArrayMappingAttribute Methods

Method	Description
`Equals`	Inherited from `System.Attribute`
`GetHashCode`	Inherited from `System.Attribute`
`GetType`	Inherited from `System.Attribute`
`IsDefaultAttribute`	Inherited from `System.Attribute`
`Match`	Inherited from `System.Attribute`
`ToString`	Inherited from `System.Attribute`

16.4 IOracleCustomType Interface

`IOracleCustomType` is an interface for converting between a Custom Type and an Oracle Object or Collection Type.

Declaration

```
// C#
public interface IOracleCustomType
```

Requirements

Provider	ODP.NET, Unmanaged Driver
Assembly	`Oracle.DataAccess.dll`
Namespace	`Oracle.DataAccess.Types`
.NET Framework	3.5, 4.5, 4.6

Thread Safety

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

16.4.1 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

Interface Method	Description
FromCustomObject	Returns the values that set the Oracle Object attributes
ToCustomObject	Provides the Oracle Object with the attribute values to set on the custom type

16.4.2 IOracleCustomType Interface Methods

`IOracleCustomType` Interface methods are listed in Table 16-23.

Table 16-23 IOracleCustomType Interface Methods

Interface Method	Description
FromCustomObject	Returns the values that set the Oracle Object attributes
ToCustomObject	Provides the Oracle Object with the attribute values to set on the custom type

16.4.2.1 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.

16.4.2.2 ToCustomObject

This interface initializes a custom object using the specified Oracle UDT.

Declaration

```
// 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.

16.5 IOracleCustomTypeFactory Interface

The `IOracleCustomTypeFactory` interface is used by ODP.NET to create custom objects that represent Oracle Objects or Collections.

Declaration

```
// C#
public interface IOracleCustomTypeFactory
```

Requirements

Provider	ODP.NET, Unmanaged Driver
Assembly	`Oracle.DataAccess.dll`
Namespace	`Oracle.DataAccess.Types`
.NET Framework	3.5, 4.5, 4.6

Thread Safety

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

16.5.1 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

Public Method	Description
CreateObject	Returns a new custom object to represent an Oracle Object or Collection

16.5.2 IOracleCustomTypeFactory Interface Methods

`IOracleCustomTypeFactory` Interface methods are listed in Table 16-25.

Table 16-25 IOracleCustomTypeFactory Interface Methods

Public Method	Description
CreateObject	Returns a new custom object to represent an Oracle Object or Collection

16.5.2.1 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.

16.6 IOracleArrayTypeFactory Interface

The `IOracleArrayTypeFactory` interface is used by ODP.NET to create arrays that represent Oracle Collections.

Declaration

```
// C#
public interface IOracleArrayTypeFactory
```

Requirements

Provider	ODP.NET, Unmanaged Driver
Assembly	`Oracle.DataAccess.dll`
Namespace	`Oracle.DataAccess.Types`
.NET Framework	3.5, 4.5, 4.6

Thread Safety

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

16.6.1 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

Public Method	Description
CreateArray	Returns a new array of the specified length to store Oracle Collection elements
CreateStatusArray	Returns a newly allocated `OracleUdtStatus` array of the specified length that will be used to store the null status of the collection elements

16.6.2 IOracleArrayTypeFactory Interface Methods

`IOracleArrayTypeFactory` Interface methods are listed in Table 16-27.

Table 16-27 IOracleArrayTypeFactory Interface Methods

Public Method	Description
CreateArray	Returns a new array of the specified length to store Oracle Collection elements
CreateStatusArray	Returns a newly allocated `OracleUdtStatus` array of the specified length that will be used to store the null status of the collection elements

16.6.2.1 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.

16.6.2.2 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.

16.7 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
```

Requirements

Provider	ODP.NET, Unmanaged Driver
Assembly	`Oracle.DataAccess.dll`
Namespace	`Oracle.DataAccess.Types`
.NET Framework	3.5, 4.5, 4.6

Thread Safety

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

16.7.1 OracleUdt Members

`OracleUdt` static methods are listed in Table 16-28.

Table 16-28   OracleUdt Static Methods

Static Method	Description
Equals	Inherited from `System.Object`
GetValue	Gets the attributes or elements from the specified Oracle UDT (Overloaded)
IsDBNull	Indicates whether or not the specified attribute being retrieved is `NULL` (Overloaded)
SetValue	Sets the attributes or elements on the specified Oracle UDT (Overloaded)

16.7.2 OracleUDT Static Methods

`OracleUDT` methods are listed in Table 16-29.

Table 16-29   OracleUdt Static Methods

Static Method	Description
Equals	Inherited from `System.Object`
GetValue	Gets the attributes or elements from the specified Oracle UDT (Overloaded)
IsDBNull	Indicates whether or not the specified attribute being retrieved is `NULL` (Overloaded)
SetValue	Sets the attributes or elements on the specified Oracle UDT (Overloaded)

16.7.2.1 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.

16.7.2.2 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.

16.7.2.3 GetValue(OracleConnection, IntPtr, int)

This method gets the attribute or elements from the specified Oracle UDT, using the specified index.

Declaration

```
// 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

`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.

16.7.2.4 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

  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.

16.7.2.5 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.

16.7.2.6 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`.

16.7.2.7 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.

16.7.2.8 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`*

  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

`ArgumentOutOfRangeException` - 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.

16.7.2.9 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.

16.7.2.10 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.

16.7.2.11 SetValue(OracleConnection, IntPtr, int, object)

This method sets the attribute or elements on the specified Oracle UDT, using the specified index and value.

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 `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.

16.7.2.12 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,
   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.

16.7.2.13 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.

16.8 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
```

Requirements

Provider	ODP.NET, Unmanaged Driver
Assembly	`Oracle.DataAccess.dll`
Namespace	`Oracle.DataAccess.Types`
.NET Framework	3.5, 4.5, 4.6

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.

16.8.1 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

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

OracleRef Static Fields

OracleRef static methods are listed in Table 16-31

Table 16-31 OracleRef Static Fields

Static Field	Description
Null	Represents a null value that can be assigned to an `OracleRef` instance

OracleRef Static Methods

`OracleRef` static methods are listed in Table 16-32.

Table 16-32 OracleRef Static Methods

Method	Description
`Equals`	Inherited from `System.Object` (Overloaded)

OracleRef Instance Properties

`OracleRef` instance properties are listed in Table 16-33.

Table 16-33 OracleRef Instance Properties

Property	Description
Connection	References the connection used by the `OracleRef`
HasChanges	References the connection used by the `OracleRef`
IsLocked	Indicates whether or not the `REF` is locked
IsNull	Indicates whether or not the Oracle `REF` is `NULL`
ObjectTableName	Returns the fully qualified object table name that is associated with the `REF`
Value	Returns a .NET representation of this Oracle `REF`

OracleRef Instance Methods

`OracleRef` instance methods are listed in Table 16-34.

Table 16-34 OracleRef Instance Methods

Method	Description
Clone	Clones the `REF`
Delete	Deletes the referenced object from the database
Dispose	Releases resources allocated for the `OracleRef` instance
`Equals`	Inherited from `System.Object`
Flush	Flushes changes made on the `REF` object to the database

Table 16-34 (Cont.) OracleRef Instance Methods

Method	Description
GetCustomObject	Returns the object that the specified REF references as a custom type (Overloaded)
GetCustomObjectForUpdate	Returns the object that the specified REF references as a custom type (Overloaded)
`GetHashCode`	Inherited from `System.Object`
`GetType`	Inherited from `System.Object`
IsEqual	Compares two `OracleREF` objects
Lock	Locks the `REF` in the database
`ToString`	Inherited from `System.Object`
Update	Updates the object referenced by the specified `REF` in the database using the specified custom object

16.8.2 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

16.8.2.1 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.

16.8.2.2 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.

16.8.3 OracleRef Static Fields

`OracleRef` static fields are listed in Table 16-35.

Table 16-35 OracleRef Static Fields

Static Field	Description
Null	Represents a null value that can be assigned to an `OracleRef` instance

16.8.3.1 Null

This static field represents a null value that can be assigned to an `OracleRef` instance.

Declaration

```
// C#
public static readonly OracleRef Null;
```

16.8.4 OracleRef Static Methods

`OracleRef` static methods are listed in Table 16-36.

Table 16-36 OracleRef Static Methods

Method	Description
`Equals`	Inherited from `System.Object` (Overloaded)

16.8.5 OracleRef Instance Properties

`OracleRef` instance properties are listed in Table 16-37.

Table 16-37 OracleRef Instance Properties

Property	Description
Connection	References the connection used by the `OracleRef`
HasChanges	References the connection used by the `OracleRef`
IsLocked	Indicates whether or not the `REF` is locked
IsNull	Indicates whether or not the Oracle `REF` is `NULL`
ObjectTableName	Returns the fully qualified object table name that is associated with the `REF`
Value	Returns a .NET representation of this Oracle `REF`

16.8.5.1 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`.

16.8.5.2 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.

16.8.5.3 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.

16.8.5.4 IsNull

This instance property indicates whether or not the Oracle `REF` is `NULL`.

Declaration

```
// C#
public bool IsNull {get;}
```

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.

16.8.5.5 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`*.

16.8.5.6 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.

16.8.6 Oracle Ref Instance Methods

`OracleRef` instance methods are listed in Table 16-38.

Table 16-38 OracleRef Instance Methods

Method	Description
Clone	Clones the `REF`
Delete	Deletes the referenced object from the database
Dispose	Releases resources allocated for the `OracleRef` instance
`Equals`	Inherited from `System.Object`
Flush	Flushes changes made on the `REF` object to the database
GetCustomObject	Returns the object that the specified REF references as a custom type (Overloaded)
GetCustomObjectForUpdate	Returns the object that the specified REF references as a custom type (Overloaded)
`GetHashCode`	Inherited from `System.Object`
`GetType`	Inherited from `System.Object`
IsEqual	Compares two `OracleREF` objects
Lock	Locks the `REF` in the database
`ToString`	Inherited from `System.Object`
Update	Updates the object referenced by the specified `REF` in the database using the specified custom object

16.8.6.1 Clone

This instance method clones the `REF`.

Declaration

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

Return Value

A clone of the current instance.

Implements

`ICloneable`

Exceptions

`InvalidOperationException` - The associated connection is not open.

16.8.6.2 Delete

This method deletes the referenced object from the database.

Declaration

```
// 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.

16.8.6.3 Dispose

This instance method releases resources allocated for the `OracleRef` instance.

Declaration

```
// 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.

16.8.6.4 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.

16.8.6.5 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.

16.8.6.6 GetCustomObject(OracleUdtFetchOption)

This method returns the object that the specified `REF` references, as a custom type, using the specified fetch option.

Declaration

```
// 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.

16.8.6.7 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

```
// 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.

16.8.6.8 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 and Oracle.ManagedDataAccess.Types Namespaces"
- OracleRef Class
- OracleRef Members

16.8.6.9 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

```
// C#
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.

16.8.6.10 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.

The connection must be opened explicitly by the application. This method does not open the connection implicitly.

16.8.6.11 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`.

16.8.6.12 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.

16.8.6.13 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.

16.9 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

Member Name	Description
`Cache`	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.
`Server`	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.
`TransactionCache`	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.

16.10 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 16-40 OracleUdtStatus Enumeration Values

Member Name	Description
`Null`	Indicates that an object attribute or collection element is `NULL`.
`NotNull`	Indicates that a non-`NULL` value exists for the object attribute or collection element.

17 Oracle Data Provider for .NET Bulk Copy Classes

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:

17.1 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
```

Requirements

Provider	ODP.NET, Unmanaged Driver
Assembly	`Oracle.DataAccess.dll`
Namespace	`Oracle.DataAccess.Client`

Provider	ODP.NET, Unmanaged Driver
.NET Framework	3.5, 4.5, 4.6

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.

Bulk copy of string data to destination number column is currently not supported.

17.1.1 OracleBulkCopy Members

`OracleBulkCopy` members are listed in the following tables.

OracleBulkCopy Constructors

`OracleBulkCopy` constructors are listed in Table 17-1.

Table 17-1 OracleBulkCopy Constructors

Constructor	Description
OracleBulkCopy Constructors	`OracleBulkCopy` constructors create new instances of the `OracleBulkCopy` class

OracleBulkCopy Properties

`OracleBulkCopy` properties are listed in Table 17-2.

Table 17-2 OracleBulkCopy Properties

Property	Description
BatchSize	Specifies the number of rows to be sent as a batch to the database
BulkCopyOptions	Specifies the `OracleBulkCopyOptions` enumeration value that determines the behavior of the bulk copy operation
BulkCopyTimeout	Specifies the number of seconds allowed for the bulk copy operation to complete before it is aborted
ColumnMappings	Specifies the column mappings between the data source and destination table
Connection	Specifies the `OracleConnection` object that the Oracle database uses to perform the bulk copy operation
DestinationPartitionName	Specifies the database partition that the data is loaded into

Table 17-2 (Cont.) OracleBulkCopy Properties

Property	Description
DestinationTableName	Specifies the database table that the data is loaded in
NotifyAfter	Defines the number of rows to be processed before a notification event is generated

OracleBulkCopy Public Methods

`OracleBulkCopy` public methods are listed in Table 17-3.

Table 17-3 OracleBulkCopy Public Methods

Method	Description
Close	Closes the `OracleBulkCopy` instance
Dispose	Releases any resources or memory allocated by the object
WriteToServer	Copies rows to a destination table

OracleBulkCopy Events

`OracleBulkCopy` events are listed in Table 17-4.

Table 17-4 OracleBulkCopy Events

Event	Description
OracleRowsCopied	Triggered every time the number of rows specified by the `OracleBulkCopy.NotifyAfter` property has been processed

17.1.2 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.

17.1.2.1 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

`ArgumentNullException` - 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.

17.1.2.2 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`*.

17.1.2.3 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.

17.1.2.4 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`*.

17.1.3 OracleBulkCopy Properties

`OracleBulkCopy` properties are listed in Table 17-5.

Table 17-5 OracleBulkCopy Properties

Property	Description
BatchSize	Specifies the number of rows to be sent as a batch to the database
BulkCopyOptions	Specifies the `OracleBulkCopyOptions` enumeration value that determines the behavior of the bulk copy operation
BulkCopyTimeout	Specifies the number of seconds allowed for the bulk copy operation to complete before it is aborted
ColumnMappings	Specifies the column mappings between the data source and destination table
Connection	Specifies the `OracleConnection` object that the Oracle database uses to perform the bulk copy operation
DestinationPartitionName	Specifies the database partition that the data is loaded into
DestinationTableName	Specifies the database table that the data is loaded in
NotifyAfter	Defines the number of rows to be processed before a notification event is generated

17.1.3.1 BatchSize

This property specifies the number of rows to be sent as a batch to the database.

Declaration

```
// 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 large batch size reduces database round trips, but it can also consume large amounts of client side memory. Excessive memory consumption slows down overall machine performance and leads to errors if the process runs out of accessible memory. It is recommended that client side memory is not consumed in excess. This can be done by reducing the batch size.

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`.

17.1.3.2 BulkCopyOptions

This property specifies the `OracleBulkCopyOptions` enumeration value that determines the behavior of the bulk copy option.

Declaration

```
// 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.

17.1.3.3 BulkCopyTimeout

This property specifies the number of seconds allowed for the bulk copy operation to complete before it is aborted.

Declaration

```
// 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.

17.1.3.4 ColumnMappings

This property specifies the column mappings between the data source and destination table.

Declaration

```
// 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.

17.1.3.5 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.

17.1.3.6 DestinationPartitionName

This property specifies the database partition that the data is loaded into.

Declaration

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

Property Value

A string value that identifies the destination partition name.

Remarks

If DestinationPartitionName is modified while a WriteToServer operation is running, the change does not affect the current operation. The new DestinationPartitionName value is used the next time a WriteToServer method is called.

17.1.3.7 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.

17.1.3.8 NotifyAfter

This property defines the number of rows to be processed before a notification event is generated.

Declaration

```
// 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.

17.1.4 OracleBulkCopy Public Methods

`OracleBulkCopy` methods are listed in Table 17-6.

Table 17-6 OracleBulkCopy Public Methods

Method	Description
Close	Closes the `OracleBulkCopy` instance
Dispose	Releases any resources or memory allocated by the object
WriteToServer	Copies rows to a destination table

17.1.4.1 Close

This method closes the `OracleBulkCopy` instance.

Declaration

```
// 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.

17.1.4.2 Dispose

This method releases any resources or memory allocated by the object.

Declaration

```
// 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.

17.1.4.3 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.

17.1.4.4 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.

17.1.4.5 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.

17.1.4.6 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.

17.1.4.7 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);
```

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

`ArgumentNullException` - 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.

`DataRowState.Deleted` is not supported and the behavior would be that all the rows except the deleted ones are copied.

17.1.4.8 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

```
// C#
public void WriteToServer(OracleRefCursor refCursor);
```

Parameters

- *`refCursor`*

  An `OracleRefCursor` object containing rows to be copied to the destination table.

Exceptions

`ArgumentNullException` - 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.

17.1.5 OracleBulkCopy Events

`OracleBulkCopy` events are listed in Table 17-7.

Table 17-7 OracleBulkCopy Events

Event	Description
OracleRowsCopied	Triggered every time the number of rows specified by the `OracleBulkCopy.NotifyAfter` property has been processed

17.1.5.1 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.

17.2 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
```

Requirements

Provider	ODP.NET, Unmanaged Driver
Assembly	`Oracle.DataAccess.dll`
Namespace	`Oracle.DataAccess.Client`
.NET Framework	3.5, 4.5, 4.6

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#

17.2.1 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

Constructor	Description
OracleBulkCopyColumnMapping Constructors	Instantiates new instances of the `OracleBulkCopyColumnMapping` class

OracleBulkCopyColumnMapping Methods

The `OracleBulkCopyColumnMapping` method is listed in Table 17-9.

Table 17-9 OracleBulkCopyColumnMapping Method

Constructor	Description
CompareTo	Compares the current instance to the supplied object and returns an integer that represents their relative values

OracleBulkCopyColumnMapping Properties

The `OracleBulkCopyColumnMapping` properties are listed in Table 17-10.

Table 17-10 OracleBulkCopyColumnMapping Properties

Property	Description
DestinationColumn	Specifies the column name of the destination table that is being mapped
DestinationOrdinal	Specifies the column ordinal value of the destination table that is being mapped
SourceColumn	Specifies the column name of the data source that is being mapped
SourceOrdinal	Specifies the column ordinal value of the data source that is being mapped

17.2.2 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.

17.2.2.1 OracleBulkCopyColumnMapping()

This constructor instantiates a new instance of the `OracleBulkCopyColumnMapping` class.

Declaration

```
// 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.

17.2.2.2 OracleBulkCopyColumnMapping(int, int)

This constructor instantiates a new instance of the `OracleBulkCopyColumnMapping` class using the provided source and destination column ordinal positions.

Declaration

```
// 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.

17.2.2.3 OracleBulkCopyColumnMapping(int, string)

This constructor instantiates a new instance of the `OracleBulkCopyColumnMapping` class using the provided source column ordinal and destination column name.

Declaration

```
// 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.

17.2.2.4 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.

17.2.2.5 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.

17.2.3 OracleBulkCopyColumnMapping Methods

The `OracleBulkCopyColumnMapping` method is listed in Table 17-11.

Table 17-11 OracleBulkCopyColumnMapping Method

Property	Description
CompareTo	Compares the current instance to the supplied object and returns an integer that represents their relative values

17.2.3.1 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

Less than zero: if the value of the current instance is less than obj.

Zero: if the value of the current instance is equal to obj.

Greater than zero: if the value of the current instance is greater than obj.

Implements

`IComparable`

17.2.4 OracleBulkCopyColumnMapping Properties

The `OracleBulkCopyColumnMapping` properties are listed in Table 17-12.

Table 17-12 OracleBulkCopyColumnMapping Properties

Property	Description
DestinationColumn	Specifies the column name of the destination table that is being mapped
DestinationOrdinal	Specifies the column ordinal value of the destination table that is being mapped
SourceColumn	Specifies the column name of the data source that is being mapped
SourceOrdinal	Specifies the column ordinal value of the data source that is being mapped

17.2.4.1 DestinationColumn

This property specifies the column name of the destination table that is being mapped.

Declaration

```
// 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.

17.2.4.2 DestinationOrdinal

This property specifies the column ordinal value of the destination table that is being mapped.

Declaration

```
// 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.

17.2.4.3 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.

17.2.4.4 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.

17.3 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
```

Requirements

Provider	ODP.NET, Unmanaged Driver
Assembly	`Oracle.DataAccess.dll`
Namespace	`Oracle.DataAccess.Client`
.NET Framework	3.5, 4.5, 4.6

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#

17.3.1 OracleBulkCopyColumnMappingCollection Members

`OracleBulkCopyColumnMappingCollection` members are listed in the following tables.

OracleBulkCopyColumnMappingCollection Properties

The `OracleBulkCopyColumnMappingCollection` properties are listed in Table 17-13.

Table 17-13 OracleBulkCopyColumnMappingCollection Properties

Property	Description
Item[index]	Gets or sets the `OracleBulkCopyColumnMappingCollection` object at the specified index

OracleBulkCopyColumnMappingCollection Public Methods

The `OracleBulkCopyColumnMappingCollection` public methods are listed in Table 17-14.

Table 17-14 OracleBulkCopyColumnMappingCollection Public Methods

Public Method	Description
Add	Adds objects to the collection
Clear	Clears the contents of the collection

Table 17-14 (Cont.) OracleBulkCopyColumnMappingCollection Public Methods

Public Method	Description
Contains	Returns a value indicating whether or not a specified `OracleBulkCopyColumnMapping` object exists in the collection
CopyTo	Copies the elements of the `OracleBulkCopyColumnMappingCollection` to an array of `OracleBulkCopyColumnMapping` items, starting at a specified index
IndexOf	Returns the index of the specified `OracleBulkCopyColumnMapping` object
Insert	Inserts a new `OracleBulkCopyColumnMapping` object in the collection, at the index specified.
Remove	Removes the specified `OracleBulkCopyColumnMapping` element from the `OracleBulkCopyColumnMappingCollection`.
RemoveAt	Removes the mapping from the collection at the specified index.

17.3.2 OracleBulkCopyColumnMappingCollection Properties

The `OracleBulkCopyColumnMappingCollection` properties are listed in Table 17-15.

Table 17-15 OracleBulkCopyColumnMappingCollection Properties

Property	Description
Item[index]	Gets or sets the `OracleBulkCopyColumnMappingCollection` object at the specified index

17.3.2.1 Item[index]

This property gets or sets the `OracleBulkCopyColumnMapping` object at the specified index.

Declaration

```
// 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

`IndexOutOfRangeException` - The specified index does not exist.

17.3.3 OracleBulkCopyColumnMappingCollection Public Methods

The `OracleBulkCopyColumnMappingCollection` public methods are listed in Table 17-16.

Table 17-16 OracleBulkCopyColumnMappingCollection Public Methods

Public Method	Description
Add	Adds objects to the collection
Clear	Clears the contents of the collection
Contains	Returns a value indicating whether or not a specified `OracleBulkCopyColumnMapping` object exists in the collection
CopyTo	Copies the elements of the `OracleBulkCopyColumnMappingCollection` to an array of `OracleBulkCopyColumnMapping` items, starting at a specified index
IndexOf	Returns the index of the specified `OracleBulkCopyColumnMapping` object
Insert	Inserts a new `OracleBulkCopyColumnMapping` object in the collection, at the index specified.
Remove	Removes the specified `OracleBulkCopyColumnMapping` element from the `OracleBulkCopyColumnMappingCollection`.
RemoveAt	Removes the mapping from the collection at the specified index.

17.3.3.1 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.

17.3.3.2 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.

17.3.3.3 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.

17.3.3.4 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

```
// C#
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.

17.3.3.5 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.

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.

17.3.3.6 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 name if the mapping is specified by ordinal position. To avoid this performance overhead, specify the mapping by column name.

17.3.3.7 Clear

This method clears the contents of the collection.

Declaration

```
// 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.

17.3.3.8 Contains

This method returns a value indicating whether or not a specified `OracleBulkCopyColumnMapping` object exists in the collection.

Declaration

```
// 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`.

17.3.3.9 CopyTo

This method copies the elements of the `OracleBulkCopyColumnMappingCollection` to an array of `OracleBulkCopyColumnMapping` items, starting at a specified index.

Declaration

```
// 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.

17.3.3.10 IndexOf

This method returns the index of the specified `OracleBulkCopyColumnMapping` object.

Declaration

```
// 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.

17.3.3.11 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.

17.3.3.12 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.

17.3.3.13 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.

17.4 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 17-17 lists all the `OracleBulkCopyOptions` enumeration values with a description of each enumerated value.

Table 17-17 OracleBulkCopyOptions Enumeration Members

Member Name	Description
`Default`	Indicates that the default value for all options are to be used
`UseInternalTransaction`	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.

Note:

All bulk copy operations are agnostic of any local or distributed transaction created by the application.

Requirements

Provider	ODP.NET, Unmanaged Driver
Assembly	`Oracle.DataAccess.dll`
Namespace	`Oracle.DataAccess.Client`
.NET Framework	3.5, 4.5, 4.6

17.5 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);
```

Requirements

Provider	ODP.NET, Unmanaged Driver
Assembly	`Oracle.DataAccess.dll`
Namespace	`Oracle.DataAccess.Client`
.NET Framework	3.5, 4.5, 4.6

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.

17.6 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
```

Requirements

Provider	ODP.NET, Unmanaged Driver
Assembly	`Oracle.DataAccess.dll`
Namespace	`Oracle.DataAccess.Client`
.NET Framework	3.5, 4.5, 4.6

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.

17.6.1 OracleRowsCopiedEventArgs Members

`OracleRowsCopiedEventArgs` members are listed in the following tables.

OracleRowsCopiedEventArgs Constructors

`OracleRowsCopiedEventArgs` constructors are listed in Table 17-18.

Table 17-18 OracleRowsCopiedEventArgs Constructors

Constructor	Description
OracleRowsCopiedEventArgs Constructors.	`OracleRowsCopiedEventArgs` creates new instances of the `OracleRowsCopiedEventArgs` class

OracleRowsCopiedEventArgs Properties

`OracleRowsCopiedEventArgs` properties are listed in Table 17-19.

Table 17-19 OracleRowsCopiedEventArgs Properties

Property	Description
Abort	Retrieves or sets a value that indicates whether or not the bulk copy operation is aborted
RowsCopied	Retrieves a value that represents the number of rows copied during the current bulk copy operation

17.6.2 OracleRowsCopiedEventArgs Constructors

`OracleRowsCopiedEventArgs` creates new instances of the `OracleRowsCopiedEventArgs` class.

Overload List:

- OracleRowsCopiedEventArgs(long)

  This constructor creates a new instance of the `OracleRowsCopiedEventArgs` object.

17.6.2.1 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.

17.6.3 OracleRowsCopiedEventArgs Properties

`OracleRowsCopiedEventArgs` properties are listed in Table 17-20.

Table 17-20 OracleRowsCopiedEventArgs Properties

Property	Description
Abort	Retrieves or sets a value that indicates whether or not the bulk copy operation is aborted
RowsCopied	Retrieves a value that represents the number of rows copied during the current bulk copy operation

17.6.3.1 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.

17.6.3.2 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.

A

Oracle Schema Collections

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:

- "Schema Discovery"
- "GetSchema"

This appendix contains the following topics:

- Common Schema Collections
- ODP.NET-Specific Schema Collection

A.1 Common Schema Collections

The common schema collections are available for all .NET Framework managed providers. ODP.NET supports the same common schema collections.

- MetaDataCollections
- DataSourceInformation
- DataTypes
- Restrictions
- ReservedWords

A.1.1 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 A-1 MetaDataCollections

Column Name	Data Type	Description
`CollectionName`	`string`	The name of the collection passed to the `GetSchema` method for retrieval.
`NumberOfRestrictions`	`int`	Number of restrictions specified for the named collection.
`NumberOfIdentifierParts`	`int`	Number of parts in the composite identifier/ database object name.

A.1.2 DataSourceInformation

Table A-2 lists `DataSourceInformation` information which may include these columns and possibly others.

Table A-2 DataSource Information

Columns	Data Type	Description
`CompositeIdentifierSeparatorPattern`	`string`	Separator for multipart names: `@` | `\` .
`DataSourceProductName`	`string`	Database name: Oracle
`DataSourceProductVersion`	`string`	Database version. Note that this is the version of the database instance currently being accessed by `DbConnection`.
`DataSourceProductVersionNormalized`	`string`	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
`GroupByBehavior`	`GroupByBehavior`	An enumeration that indicates the relationship between the columns in a `GROUP BY` clause and the non-aggregated columns in a select list.
`IdentifierPattern`	`string`	Format for a valid identifier.
`IdentifierCase`	`IdentifierCase`	An enumeration that specifies whether or not to treat non-quoted identifiers as case sensitive.
`OrderByColumnsInSelect`	`bool`	A boolean that indicates whether or not the select list must contain the columns in an `ORDER BY` clause.
`ParameterMarkerFormat`	`string`	A string indicating whether or not parameter markers begin with a special character.
`ParameterMarkerPattern`	`string`	The format of a parameter marker.
`ParameterNameMaxLength`	`int`	Maximum length of a parameter.
`ParameterNamePattern`	`string`	The format for a valid parameter name.
`QuotedIdentifierPattern`	`string`	The format of a quoted identifier.
`QuotedIdentifierCase`	`IdentifierCase`	An enumeration that specifies whether or not to treat quote identifiers as case sensitive.
`StringLiteralPattern`	`string`	The format for a string literal.
`SupportedJoinOperators`	`SupportedJoin Operators`	An enumeration indicating the types of SQL join statements supported by the data source.

A.1.3 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 Data Types

ColumnName	Data Type	Description
`TypeName`	`string`	The provider-specific data type name. Example: `TIMESTAMP WITH LOCAL TIME ZONE`
`ProviderDbType`	`int`	The provider-specific type value. Example: `124`
`ColumnSize`	`long`	The length of a non-numeric column or parameter. Example:`27`
`CreateFormat`	`string`	A format string that indicates how to add this column to a DDL statement. Example: `TIMESTAMP({0} WITH LOCAL TIME ZONE)`
`CreateParameters`	`string`	The parameters specified to create a column of this data type. Example: 8
`DataType`	`string`	The .NET type for the data type. Example: `System.DateTime`
`IsAutoIncrementable`	`bool`	A boolean value that indicates whether or not this data type can be auto-incremented. Example: `false`
`IsBestMatch`	`bool`	A boolean value that indicates whether or not this data type is the best match to values in the `DataType` column. Example: `false`

Table A-3 (Cont.) Data Types

ColumnName	Data Type	Description
`IsCaseSensitive`	`bool`	A boolean value that indicates whether or not this data type is case-sensitive. Example: `false`
`IsFixedLength`	`bool`	A boolean value that indicates whether or not this data type has a fixed length. Example: `true`
`IsFixedPrecisionScale`	`bool`	A boolean value that indicates whether or not this data type has a fixed precision and scale. Example: `false`
`IsLong`	`bool`	A boolean value that indicates whether or not this data type contains very long data. Example: `false`
`IsNullable`	`bool`	A boolean value that indicates whether or not this data type is nullable. Example: `true`
`IsSearchable`	`bool`	A boolean value that indicates whether or not the data type can be used in a `WHERE` clause with any operator, except the `LIKE` predicate. Example: `true`
`IsSearchableWithLike`	`bool`	A boolean value that indicates whether or not this data type can be used with the `LIKE` predicate. Example: `false`
`IsUnsigned`	`bool`	A boolean value that indicates whether or not the data type is unsigned.
`MaximumScale`	`short`	The maximum number of digits allowed to the right of the decimal point.
`MinimumScale`	`short`	The minimum number of digits allowed to the right of the decimal point.
`IsConcurrencyType`	`bool`	A boolean value that indicates whether or not the database updates the data type every time the row is changed and the value of the column differs from all previous values. Example: `false`

Table A-3 (Cont.) Data Types

ColumnName	Data Type	Description
`MinimumVersion`	`String`	The earliest version of the database that can be used. Example:`09.00.00.00.00`
`IsLiteralSupported`	`bool`	A boolean value that indicates whether or not the data type can be expressed as a literal. Example: `true`
`LiteralPrefix`	`string`	The prefix of a specified literal. Example: `TO_TIMESTAMP_TZ('`
`LiteralSuffix`	`string`	The suffix of a specified literal. Example: `','YYYY-MM-DD HH24:MI:SS.FF')`

A.1.4 Restrictions

Table A-4 lists Restrictions, including the following columns.

Table A-4 Restrictions

ColumnName	Data Type	Description
`CollectionName`	`string`	The collection that the restrictions apply to.
`RestrictionName`	`string`	The restriction name.
`RestrictionNumber`	`int`	A number that indicates the location of the restriction.

A.1.5 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 A-5 ReservedWords

ColumnName	Data Type	Description
`ReservedWord`	`string`	Provider-specific reserved words

A.2 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

A.2.1 Tables

Table A-6 lists the column name, data type, and description of the Tables Schema Schema Collection.

Table A-6 Tables

Column Name	Data Type	Description
`OWNER`	`String`	Owner of the Table.
`TABLE_NAME`	`String`	Name of the Table.
`TYPE`	`String`	Type of Table, for example, `System` or `User`.

A.2.2 Columns

Table A-7 lists the column name, data type, and description of the Columns Schema Collection .

Table A-7 Columns

ColumnName	Data Type	Description
`OWNER`	`String`	Owner of the table or view.
`TABLE_NAME`	`String`	Name of the table or view.
`COLUMN_NAME`	`String`	Name of the column.

Table A-7 (Cont.) Columns

ColumnName	Data Type	Description
`ID`	`Decimal`	Sequence number of the column as created.
`DATATYPE`	`String`	Data type of the column.
`LENGTH`	`Decimal`	Length of the column in bytes.
`PRECISION`	`Decimal`	Decimal precision for `NUMBER` data type; binary precision for `FLOAT` data type, null for all other data types.
`Scale`	`Decimal`	Digits to right of decimal point in a number.
`NULLABLE`	`String`	Specifies whether or not a column allows `NULL`s.
`CHAR_USED`	`String`	Indicates whether the column uses `BYTE` length semantics (B) or `CHAR` length semantics (C).
`LengthInChars`	`Decimal`	Length of the column in characters. This value only applies to `CHAR`, `VARCHAR2`, `NCHAR`, and `NVARCHAR2`.

A.2.3 Views

Table A-8 lists the column name, data type, and description of the Views Schema Collection.

Table A-8 Views

Column Name	Data Type	Description
`OWNER`	`String`	Owner of the view.
`VIEW_NAME`	`String`	Name of the view.
`TEXT_LENGTH`	`Decimal`	Length of the view text.
`TEXT`	`String`	View text.
`TYPE_TEXT_LENGTH`	`Decimal`	Length of the type clause of the typed view.
`TYPE_TEXT`	`String`	Type clause of the typed view.
`OID_TEXT_LENGTH`	`Decimal`	Length of the `WITH OID` clause of the typed view.
`OID_TEXT`	`String`	`WITH OID` clause of the typed view.
`VIEW_TYPE_OWNER`	`String`	Owner of the view type if the view is a typed view.
`VIEW_TYPE`	`String`	Type of the view if the view is a typed view.
`SUPERVIEW_NAME`	`String`	Name of the superview. (Oracle9*i* or later)

A.2.4 XMLSchema

Table A-9 lists the column name, data type and description of the XMLSchema Schema Collection.

Note:

This collection is only available with Oracle Database 10*g* and later.

Table A-9 XMLSchema

Column Name	Data Type	Description
OWNER	String	Owner of the XML schema.
SCHEMA_URL	String	Schema URL of the XML schema.
LOCAL	String	Indicates whether the XML schema is local (YES) or global (NO).
SCHEMA	String	XML schema document.
INT_OBJNAME	String	Internal database object name for the schema.
QUAL_SCHEMA_URL	String	Fully qualified schema URL.
HIER_TYPE	String	Hierarchy type for the schema.

A.2.5 Users

Table A-10 lists the column name, data type and description of the Users Schema Collection.

Table A-10 Users

Column Name	Data Type	Description
NAME	String	Name of the user.
ID	Decimal	ID number of the user.
CREATEDATE	DateTime	User creation date.

A.2.6 Synonyms

Table A-11 lists the column name, data type and description of the Synonyms Schema Collection.

Table A-11 Synonyms

Column Name	Data Type	Description
OWNER	String	Owner of the synonym.
SYNONYM_NAME	String	Name of the synonym.
TABLE_OWNER	String	Owner of the object referenced by the synonym. Although the column is called TABLE_OWNER, the object owned is not necessarily a table. It can be any general object such as a view, sequence, stored procedure, synonym, and so on.

Table A-11 (Cont.) Synonyms

Column Name	Data Type	Description
TABLE_NAME	String	Name of the object referenced by the synonym. Although the column is called TABLE_NAME, the object does not necessarily have to be a table. It can be any general object such as a view, sequence, stored procedure, synonym, and so on.
DB_LINK	String	Name of the database link referenced, if any.

A.2.7 Sequences

Table A-12 lists the column name, data type, and description of the Sequences Schema Collection.

Table A-12 Sequences

Column Name	Data Type	Description
SEQUENCE_OWNER	String	Name of the owner of the sequence.
SEQUENCE_NAME	String	Sequence name.
MIN_VALUE	Decimal	Minimum value of the sequence.
MAX_VALUE	Decimal	Maximum value of the sequence.
INCREMENT_BY	Decimal	Value by which sequence is incremented.
CYCLE_FLAG	String	Indicates if sequence wraps around on reaching limit.
ORDER_FLAG	String	Indicates if sequence numbers are generated in order.
CACHE_SIZE	Decimal	Number of sequence numbers to cache.
LAST_NUMBER	Decimal	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.

A.2.8 Functions

Table A-13 lists the column name, data type, and description of the Functions Schema Collection.

Table A-13 Functions

Column Name	Data Type	Description
OWNER	String	Owner of the function.
OBJECT_NAME	String	Name of the function.
SUBOBJECT_NAME	String	Name of the subobject (for example, partition).
OBJECT_ID	Decimal	Dictionary object number of the function.

Table A-13 (Cont.) Functions

Column Name	Data Type	Description
`DATA_OBJECT_ID`	`Decimal`	Dictionary object number of the segment that contains the function.
`CREATED`	`DateTime`	Timestamp for the creation of the function.
`LAST_DDL_TIME`	`DateTime`	Timestamp for the last modification of the function resulting from a DDL statement (including grants and revokes).
`TIMESTAMP`	`String`	Timestamp for the specification of the function (character data).
`STATUS`	`String`	Status of the function (`VALID`, `INVALID`, or N/A).
`TEMPORARY`	`String`	Whether or not the function is temporary (the current session can see only data that it placed in this object itself).
`GENERATED`	`String`	Indicates whether the name of this function is system generated (`Y`) or not (`N`).
`SECONDARY`	`String`	Whether or not this is a secondary object created by the `ODCIIndexCreate` method of the Oracle Data Cartridge (`Y` | `N`).

A.2.9 Procedures

Table A-14 lists the column name, data type, and description of the Procedures Schema Collection.

Table A-14 Procedures

Column Name	Data Type	Description
`OWNER`	`String`	Owner of the procedure.
`OBJECT_NAME`	`String`	Name of the procedure.
`SUBOBJECT_NAME`	`String`	Name of the subobject (for example, partition).
`OBJECT_ID`	`Decimal`	Dictionary object number of the procedure.
`DATA_OBJECT_ID`	`Decimal`	Dictionary object number of the segment that contains the procedure.
`CREATED`	`DateTime`	Timestamp for the creation of the procedure.
`LAST_DDL_TIME`	`Decimal`	Timestamp for the last modification of the procedure resulting from a DDL statement (including grants and revokes).
`TIMESTAMP`	`String`	Timestamp for the specification of the procedure (character data).
`STATUS`	`String`	Status of the procedure (`VALID`, `INVALID`, or N/A).
`TEMPORARY`	`String`	Whether or not the procedure is temporary (the current session can see only data that it placed in this object itself).
`GENERATED`	`String`	Indicates whether the name of this procedure is system generated (`Y`) or not (`N`).

Table A-14 (Cont.) Procedures

Column Name	Data Type	Description
SECONDARY	String	Whether or not this is a secondary object created by the ODCIIndexCreate method of the Oracle Data Cartridge (Y | N).

A.2.10 ProcedureParameters

Table A-15 lists the column name, data type and description of the ProcedureParameters Schema Collection.

Table A-15 ProcedureParameters

Column Name	Data Type	Description
OWNER	String	Owner of the object.
OBJECT_NAME	String	Name of the procedure or function.
PACKAGE_NAME	String	Name of the package.
OBJECT_ID	Decimal	Object number of the object.
OVERLOAD	String	Indicates the *n*th overloading ordered by its appearance in the source; otherwise, it is NULL.
SUBPROGRAM_ID	Decimal	Subprogram id for the procedure or function
ARGUMENT_NAME	String	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.
POSITION	Decimal	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.
SEQUENCE	Decimal	Defines the sequential order of the argument. Argument sequence starts from 1.
DATA_LEVEL	Decimal	Nesting depth of the argument for composite types.
DATA_TYPE	String	Data type of the argument.
DEFAULT_VALUE	String	Default value for the argument.
DEFAULT_LENGTH	Decimal	Length of the default value for the argument.
IN_OUT	String	Direction of the argument: [IN] [OUT] [IN/OUT].
DATA_LENGTH	Decimal	Length of the column (in bytes).
DATA_PRECISION	Decimal	Length in decimal digits (NUMBER) or binary digits (FLOAT).
DATA_SCALE	Decimal	Digits to the right of the decimal point in a number.
RADIX	Decimal	Argument radix for a number.
CHARACTER_SET_NAME	String	Character set name for the argument.
TYPE_OWNER	String	Owner of the type of the argument.

Table A-15 (Cont.) ProcedureParameters

Column Name	Data Type	Description
TYPE_NAME	String	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.
TYPE_SUBNAME	String	Displays the name of the type declared in the package identified in the TYPE_NAME column. Relevant only for package local types.
TYPE_LINK	String	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.
PLS_TYPE	String	For numeric arguments, the name of the PL/SQL type of the argument. Otherwise, Null.
CHAR_LENGTH	Decimal	Character limit for string data types.
CHAR_USED	String	Indicates whether the byte limit (B) or character limit (C) is official for the string.

A.2.11 Arguments

Table A-16 lists the column name, data type, and description of the Arguments Schema Collection.

Table A-16 Arguments

Column Name	Data Type	Description
OWNER	String	Owner of the object.
PACKAGE_NAME	String	Name of the package.
OBJECT_NAME	String	Name of the procedure or function.
ARGUMENT_NAME	String	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.
POSITION	Decimal	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.
SEQUENCE	Decimal	Defines the sequential order of the argument. Argument sequence starts from 1.
DEFAULT_VALUE	String	Default value for the argument.
DEFAULT_LENGTH	Decimal	Length of the default value for the argument.
IN_OUT	String	Direction of the argument: [IN] [OUT] [IN/OUT].
DATA_LENGTH	Decimal	Length of the column (in bytes).
DATA_PRECISION	Decimal	Length in decimal digits (NUMBER) or binary digits (FLOAT).
DATA_SCALE	Decimal	Digits to the right of the decimal point in a number.

Table A-16 (Cont.) Arguments

Column Name	Data Type	Description
`DATA_TYPE`	`String`	Data type of the argument.
`CHAR_USED`	`String`	Indicates whether the column uses `BYTE` length semantics (B) or `CHAR` length semantics (C).

A.2.12 Packages

Table A-17 lists the column name, data type, and description of the Packages Schema Collection.

Table A-17 Packages

Column Name	Data Type	Description
`OWNER`	`String`	Owner of the package.
`OBJECT_NAME`	`String`	Name of the package.
`SUBOBJECT_NAME`	`String`	Name of the subobject (for example, partition).
`OBJECT_ID`	`Decimal`	Dictionary object number of the package.
`DATA_OBJECT_ID`	`Decimal`	Dictionary object number of the segment that contains the package.
`CREATED`	`DateTime`	Timestamp for the creation of the package.
`LAST_DDL_TIME`	`DateTime`	Timestamp for the last modification of the package resulting from a DDL statement (including grants and revokes).
`TIMESTAMP`	`String`	Timestamp for the specification of the package (character data).
`STATUS`	`String`	Status of the package (`VALID`, `INVALID`, or N/A).
`TEMPORARY`	`String`	Whether or not the package is temporary (the current session can see only data that it placed in this object itself).
`GENERATED`	`String`	Indicates whether the name of this package was system generated (`Y`) or not (`N`).
`SECONDARY`	`String`	Whether or not this is a secondary object created by the `ODCIIndexCreate` method of the Oracle Data Cartridge (`Y` | `N`).

A.2.13 PackageBodies

Table A-18 lists the column name, data type, and description of the PackageBodies Schema Collection.

Table A-18 PackageBodies

Column Name	Data Type	Description
`OWNER`	`String`	Owner of the package body.

Table A-18 (Cont.) PackageBodies

Column Name	Data Type	Description
OBJECT_NAME	String	Name of the package body.
SUBOBJECT_NAME	String	Name of the subobject (for example, partition).
OBJECT_ID	Decimal	Dictionary object number of the package body.
DATA_OBJECT_ID	Decimal	Dictionary object number of the segment that contains the package body.
CREATED	DateTime	Timestamp for the creation of the package body.
LAST_DDL_TIME	DateTime	Timestamp for the last modification of the package body resulting from a DDL statement (including grants and revokes).
TIMESTAMP	String	Timestamp for the specification of the package body (character data).
STATUS	String	Status of the package body (VALID, INVALID, or N/A).
TEMPORARY	String	Whether the package body is temporary (the current session can see only data that it placed in this object itself).
GENERATED	String	Indicates whether the name of this package body is system generated (Y) or not (N).
SECONDARY	String	Whether or not this is a secondary object created by the ODCIIndexCreate method of the Oracle Data Cartridge (Y | N).

A.2.14 JavaClasses

Table A-19 lists the column name, data type, and description of the JavaClasses Schema Collection.

Table A-19 JavaClasses

Column Name	Data Type	Description
OWNER	String	Owner of the Java class.
NAME	String	Name of the Java class.
MAJOR	Decimal	Major version number of the Java class, as defined in the JVM specification.
MINOR	Decimal	Minor version number of the Java class, as defined in the JVM specification.
KIND	String	Indicates whether the stored object is a Java class (CLASS) or a Java interface (INTERFACE).
ACCESSIBILITY	String	Accessibility of the Java class.
IS_INNER	String	Indicates whether this Java class is an inner class (YES) or not (NO).
IS_ABSTRACT	String	Indicates whether this Java class is an abstract class (YES) or not (NO).

Table A-19 (Cont.) JavaClasses

Column Name	Data Type	Description
IS_FINAL	String	Indicates whether this Java class is a final class (YES) or not (NO).
IS_DEBUG	String	Indicates whether this Java class contains debug information (YES) or not (NO).
SOURCE	String	Source designation of the Java class.
SUPER	String	Super class of this Java class.
OUTER	String	Outer class of this Java class if this Java class is an inner class.

A.2.15 Indexes

Table A-20 lists the column name, data type, and description of the Indexes Schema Collection.

Table A-20 Indexes

Column Name	Data Type	Description
OWNER	String	Owner of the index.
INDEX_NAME	String	Name of the index.
INDEX_TYPE	String	Type of the index: • NORMAL • BITMAP • FUNCTION-BASED NORMAL • FUNCTION-BASED BITMAP • DOMAIN
TABLE_OWNER	String	Owner of the indexed object.
TABLE_NAME	String	Name of the indexed object.
TABLE_TYPE	String	Type of the indexed object (for example, TABLE or CLUSTER).
UNIQUENESS	String	Indicates whether the index is UNIQUE or NONUNIQUE.
COMPRESSION	String	Indicates whether index compression is enabled (ENABLED) or not (DISABLED).
PREFIX_LENGTH	Decimal	Number of columns in the prefix of the compression key.
TABLESPACE_NAME	String	Name of the tablespace containing the index.
INI_TRANS	Decimal	Initial number of transactions.
MAX_TRANS	Decimal	Maximum number of transactions.
INITIAL_EXTENT	Decimal	Size of the initial extent.
NEXT_EXTENT	Decimal	Size of secondary extents.
MIN_EXTENTS	Decimal	Minimum number of extents allowed in the segment.
MAX_EXTENTS	Decimal	Maximum number of extents allowed in the segment.

Table A-20 (Cont.) Indexes

Column Name	Data Type	Description
PCT_INCREASE	Decimal	Percentage increase in extent size.
PCT_THRESHOLD	Decimal	Threshold percentage of block space allowed per index entry.
INCLUDE_COLUMN	Decimal	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.
FREELISTS	Decimal	Number of process freelists allocated to this segment.
FREELIST_GROUPS	Decimal	Number of freelist groups allocated to this segment.
PCT_FREE	Decimal	Minimum percentage of free space in a block.
LOGGING	String	Logging information.
BLEVEL	Decimal	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.
LEAF_BLOCKS	Decimal	Number of leaf blocks in the index.
DISTINCT_KEYS	Decimal	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).
AVG_LEAF_BLOCKS_PER_KEY	Decimal	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.
AVG_DATA_BLOCKS_PER_KEY	Decimal	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.
CLUSTERING_FACTOR	Decimal	Indicates the amount of order of the rows in the table based on the values of the index.
STATUS	String	Indicates whether a nonpartitioned index is VALID or UNUSABLE.
NUM_ROWS	Decimal	Number of rows in the index.
SAMPLE_SIZE	Decimal	Size of the sample used to analyze the index.
LAST_ANALYZED	Date	Date on which this index was most recently analyzed.
DEGREE	String	Number of threads per instance for scanning the index.
INSTANCES	String	Number of instances across which the indexes to be scanned.
PARTITIONED	String	Indicates whether the index is partitioned (YES) or not (NO).
TEMPORARY	String	Indicates whether or not the index is on a temporary table.

Table A-20 (Cont.) Indexes

Column Name	Data Type	Description
GENERATED	String	Indicates whether the name of the index is system generated (Y) or not (N).
SECONDARY	String	Indicates whether the index is a secondary object created by the ODCIIndexCreate method of the Oracle Data Cartridge (Y) or not (N).
BUFFER_POOL	String	Name of the default buffer pool to be used for the index blocks.
USER_STATS	String	Indicates whether statistics were entered directly by the user (YES) or not (NO).
DURATION	String	Indicates the duration of a temporary table.
PCT_DIRECT_ACCESS	Decimal	For a secondary index on an index-organized table, the percentage of rows with VALID guess.
ITYP_OWNER	String	For a domain index, the owner of the index type.
ITYP_NAME	String	For a domain index, the name of the index type.
PARAMETERS	String	For a domain index, the parameter string.
GLOBAL_STATS	String	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).
DOMIDX_STATUS	String	Status of the domain index: • NULL - Index is not a domain index. • VALID - Index is a valid domain index. • IDXTYP_INVLD - Indextype of the domain index is invalid.
DOMIDX_OPSTATUS	String	Status of the operation on the domain index: • NULL - Index is not a domain index. • VALID - Operation performed without errors. • FAILED - Operation failed with an error.
FUNCIDX_STATUS	String	Status of a function-based index: • NULL - Index is not a function-based index. • ENABLED - Function-based index is enabled. • DISABLED - Function-based index is disabled.
JOIN_INDEX	String	Indicates whether the index is a join index (YES) or not (NO).
IOT_REDUNDANT_PKEY_ELIM	String	Indicates whether redundant primary key columns are eliminated from secondary indexes on index-organized tables (YES) or not (NO).
DROPPED	String	Indicates whether the index has been dropped and is in the recycle bin (YES) or not (NO); null for partitioned tables.

A.2.16 IndexColumns

Table A-21 lists the column name, data type, and description of the IndexColumns Schema Collection.

Table A-21 IndexColumns

Column Name	Data Type	Description
`INDEX_OWNER`	`String`	Owner of the index.
`INDEX_NAME`	String	Name of the index.
`TABLE_OWNER`	String	Owner of the table or cluster.
`TABLE_NAME`	String	Name of the table or cluster.
`COLUMN_NAME`	String	Column name or attribute of object type column.
`COLUMN_POSITION`	`Decimal`	Position of column or attribute within the index.
`COLUMN_LENGTH`	`Decimal`	Indexed length of the column.
`DESCEND`	String	Whether the column is sorted in descending order (`Y`/`N`).
`CHAR_LENGTH`	`Decimal`	Maximum codepoint length of the column. (Oracle9*i* or later)

A.2.17 PrimaryKeys

Table A-22 lists the column name, data type, and description of the PrimaryKeys Schema Collection.

Table A-22 PrimaryKeys

Column Name	Data Type	Description
`OWNER`	`String`	Owner of the constraint definition.
`CONSTRAINT_NAME`	`String`	Name of the constraint definition.
`TABLE_NAME`	`String`	Name associated with the table (or view) with constraint definition.
`SEARCH_CONDITION`	`String`	Text of search condition for a check constraint.
`R_OWNER`	`String`	Owner of table referred to in a referential constraint.
`R_CONSTRAINT_NAME`	`String`	Name of the unique constraint definition for referenced table.
`DELETE_RULE`	`String`	Delete rule for a referential constraint (`CASCADE` or `NO ACTION`).
`STATUS`	`String`	Enforcement status of constraint (`ENABLED` or `DISABLED`).
`DEFERRABLE`	`String`	Whether or not the constraint is deferrable.
`VALIDATED`	`String`	Whether all data obeys the constraint (`VALIDATED` or `NOT VALIDATED`).
`GENERATED`	`String`	Whether the name of the constraint is user or system generated.

Table A-22 (Cont.) PrimaryKeys

Column Name	Data Type	Description
BAD	String	Indicates that this constraint specifies a century in an ambiguous manner. (Yes| No) To avoid errors resulting from this ambiguity, rewrite the constraint using the TO_DATE function with a four-digit year.
RELY	String	Whether an enabled constraint is enforced or unenforced.
LAST_CHANGE	DateTime	When the constraint was last enabled or disabled.
INDEX_OWNER	String	Name of the user owning the index. (Oracle9*i* or later)
INDEX_NAME	String	Name of the index (only shown for unique and primary-key constraints). (Oracle9*i* or later)

A.2.18 ForeignKeys

Table A-23 lists the column name, data type, and description of the ForeignKeys Schema Collection.

Table A-23 ForeignKeys

Column Name	Data Type	Description
PRIMARY_KEY_CONSTRAINT_NAME	String	Name of the constraint definition.
PRIMARY_KEY_OWNER	String	Owner of the constraint definition.
PRIMARY_KEY_TABLE_NAME	String	Name associated with the table (or view) with constraint definition.
FOREIGN_KEY_OWNER	String	Owner of the constraint definition.
FOREIGN_KEY_CONSTRAINT_NAME	String	Name of the constraint definition.
FOREIGN_KEY_TABLE_NAME	String	Name associated with the table (or view) with constraint definition.
SEARCH_CONDITION	String	Text of search condition for a check constraint
R_OWNER	String	Owner of table referred to, in a referential constraint.
R_CONSTRAINT_NAME	String	Name of the unique constraint definition for referenced table.
DELETE_RULE	String	Delete rule for a referential constraint (CASCADE or NO ACTION).
STATUS	String	Enforcement status of constraint (ENABLED or DISABLED).
VALIDATED	String	Whether or not all data obeys the constraint (VALIDATED or NOT VALIDATED).

Table A-23 (Cont.) ForeignKeys

Column Name	Data Type	Description
`GENERATED`	`String`	Whether the name of the constraint is user or system generated.
`RELY`	`String`	Whether an enabled constraint is enforced or unenforced.
`LAST_CHANGE`	`DateTime`	When the constraint was last enabled or disabled.
`INDEX_OWNER`	`String`	Name of the user owning the index. (Oracle9*i* or later)
`INDEX_NAME`	`String`	Name of the index. (Oracle9*i* or later)

A.2.19 ForeignKeyColumns

Table A-24 lists the column name, data type, and description of the ForeignKeyColumns Schema Collection.

Table A-24 ForeignKeyColumns

Column Name	Data Type	Description
`OWNER`	`String`	Owner of the constraint definition.
`CONSTRAINT_NAME`	`String`	Name of the constraint definition.
`TABLE_NAME`	`String`	Name of the table with constraint definition.
`COLUMN_NAME`	`String`	Name of the column or attribute of the object type column specified in the constraint definition.
`POSITION`	`String`	Original position of column or attribute in the definition of the object.

A.2.20 UniqueKeys

Table A-25 lists the column name, data type, and description of the UniqueKeys Schema Collection.

Table A-25 UniqueKeys

Column Name	Data Type	Description
`OWNER`	`String`	Owner of the constraint definition.
`CONSTRAINT_NAME`	`String`	Name of the constraint definition.
`TABLE_NAME`	`String`	Name associated with the table (or view) with constraint definition.
`SEARCH_CONDITION`	`String`	Text of search condition for a check constraint.
`R_OWNER`	`String`	Owner of table referred to in a referential constraint.
`R_CONSTRAINT_NAME`	`String`	Name of the unique constraint definition for referenced table.

Table A-25 (Cont.) UniqueKeys

Column Name	Data Type	Description
`DELETE_RULE`	`String`	Delete rule for a referential constraint (`CASCADE` or `NO ACTION`).
`STATUS`	`String`	Enforcement status of constraint (`ENABLED` or `DISABLED`).
`DEFERRABLE`	`String`	Whether or not the constraint is deferrable.
`VALIDATED`	`String`	Whether all data obeys the constraint (`VALIDATED` or `NOT VALIDATED`).
`GENERATED`	`String`	Whether the name of the constraint is user or system generated.
`BAD`	`String`	Indicates that this constraint specifies a century in an ambiguous manner. (`Yes`| `No`) To avoid errors resulting from this ambiguity, rewrite the constraint using the `TO_DATE` function with a four-digit year.
`RELY`	`String`	Whether an enabled constraint is enforced or not.
`LAST_CHANGE`	`String`	When the constraint was last enabled or disabled.
`INDEX_OWNER`	`String`	Name of the user owning the index. (Oracle9*i* or later)
`INDEX_NAME`	`String`	Name of the index (only shown for unique and primary-key constraints). (Oracle9*i* or later)

B
Mapping LINQ Canonical Functions and Oracle Functions

This appendix lists the Entity Framework canonical functions and the corresponding ODP.NET provider functions to which they map.

Aggregate Canonical Functions

Table B-1 Mapping of Aggregate Canonical Functions and Oracle Functions

Canonical Function	Oracle Function
`Avg ( `*`expression`*` )`	`AVG(`*`expression`*`)`
`BigCount ( `*`expression`*` )`	`COUNT(`*`expression`*`)`
`Count ( `*`expression`*` )`	`COUNT(`*`expression`*`)`
`Max ( `*`expression`*` )`	`MAX(`*`expression`*`)`
`Min ( `*`expression`*` )`	`MIN(`*`expression`*`)`
`StDev ( `*`expression`*` )`	`STDDEV(`*`expression`*`)`
`StDevP( `*`expression`*`)`	`STDEVP(`*`expression`*`)`
`Sum ( `*`expression`*` )`	`SUM (`*`expression`*`)`
`Var(`*`expression`*`)`	`VAR(`*`expression`*`)`
`VarP(`*`expression`*`)`	`VARP(`*`expression`*`)`

Math Canonical Functions

Table B-2 Mapping of Math Canonical Functions and Oracle Functions

Canonical Function	Oracle Function
`Abs ( `*`value`*` )`	`ABS (`*`value`*`)`
`Ceiling ( `*`value`*` )`	`CEIL(`*`value`*`)`
`Floor ( `*`value`*` )`	`FLOOR(`*`value`*`)`
`Power(`*`value`*`, `*`exponent`*`)`	`POWER(`*`value`*`, `*`exponent`*`)`
`Round ( `*`value`*` )`	`ROUND(`*`value`*`)`
`Round ( `*`value`*`, `*`digits`*` )`	`ROUND(`*`value`*`, `*`digits`*`)`
`Truncate(`*`value`*`, `*`digits`*`)`	`TRUNC(`*`value`*`, `*`digits`*`)`

String Canonical Functions

Table B-3 Mapping of String Canonical Functions and Oracle Functions

Canonical Function	Oracle Function

`Concat ( string1, string2)`	`CONCAT(string1, string2)`
	or
	`( (string1) || (string2))`

`Contains(string, target)`	`INSTR(string, target)`
`EndsWith(string, target)`	`INSTR(REVERSE(string), REVERSE(target))`
Comparison operators `(<, <=, >, >=, <>, !=)`	Comparison operators `(<, <=, >, >=, <>, !=)`
`IndexOf( target, string)`	`INSTR(string2, target)`
`Left ( string1, length)`	`SUBSTR(string1, length)`
`Length ( string )`	`LENGTH(string)`
`LTrim( string )`	`LTRIM(string)`
`Replace ( string1, string2, string3)`	`REPLACE(string1, string2, string3)`
`Reverse ( string )`	`REVERSE(string)`
`Right ( string, length)`	`(CASE WHEN LENGTH(string) >= (length) THEN SUBSTR (string) ,-(length), length ) ELSE string END)`
`RTrim( string )`	`RTRIM(string)`
`Substring ( string, start, length)`	`SUBSTR(( string, start, length)`
`StartsWith(string, target)`	`INSTR(string, target)`
`ToLower ( string )`	`LOWER(string)`
`ToUpper( string )`	`UPPER`
`Trim ( string )`	`LTRIM(RTRIM(string))`

Date And Time Canonical Functions

Table B-4 Mapping of Date And Time Canonical Functions and Oracle Functions

Canonical Function	Oracle Function
`AddNanoseconds(expression, number)`	`(expression) + INTERVAL`
`AddMicroseconds(expression, number)`	`(expression) + INTERVAL`
`AddMilliseconds(expression, number)`	`(expression) + INTERVAL`
`AddSeconds(expression, number)`	`(expression) + INTERVAL`
`AddMinutes(expression, number)`	`(expression) + INTERVAL`
`AddHours(expression, number)`	`(expression) + INTERVAL`

Table B-4 (Cont.) Mapping of Date And Time Canonical Functions and Oracle Functions

Canonical Function	Oracle Function
`AddDays(`*`expression`*`, `*`number`*`)`	`(`*`expression`*`) + INTERVAL`
`AddMonths(`*`expression`*`, `*`number`*`)`	`(`*`expression`*`) + INTERVAL`
`AddYears(`*`expression`*`, `*`number`*`)`	`(`*`expression`*`) + INTERVAL`
`CreateDateTime(`*`year`*`, `*`month`*`, `*`day`*`, `*`hour`*`, `*`minute`*`, `*`second`*`)`	`TO_TIMESTAMP`
`CreateDateTimeOffset(`*`year`*`, `*`month`*`, `*`day`*`, `*`hour`*`, `*`minute`*`, `*`second`*`, `*`tzoffset`*`)`	`TO_TIMESTAMP_TZ`
`CreateTime(`*`hour`*`, `*`minute`*`, `*`second`*`)`	Time literals are not supported in Oracle
`CurrentDateTime()`	`LOCALTIMESTAMP`
`CurrentDateTimeOffset()`	`SYSTIMESTAMP`
`CurrentUtcDateTime()`	`SYS_EXTRACT_UTC` `(LOCALTIMESTAMP)`
`Day(`*`expression`*`)`	`EXTRACT(DAY FROM `*`expression`*`)`
`DayOfYear(`*`expression`*`)`	`TO_NUMBER(TO_CHAR(CAST(`*`expression`*` AS TIMESTAMP), 'DDD'))`
`DiffNanoseconds(`*`startExpression`*`, `*`endExpression`*`)`	`EXTRACT` and arithmetic operations
`DiffMilliseconds(`*`startExpression`*`, `*`endExpression`*`)`	`EXTRACT` and arithmetic operations
`DiffMicroseconds(`*`startExpression`*`, `*`endExpression`*`)`	`EXTRACT` and arithmetic operations
`DiffSeconds(`*`startExpression`*`, `*`endExpression`*`)`	`EXTRACT` and arithmetic operations
`DiffMinutes(`*`startExpression`*`, `*`endExpression`*`)`	`EXTRACT` and arithmetic operations
`DiffHours(`*`startExpression`*`, `*`endExpression`*`)`	`EXTRACT` and arithmetic operations
`DiffDays(`*`startExpression`*`, `*`endExpression`*`)`	`EXTRACT` and arithmetic operations
`DiffMonths(`*`startExpression`*`, `*`endExpression`*`)`	`EXTRACT` and arithmetic operations
`DiffYears(`*`startExpression`*`, `*`endExpression`*` )`	`EXTRACT` and arithmetic operations
Comparison operators `(<, <=, >, >=, <>, !=)`	`<, <=, >, >=, <>, !=` operators
`GetTotalOffsetMinutes ( `*`datetimeoffset`*` )`	`(EXTRACT(TIMEZONE_HOUR FROM (`*`expression`*`))) * 60 + EXTRACT (TIMEZONE_MINUTE FROM(`*`expression`*`))` (Require multiple operations.)
`Hour (`*`expression`*`)`	`EXTRACT(HOUR FROM `*`expression`*`)`

Table B-4 (Cont.) Mapping of Date And Time Canonical Functions and Oracle Functions

Canonical Function	Oracle Function

`Millisecond(`*`expression`*`)`	`NVL(TO_NUMBER(SUBSTR(TO_CHAR(CAST(`*`expression`*` AS TIMESTAMP), 'DD-`
	`MON-RR HH24:MI:SSXFF'), 20, 3)), 0)`

`Minute(`*`expression`*`)`	`EXTRACT(MINUTE FROM `*`expression`*`)`
`Month (`*`expression`*`)`	`EXTRACT(MONTH FROM `*`expression`*`)`
`Second(`*`expression`*`)`	`EXTRACT (SECOND FROM `*`expression`*`)`
`TruncateDate(`*`expression`*`)`	`TRUNC(`*`expression`*`)`
`Year(`*`expression`*`)`	`EXTRACT(YEAR FROM `*`expression`*`)`

Bitwise Canonical Functions

Table B-5 Mapping of Bitwise Canonical Functions and Oracle Functions

Canonical Function	Oracle Function
`BitWiseAnd ( `*`value1`*` , `*`value2`*` )`	`BITAND(`*`value1`*`, `*`value2`*`)`
`BitWiseNot ( `*`value`*` )`	`( 0 - `*`value`*`) - 1`
`BitWiseOr ( `*`value1`*` , `*`value2`*` )`	`Value1 - BITAND(`*`value1`*`, `*`value2`*`) + `*`value2`*
`BitWiseXor ( `*`value1`*` , `*`value2`*` )`	`Value1 - 2 * BITAND(`*`value1`*`, `*`value2`*`) + `*`value2`*

Other Canonical Functions

Table B-6 Mapping of Other Canonical Functions and Oracle Functions

Canonical Function	Oracle Function
`NewGuid()`	`SYS_GUID`

Glossary

assembly

Assembly is Microsoft's term for the module that is created when a DLL or .EXE is complied by a .NET compiler.

BFILES

External binary files that exist outside the database tablespaces residing in the operating system. BFILES are referenced from the database semantics, and are also known as external LOBs.

Binary Large Object (BLOB)

A large object 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 `BLOB`s, `CLOB`s, and `NCLOB`s while external LOBs include `BFILE`s.

Microsoft .NET Framework Class Library

The Microsoft .NET Framework Class Library provides the classes for the .NET framework model.

namespace

- .NET:

  A namespace is naming device for grouping related types. More than one namespace can be contained in an assembly.

- XML Documents:

  A namespace describes a set of related element names or attributes within an XML document.

National Character Large Object (NCLOB)

The LOB 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.

Index