

Oracle® Rdb Data Provider for .NET

Developer's Guide

Release 7.3.4.0

June 2015

Oracle Rdb Data Provider for .NET Developer's Guide, Release 7.3.4.0

Copyright © 2011, 2015 Oracle and/or its affiliates. All rights reserved.

Primary Author: Jim Murray.

Contributing Author:

Contributor: Colleen Mitchneck.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, duplication, disclosure, modification, and adaptation shall be subject to the restrictions and license terms set forth in the applicable Government contract, and, to the extent applicable by the terms of the Government contract, the additional rights set forth in FAR 52.227-19, Commercial Computer Software License (December 2007). Oracle America, Inc., 500 Oracle Parkway, Redwood City, CA 94065.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD,

Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information on content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services.

Contents

Preface	6
Chapter 1 Introducing Oracle Rdb Data Provider for .NET	9
1.1 Overview of Oracle Rdb Data Provider for .NET (ORDP.NET).....	9
1.2 ORDP.NET Assembly	9
1.3 Oracle.DataAccess.RdbClient Classes	9
1.4 Oracle.DataAccess.RdbClient Enumerations.....	10
1.5 Using ORDP.NET in a Simple Application	10
Chapter 2 Installing and Configuring	12
2.1 System Requirements	12
2.2 Installing Oracle Rdb Data Provider for .NET	12
2.3 File Locations	12
2.4 Post Installation Procedures.....	12
Chapter 3 Features of Oracle Rdb Data Provider for .NET	13
3.1 Connecting to an Oracle Rdb Database	13
3.1.1 Connection String Attributes	13
3.1.2 SQL/Services Service connections	14
3.1.3 JDBC Server connections	15
3.1.4 Connection Pooling.....	16
3.1.5 Connection Pool Management.....	19
3.2 ADO.NET 2.0 Features.....	25
3.2.1 About ADO.NET 2.0	25
3.2.2 Base Classes and Provider Factory Classes	25
3.2.3 Connection String Builder	26
3.2.4 Support for Schema Discovery	27
3.2.5 User Customization of Metadata	27
3.3 Controlling the Number of Rows Fetched in One Server Round-Trip.....	28
3.3.1 Use of FetchSize	28
3.3.2 Fine-Tuning FetchSize.....	28
3.3.3 Setting FetchSize Value at Design Time	29
3.4 Transaction	29
3.4.1 Implicit Transactions	29
3.4.2 Explicit Transactions	30
3.4.3 TransactionScope.....	30
3.5 Guaranteeing Uniqueness in Updating DataSet to Database	31

3.5.1	What Constitutes Uniqueness in DataRow?	32
3.5.2	Configuring PrimaryKey and Constraints Properties	32
3.5.3	Updating Without PrimaryKey and Constraints Configuration	33
3.6	<i>Data Provider Pattern in .NET V2.0</i>	33
3.6.1	Identification of the Oracle Rdb Data Provider for .NET	36
3.6.2	The DbDataProviderFactory for ORDP.NET	36
3.6.3	The DbConnectionStringBuilder for ORDP.NET	36
3.6.4	Registration of the DbProviderFactory for ORDP.NET	36
3.7	<i>External Procedures</i>	37
3.8	<i>SSL and THIN connectivity</i>	39
3.8.1	CertName connection string attribute	40
3.8.2	SSLMode - NONE	41
3.8.3	SSLMode - VERIFYFULL	41
3.8.4	SSLMode - VERIFYCA	42
3.8.5	SSLMode - ALLOWUNTRUSTED	42
3.8.6	SSLMode - REQUIRED	42
3.9	<i>Debug Tracing</i>	43
3.9.2	Registry Settings for Tracing Calls	44
Chapter 4	Oracle.DataAccess.RdbClient Namespace	45
4.1	<i>Overview of Oracle Rdb Data Provider Classes</i>	45
4.2	<i>Oracle Rdb Data Provider Classes</i>	46
4.2.1	RdbCommand Class	46
4.2.2	RdbCommandBuilder Class	61
4.2.3	RdbConnection Class	68
4.2.4	RdbDataAdapter Class	103
4.2.5	RdbDataReader Class	113
4.2.6	RdbError Class	138
4.2.7	RdbErrorCollection Class	142
4.2.8	RdbException Class	145
4.2.9	RdbInfoMessageEventArgs Class	149
4.2.10	RdbInfoMessageEventHandler Delegate	152
4.2.11	RdbParameter Class	153
4.2.12	RdbParameterCollection Class	168
4.2.13	RdbRowUpdatedEventHandler Delegate	186
4.2.14	RdbRowUpdatedEventArgs Class	186
4.2.15	RdbRowUpdatingEventArgs Class	189
4.2.16	RdbRowUpdatingEventHandler Delegate	192
4.2.17	RdbTransaction Class	193
4.2.18	RdbConnectionStringBuilder Class	199
4.2.19	RdbFactory Class	207
4.3	<i>Oracle Rdb Data Provider Enumerations</i>	210
4.3.1	RdbCommandTypes Enumeration	210

Chapter 5	Oracle Rdb Schema Collections	211
5.1	<i>Common Schema Collections</i>	<i>211</i>
5.1.1	MetaDataCollections.....	211
5.1.2	DataSourceInformation.....	211
5.1.3	DataTypes	212
5.1.4	Restrictions	214
5.1.5	ReservedWords	214
5.2	<i>ORDP.NET-Specific Schema Collection</i>	<i>215</i>
5.2.1	Tables	215
5.2.2	Columns	215
5.2.3	Views	216
5.2.4	Synonyms.....	216
5.2.5	Sequences.....	216
5.2.6	Functions.....	216
5.2.7	Procedures.....	217
5.2.8	ProcedureParameters.....	217
5.2.9	Indexes	217
5.2.10	IndexColumns.....	217
5.2.11	PrimaryKeys	218
5.2.12	PrimaryKeyColumns.....	218
5.2.13	ForeignKeys.....	218
5.2.14	ForeignKeyColumns	218
5.2.15	UniqueKeys.....	219
5.2.16	UniqueKeyColumns.....	219
5.2.17	Domains	219
5.2.18	Outlines	220
5.2.19	Constraints	220

[Glossary](#)

Send Us Your Comments

Oracle Rdb Data Provider for .NET Developer's Guide, Release 7.3.4.0

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

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

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

- Electronic mail: nedc-doc_us@oracle.com
- FAX — 603-897-3825 Attn: Oracle Rdb
- Postal service:
Oracle Corporation
Oracle Rdb Documentation
One Oracle Drive
Nashua, NH 03062-2804
USA

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

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

Preface

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

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

This preface contains these topics:

- Audience
- Access to Oracle Support
- Organization
- Related Documentation
- Conventions

Audience

Oracle Rdb Data Provider for .NET Developer's Guide is intended for developers who are developing applications to access an Oracle Rdb database using Oracle Rdb Data Provider for .NET. This documentation is also valuable to systems analysts, project managers, and others interested in the development of database applications.

To use this document, you must be familiar with Microsoft .NET Framework classes and ADO.NET and have a working knowledge of application programming using Microsoft C#, Visual Basic, or C++.

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

Access to Oracle Support

Oracle customers 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.

Organization

This document contains:

- [**Chapter 1, "Introducing Oracle Rdb Data Provider for .NET"**](#)
Provides an overview of Oracle Rdb Data Provider for .NET.
- [**Chapter 2, "Installing and Configuring"**](#)
Describes how to install Oracle Rdb Data Provider for .NET and provides system requirements.
Read this chapter *before* installing or using Oracle Rdb Data Provider for .NET.
- [**Chapter 3, "Features of Oracle Rdb Data Provider for .NET"**](#)
Describes provider-specific features of Oracle Rdb Data Provider for .NET.
- [**Chapter 4, "Oracle.DataAccess.RdbClient Namespace"**](#)
Describes the classes and public methods Oracle Rdb Data Provider for .NET exposes for ADO.NET programmers.
- [**Chapter 5, "Oracle Rdb Schema Collections"**](#)
Describes the schema collections Oracle Rdb Data Provider for .NET exposes for ADO.NET programmers using `RdbConnection.GetSchema`.
- [**Glossary**](#)
Defines terms used in this document.

Related Documentation

For more information, see these Oracle Rdb resources:

- *Oracle Rdb7 Guide to Database Design and Definition*
- *Oracle Rdb7 Guide to Database Performance and Tuning*
- *Oracle Rdb Introduction to SQL*
- *Oracle Rdb 7.2 SQL Reference Manual*
- *Oracle Rdb Guide to SQL Programming*
- *Oracle SQL/Services Server Configuration Guide*
- *Guide to Using the Oracle Rdb Oracle SQL/Services (tm) Client API*
- *Oracle Rdb JDBC Driver Users Guide*

To download free release notes, installation documentation, white papers, or other collateral, please visit the Oracle Rdb web site:

<http://www.oracle.com/technetwork/database/rdb>

For additional information, see:

<http://msdn.microsoft.com/netframework>

Conventions

Oracle Rdb Data Provider for .NET is often referred to as ORDP.NET.

Oracle Rdb is often referred to as Rdb.

Hewlett-Packard Company is often referred to as HP.

The following conventions are used in this document:

word	A lowercase word in a format example indicates a syntax element that you supply.
[]	Brackets enclose optional clauses from which you can choose one or none.
{ }	Braces enclose clauses from which you must choose one alternative.
...	A horizontal ellipsis means you can repeat the previous item
• • •	A vertical ellipsis in an example means that information not directly related to the example has been omitted.

Conventions in Code Examples

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

```
SELECT last_name FROM employees WHERE last_name = 'TOLIVER';
```

Chapter 1 Introducing Oracle Rdb Data Provider for .NET

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

This chapter contains these topics:

- Overview of Oracle Rdb Data Provider for .NET
- ORDP.NET Assembly
- Using ORDP.NET in a Simple Application

1.1 Overview of Oracle Rdb Data Provider for .NET (ORDP.NET)

ORDP.NET uses Oracle Rdb native APIs to offer fast and reliable access to Oracle Rdb data and features from any .NET application. ORDP.NET also uses and inherits classes and interfaces available in the [Microsoft .NET Framework Class Library](#).

1.2 ORDP.NET Assembly

`Oracle.DataAccess.Rdb.dll` assembly provides the `Oracle.DataAccess.RdbClient` namespace that contains ORDP.NET classes.

1.3 Oracle.DataAccess.RdbClient Classes

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

[Table 1-1](#) lists the client classes:

Table 1-1 Oracle.DataAccess.RdbClient Classes

Class	Description.
RdbCommand Class	An <code>RdbCommand</code> object represents a SQL command, a stored procedure, or a table name
RdbCommandBuilder Class	An <code>RdbCommandBuilder</code> object provides automatic SQL generation for the <code>RdbDataAdapter</code> when updates are made to the database
RdbConnection Class	An <code>RdbConnection</code> object represents a connection to an Rdb database
RdbConnectionStringBuilder Class	The <code>RdbConnectionStringBuilder</code> class allows ORDP.NET specific connections strings to be created easily.
RdbDataAdapter Class	An <code>RdbDataAdapter</code> object represents a data provider object that communicates with the <code>DataSet</code>
RdbDataReader Class	An <code>RdbDataReader</code> object represents a forward-only, read-only, in-memory result set
RdbError Class	The <code>RdbError</code> object represents an error reported by an Rdb database
RdbErrorCollection Class	An <code>RdbErrorCollection</code> object represents a collection of <code>RdbErrors</code>
RdbException Class	The <code>RdbException</code> object represents an exception that is thrown when Rdb Data Provider for .NET encounters an error
RdbFactory Class	The <code>RdbFactory</code> class represents a set of methods for creating instances of the Rdb Data Provider's implementation of the data source classes

RdbInfoMessageEventHandler Delegate	The <code>RdbInfoMessageEventHandler</code> delegate represents the signature of the method that handles the <code>RdbConnection.InfoMessage</code> event
RdbInfoMessageEventArgs Class	The <code>RdbInfoMessageEventArgs</code> object provides event data for the <code>RdbConnection.InfoMessage</code> event
RdbParameter Class	An <code>RdbParameter</code> object represents a parameter for an <code>RdbCommand</code>
RdbParameterCollection Class	An <code>RdbParameterCollection</code> object represents a collection of <code>RdbParameters</code>
RdbRowUpdatedEventArgs Class	The <code>RdbRowUpdatedEventArgs</code> object provides event data for the <code>RdbDataAdapter.RowUpdated</code> event
RdbRowUpdatedEventHandler Delegate	The <code>RdbRowUpdatedEventHandler</code> delegate represents the signature of the method that handles the <code>RdbDataAdapter.RowUpdated</code> event
RdbRowUpdatingEventArgs Class	The <code>RdbRowUpdatingEventArgs</code> object provides event data for the <code>RdbDataAdapter.RowUpdating</code> event
RdbRowUpdatingEventHandler Delegate	The <code>RdbRowUpdatingEventHandler</code> delegate represents the signature of the method that handles the <code>RdbDataAdapter.RowUpdating</code> event
RdbTransaction Class	An <code>RdbTransaction</code> object represents a local transaction

1.4 Oracle.DataAccess.RdbClient Enumerations

This namespace is the Oracle Rdb Data Provider for .NET.

[Table 1-2](#) lists the client enumerations:

Table 1-2 Oracle.DataAccess.RdbClient Enumerations

Class	Description.
RdbCommandTypes Enumeration	<code>RdbCommandTypes</code> enumerated values are used to specify the extended <code>CommandTypes</code> recognized by ORDP.NET.

1.5 Using ORDP.NET in a Simple Application

The following is a very simple C# application that connects to an Oracle Rdb database using an underlying SQL/Services connection and displays its version number before disconnecting.

```
// C#
using System;
using Oracle.DataAccess.RdbClient;
class Example
{
    RdbConnection conn;
    void Connect()
    {
        conn = new RdbConnection();
        conn.ConnectionString = "User Id=rdb_user;Password=rdb_pw;" +
            "Server=MYNODE:MY_SRV;Database=MY_DBS:MF_PERSONNEL;";
        conn.Open();
        Console.WriteLine("Connected to Rdb" + conn.ServerVersion);
    }
    void Close()
    {
    }
}
```

```
{
    conn.Close();
    conn.Dispose();
}

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

Chapter 2 Installing and Configuring

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

This chapter contains these topics:

- [System Requirements](#)
- [Installing Oracle Rdb Data Provider for .NET](#)
- [File Locations](#)
- [Post Installation Procedures](#)

2.1 System Requirements

Please see the Oracle Rdb Data Provider for .NET release notes for details.

2.2 Installing Oracle Rdb Data Provider for .NET

ORDP.NET is now installed as part of Oracle Rdb Developer's Tools for Visual Studio (ORDT)

Please see the Oracle Rdb Developer's Tools for Visual Studio and the Oracle Rdb Data Provider for .NET release notes for details.

2.3 File Locations

Please see the Oracle Rdb Data Provider for .NET release notes for details.

2.4 Post Installation Procedures

Please see the Oracle Rdb Data Provider for .NET release notes for details.

Chapter 3 Features of Oracle Rdb Data Provider for .NET

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

This chapter contains these topics:

- [Connecting to the Oracle Rdb Database](#)
- [Controlling the Number of Rows Fetched in One Server Round-Trip](#)
- [Transaction](#)
- [Guaranteeing Uniqueness in Updating DataSet to Database](#)
- [Data Provider Pattern in .NET V2.0](#)
- [External Procedures](#)
- [Debug Tracing](#)

3.1 Connecting to an Oracle Rdb Database

ORDP.NET will accept connections to Oracle Rdb Database using either SQL/Services services or an Oracle JDBC for Rdb Server.

The following sections describe:

- [Connection String Attributes](#)
- [SQL/Services Service connections](#)
- [JDBC Server connections](#)
- [Connection Pooling](#)
- [Connection Pool Management](#)

3.1.1 Connection String Attributes

The connection string provides the necessary information for ORDP.NET to determine the type of server to use, the node and other connection criteria.

Details about the connection string may be found in the [ConnectionString](#) property section of RdbConnection, in particular, [Table 4–17](#) lists the supported connection string attributes.

In order for ORDP.NET to correctly connect to an Oracle Rdb database, the type of connection required must be determined. The connection string attribute `Type` is used to specify the type of connection to use.

[Table 3–1](#) lists the supported connection types.

Table 3-1 Supported Connection Types

Connection Type	Description
POOLEDSQLS	Specifies that ORDP.NET should use SQL/Services to connect to the database as a pooled connection. The rest of the connection attributes should specify the SQL/Services service information needed to make a successful connection. See SQL/Services Service connections for more information on SQL/Services connections used in connect statement FILENAME parameter passed to SQL/Services. See Connection Pooling for more information about connection pooling.
POOLEDTHIN	Specifies that an Oracle JDBC for Rdb Server should be used to make the pooled connection to the database. The rest of the connection attributes

Connection Type	Description
	should specify the JDBC server information needed to make a successful connection. See JDBC Server connections for more information on this type of connection See Connection Pooling for more information about connection pooling.
SQS	Which is also the default value if the <code>Type</code> attribute is not specified, tells ORDP.NET to use SQL/Services to connect to the database. The rest of the connection attributes should specify the SQL/Services service information needed to make a successful connection. See SQL/Services Service connections for more information on SQL/Services connections used in connect statement <code>FILENAME</code> parameter passed to SQL/Services.
THIN	Specifies that an Oracle JDBC for Rdb Server should be used to make the connection to the database. The rest of the connection attributes should specify the JDBC server information needed to make a successful connection. See JDBC Server connections for more information on this type of connection.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
- [RdbConnection Members](#)
- [RdbConnection Constructors](#)
- [RdbConnection Static Methods](#)
- [RdbConnection Properties](#)
- [RdbConnection Public Methods](#)
- [RdbConnection Events](#)
- [RdbConnection Event Delegates](#)

3.1.2 SQL/Services Service connections

If Oracle SQL/Services is installed on the database server, connections to Oracle Rdb databases on that server may be made using SQL/Services service connections.

ORDP.NET uses the connection string attributes to create a session for use with the standard SQL/Services API. See your SQL/Services documentation on how to configure an SQL/Services Service for use by external applications as well as information on database specification and authorization.

Note:

ORDP.NET supports the use of both universal and database services within SQL/Services. If a universal service is used, the connection string must contain a database attribute with a valid and accessible database file specification. The service must use "SQLSERVICES" protocol.

ORDP.NET does not support the use of transaction reusable database services.

During installation ORDP.NET will copy a template `SQSAPI32.INI` file to the ORDP.NET installation directory.

This template may be modified to suit your SQL/Services configuration, but it must be copied to your system directory, as specified in your SQL/Services documentation, before the settings specified will take effect.

When used in conjunction with the SQS or the POOLEDSQLS Connection Types, the connection string attribute `Server` has the following format:

<server node>:<service name>

Where

- <server node> is a valid TCP/IP node specification
- <service name> is the name of a valid running SQL/Services Service on the specified node.

[Table 3-2](#) lists the relationship between the connection string attributes and the SQL/Services API components.

Table 3-2 SQL/Services component relationship

Connection String Attribute	SQL/Services components
Database	Used for the connect statement <code>FILENAME</code> parameter passed to SQL/Services.
Password	Used for the password within the SQL/Services association and in conjunction with the <code>USING</code> parameter within the connect statement passed to SQL/Services.
Server	Used in the SQL/Services association specifying the node and service to use for the connection.
User Id	Used for the <code>user_name</code> within the SQL/Services association and in conjunction with the <code>USER</code> parameter within the connect statement passed to SQL/Services.

The following example uses connection string attributes to connect to an Oracle Rdb Database using an SQL/Services universal service:

Example

```
// C#
.
.
.
RdbConnection conn = new RdbConnection();
conn.ConnectionString =
    "Type=SQS;User Id=rdb_user;Password=rdb_pw;" +
    "Server=MYNODE:MY_SRV;Database=MY_DBS:MF_PERSONNEL;";
conn.Open();
.
.
.
```

See [ConnectionString](#) for more information on connection string attributes.

3.1.3 JDBC Server connections

If Oracle JDBC for Rdb is installed on the database server, connections to Oracle Rdb databases on that server may be made using a running JDBC server.

ORDP.NET uses the connection string attributes to create a session for use with the Oracle JDBC for Rdb server. See your Oracle JDBC for Rdb documentation on how to configure a server for remote client use.

When used in conjunction with the THIN or the POOLEDTHIN Connection Types the connection string attribute Server has the following format:

<server node>:<service port>

Where

- <server node> is a valid TCP/IP node specification
- <server port> is the port number used by the Oracle JDBC for Rdb server

[Table 3-3](#) lists the relationship between the connection string attributes and the Oracle JDBC for Rdb server connection.

Table 3-3 JDBC Server component relationship

Connection String Attribute	JDBC Server components
Database	Used for the connect statement FILENAME parameter passed to the JDBC server .
Password	Used for the connect statement USING parameter passed to the JDBC server.
Server	Used to establish the connection to the JDBC server on the specified server node and port.
User Id	Used for the connect statement USER parameter passed to the JDBC server.

The following example uses connection string attributes to connect to an Oracle Rdb Database using a JDBC server:

Example

```
// C#
.
.
.
RdbConnection conn = new RdbConnection();
conn.ConnectionString =
    "Type=THIN;User Id=rdb_user;Password=rdb_pw;" +
    "Server=MYNODE:1701;Database=MY_DBS:MF_PERSONNEL;";
conn.Open();
.
.
.
```

See [ConnectionString](#) for more information on connection string attributes.

3.1.4 Connection Pooling

ORDP.NET allows simple connection pooling that is available for use in your application.

Connection pooling is only available to threads within the same application environment. Two separate instances of an application directly accessing ORDP will have their own connection pools. These pools are independent of each other.

Currently a single pool cannot be shared across multiple application invocations.

It is most suitable for use in a middle-tier broker application that may make connections to the underlying database system on behalf of its clients. The broker may use ORDP connection pooling to help reduce resource use and improve connection performance.

ORDP.NET connection pooling is enabled by using the `ConnectionString` attribute `TYPE=POOLEDSQLS` or `TYPE=POOLEDTHIN`. A `POOLEDSQLS` connection has the same basic characteristics as an `SQLS` type connection except that any connections made will be pooled. Similarly, a `POOLEDTHIN` connection is a `THIN` connection that will be pooled.

Connection Pooling Example

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

```
// C#
using System;
RdbConnection conn = new RdbConnection();

class ConnectionPoolingSample
{
    static void Main()
    {
        conn.ConnectionString =
            "Type=POOLEDTHIN;User Id=rdb_user;Password=rdb_pw;" +
            "Server=MYNODE:1701;Database=MY_DBS:MF_PERSONNEL;";
        conn.Open();
        Console.WriteLine("Connection pool successfully created");

        // Close the RdbConnection object
        conn.Close();
        Console.WriteLine("Connection is placed back into the pool.");
    }
}
```

3.1.4.1 Using Connection Pooling

When connection pooling is enabled, the `Open` and `Close` methods of the `RdbConnection` 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 following attribute values from the `ConnectionString` property as a signature, to uniquely identify a pool:

- User Id
- Password
- Server
- Database

If there is no existing pool with the exact attribute values as the `ConnectionString` property, the connection pooling service creates a new connection pool. If a pool already exists with the requested signature, a connection is returned to the application from that pool. If there is no free connection in the pool at the time of the connection request, a new connection will be established.

When the application closes a connection, the connection pooling service determines whether or not the connection lifetime has exceeded the value of the `PoolConnectionLifetime` attribute. If so, the connection pooling service closes the connection; otherwise, the connection may go back to the connection pool. The connection pooling service enforces the connection lifetime only when a connection is going back to the connection pool and only if the number of free connections already in the pool is less than the value specified by the `PoolMinFree` attribute.

In addition to the connection lifetime check, several other conditions must be satisfied before a connection will be returned into the connection pool. All of the following conditions must be true for a connection to be returned into the connection pool:

1. The connection lifetime has not expired or the number of free connections already in the pool is less than the value specified by the `PoolMinFree` attribute.
2. The connection is still actively connected to the database.
3. The number of free connections already in the pool is less than the value specified by the `PoolMaxFree` attribute.

If any condition is not met, the connection will be closed and will be no longer available as a pooled connection.

3.1.4.2 Connection Viability Check

When using Pooled connections, while a pooled connection is sitting in the free connection pool, it is possible that the underlying server, service or database may become unavailable, for example, if a connection is using a SQL/Services Service that has been restarted since the time that the pooled connection was placed in the free queue.

By default, ORDP will not check to see if the connection is viable at the time that it releases it from the free pool to be used by the new connection request. Connection viability operations cost network resources and increase the amount of time taken to do the connection.

If the connection is no longer viable, no exception will be raised on the open of the POOLED connection, however subsequent database operation will fail with either a network, socket or database exception.

You may choose to request ORDP to do a connection viability check prior to it being released from the free connection pool.

To tell ORDP to carry out connection viability checks during pooled connection operations, you must specify the following `ConnectionString` property attribute:

```
PoolValidateConnection=true;
```

For example:

```
string cnxString =  
@"Server=MYNODE.ME.COM:GENERIC;Database=user1:[murray]mf_personnel;User  
Id=murray;Pwd=secret;Type=POOLEDSQS;PoolValidateConnection=true";
```

To check connection viability, ORDP will try to connect to the database using the specified connectivity and carry out a simple operation. If the operation fails, ORDP will assume that the connection is no longer available and remove it from the free queue. It will then choose another free connection or create a fresh one.

You may also specify connectivity viability checks when using explicit connection pools.

This feature is available in both SQS and THIN connectivity and is only applicable to POOLEDQS and POOLEDTHIN connection types

Note:

The `PoolValidateConnection` attribute validates connections coming out of the pool. This attribute should be used only when absolutely necessary, because it uses 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.

3.1.5 Connection Pool Management

Connection pools within ORDP.NET may be created implicitly or explicitly. When ORDP.NET tries to open a connection that has the `ConnectionString` property `Type` attribute of POOLEDQS or POOLEDTHIN and no pool has been established that matches the connection string attributes, an implicit connection pool will be created.

Alternatively, a developer may create an explicit connection pool by opening a [Pool Manager](#) connection .

3.1.5.1 Implicit connection pools

Implicit connection pools are created when needed when ORDP.NET opens one of the pooled type of connections and no connection pool currently exists matching the `ConnectionString` attributes.

By default, an implicit pool has the following restrictions placed on it:

- the number of free connections that may be maintained will be set to 10. This is the same as specifying the attribute `PoolMaxFree=10`.
- the total number of connections, free or in use, that may be maintained is unlimited. This is the same as specifying the attribute `PoolMaxSize=0`.
- the minimum number of free connections kept at anytime is 0. This is the same as specifying the attribute `PoolMinFree=0`.
- connections taken from the pool have an unlimited `Connection Lifetime`. This is the same as specifying the attribute `PoolConnectionLifetime=0`.
- if no free connection is available when a connection request is made, a new connection will be established immediately. This is the same as specifying the attribute `PoolConnectionTimeout=0`.
- No connection viability check is done prior to releasing a free pooled connection for use.

See [ConnectionPoolExample](#) for an example of creating an implicit connection pool.

Developers may choose to establish different default pool criteria to the values described above, the following section shows how to do this.

3.1.5.2 Establishing Default Pool Characteristics

Developers may change the default pool characteristics by either:

- Using the [RdbConnection.EstablishPoolCriteria](#) method, or
- by establishing a [Pool Manager](#) connection with no server specified.

This will establish the default values for the various pool attributes that will be used whenever an implicit pool is created.

The following example shows how you may use `EstablishPoolCriteria` to set the default pool characteristics:

Example

```
// C#
.
.
.
string conStr =
    @"Type=POOLEDSQLS;Server=node1.oracle.com:GENSRVC;
    Database=mydb;User Id=myname;Password=mypassword;";

int maxPool = 10; // only allow 10 concurrent connections in any pool
int maxFree = 5; // maintain a maximum of 5 free connections in any pool
int minFree = 3; // maintain a minimum of 3 free connections in any pool
int connLife = 0; // allow free connections to exist indefinitely
int cleanerPeriod = 0; // don't run cleaner thread
int connTimeout = 0; // raise an exception immediately if maxPool
    // exceeded and we tried to get a new connection
bool validateServer = false; // true would mean an extra network i/o on
    // the connection being returned from pool

RdbConnection.EstablishPoolCriteria(maxPool, maxFree, minFree,
    connLife, cleanerPeriod, connTimeout, validateServer);

// pools created from now on will take the defaults as above
RdbConnection conn = new RdbConnection();

conn.ConnectionString =
    @"Type=POOLEDTHIN;User Id=rdb_user;Password=rdb_pw;
    Server=MYNODE:1701;Database=MY_DBS:MF_PERSONNEL;";
conn.Open();
Console.WriteLine("Connection pool successfully created");

    // Close the RdbConnection object
conn.Close();
Console.WriteLine("Connection is placed back into the pool.");

// open the connection and as this is the first one it will establish the
// connection pool with limits as specified above

conn.Open();
// the close will return the connection to the connection pool
```

```
// which will have one free connection now
connClose();
.
.
.
```

Alternatively, developers may change the default pool characteristics by establishing a [Pool Manager](#) connection with *no* server specified. This control connection will establish the default values for the various pool attributes that will be used whenever an implicit pool is created:

Example

```
// C#

using System;
RdbConnection conn = new RdbConnection();
RdbConnection poolMan = new RdbConnection();

class ConnectionPoolingSample2
{
    static void Main()
    {
        // no server specification in ConnectionString but poolmanager=true
        // indicates this is a control connection for the connection pooling
        // service management

        poolMan.ConnectionString =
@"poolmanager=true;poolminfree=10;poolmaxsize=40;
    poolmaxfree=20;poolconnectiontimeout=5000;
    poolcleanerperiod=60000";

        poolMan.Open(); // establishes the defaults
        poolMan.Close(); // don't need this connection any more

        // pools created from now on will take the defaults as above
        conn.ConnectionString =
@"Type=POOLEDTHIN;User Id=rdb_user;Password=rdb_pw;
    Server=MYNODE:1701;Database=MY_DBS:MF_PERSONNEL;";
        conn.Open();
        Console.WriteLine("Connection pool successfully created");

        // Close the RdbConnection object
        conn.Close();
        Console.WriteLine("Connection is placed back into the pool.");
    }
}
```

3.1.5.3 Explicit connection pools

ORDP.NET connection pool management provides explicit connection pool control to ORDP.NET applications. An explicit connection pool may be created by establishing a [Pool Manager](#) and providing the pool requirements as attributes to the Pool Manager ConnectionString.

[Table 3-4](#) lists the connection string attributes used when establishing a Pool Manager.

If a Server attribute is provided for the Pool Manager ConnectionString property, when the Pool Manager connection is opened, the connection pooling service creates a connection pool by using

the following attribute values from the `ConnectionString` property as a signature, to uniquely identify the pool:

- User Id
- Password
- Server
- Database

Example

```
// C#

using System;
RdbConnection conn = new RdbConnection();
RdbConnection poolMan = new RdbConnection();

class ConnectionPoolingSample3
{
    static void Main()
    {
        // Server specification in the ConnectionString and poolmanager=true
        // indicates this is an explicit connection pool definition

        poolMan.ConnectionString =
        @"Type=POOLEDTHIN;User Id=rdb_user;Password=rdb_pw;
        Server=MYNODE:1701;Database=MY_DBS:MF_PERSONNEL;
        poolmanager=true;poolminfree=10;poolmaxsize=40;
        poolmaxfree=20;poolconnectiontimeout=5000;
        poolcleanerperiod=60000";

        poolMan.Open(); // establishes the pool explicitly.
        Console.WriteLine("Connection pool successfully created");

        poolMan.Close(); // don't need this connection any more but
                        // pool will stay around

        // new connections using the same ConnectionString attributes
        // will use the pool declared above
        conn.ConnectionString =
        @"Type=POOLEDTHIN;User Id=rdb_user;Password=rdb_pw;
        Server=MYNODE:1701;Database=MY_DBS:MF_PERSONNEL;";
        conn.Open();

        // Close the RdbConnection object
        conn.Close();
        Console.WriteLine("Connection is placed back into the pool.");
    }
}
```

3.1.5.4 Establishing a Pool Manager

To allow greater control of connection pooling ORDP.NET allows the creation of special `RdbConnections` called `Pool Managers` that may be used to establish the default conditions of implicit pools or to establish and maintain the conditions of individual explicit connection pools.

A Pool Manager connection is identified by the `PoolManager` attribute of the `ConnectionString` property being set to `True`.

If the `PoolManager ConnectionString` property specifies a `Server` attribute, calling the `Open()` method for this connection will cause an explicit connection pool to be created matching the `PoolManager ConnectionString` attributes.

If the `ConnectionString` property for the Pool Manager connection does not specify a `Server` attribute, calling the `Open()` method for this connection will cause the connection pooling service to use the `PoolManager ConnectionString` attributes as the default values for subsequent pool creations. No connection pool will be created during this connection open.

[Table 3-4](#) lists the connection string attributes used when establishing a Pool Manager.

Table 3-4 Pool Manager ConnectionString Attributes

Connection String Attribute	Default value	Description
Database or Data Source	empty string	Identifies the database associated with the connection. If null or an empty string, the default database for the specified server will be used. Used to identify the connection pool.
Password or Pwd	empty string	Password for the user specified by <code>User Id</code> . This attribute specifies an Rdb user's password. Password is case insensitive. Used to identify the connection pool.
PoolConnectionLifetime	0	The amount of time in seconds the connection pooling service should allow a free connection to live. This is only checked when a connection is returned to the pool.
PoolConnectionTimeout	0	The amount of time in seconds the connection pooling service should wait for a free connection when <code>PoolMaxSize</code> connections have been made. A value of 0 means return an exception immediately. A value of -1 means wait indefinitely.
PoolManager	False	If <code>True</code> , identifies this connection as a Pool Manager connection.
PoolMaxFree	0	Establishes the maximum number of free connections maintained by the pool. A value of 0 indicates that there should be no limit to the number of free connections.
PoolMinFree	0	Establishes the minimum number of free connections maintained by the pool. On pool creation, this will be the number of initial free connections that will be placed into the pool.
PoolValidateConnection	false	If true, validate the connection when taken from the pool. Validation of a connection requires a network round-trip to the server.
Server	empty string	Identifies the server to use for the connection. If null or empty, a control Pool Manager connection will be established. If the <code>Server</code> attribute is not null and not empty, an explicit pool will be created.
Type	POOLEDSQS	If <code>Type</code> is specified and is "POOLEDTHIN", the <code>Server</code> must be a valid Oracle JDBC for Rdb connection URL. If <code>Type</code> is not specified or is "POOLEDSQS" the <code>Server</code> must be a valid Oracle SQL/Services for Rdb connection specification. Used to identify a connection pool. Specifies the type of the Server connection. Valid types are:

Connection String Attribute	Default value	Description
		<ul style="list-style-type: none"> • POOLEDSQLS - make a pooled Oracle SQL/Services connection • POOLEDTHIN - make a pooled connection to an Oracle JDBC for Rdb Server
		If not specified or an empty string is specified, the default type will be used.
User Id or User or Username	empty string	This attribute specifies the Rdb user name. Used to identify a connection pool.

Other connection attributes may also be present in the connection string. These must be valid connection string attributes as shown in [Supported Connection String Attributes](#) and will be used during the creation of any initial connections required to establish the `PoolMinFree` number of free connections for the pool.

When a `Pool Manager` connection is opened and a `Server` attribute is present in the `ConnectionString`, the connection pooling service creates a connection pool by using the following attribute values from the `ConnectionString` property as a signature, to uniquely identify a pool:

- User Id
- Password
- Server
- Database

When a connection pool is created, the connection pooling service initially creates the number of connections defined by the `PoolMinFree` attribute of the `ConnectionString` property. This number of connections is always maintained by the connection pooling service for the connection pool except when the pool is cleared by a call to the `RdbConnection.ClearAllPools` method.

The `PoolMaxFree` attribute of the `ConnectionString` property sets the maximum number of free connections for a connection pool. This limit is checked when a connection is released back into the free pool. If the number of free connections in the pool is equal to or exceeds the `PoolMaxFree`, the connection will be closed and not returned to the pool.

The `PoolMaxSize` 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 `PoolMaxSize` has been reached, the connection pooling service will wait for the time defined by the `PoolConnectionTimeout` attribute.

If the `PoolConnectionTimeout` 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. If the `PoolConnectionTimeout` is `-1`, the connection pooling service will wait indefinitely for a connection.

The `PoolConnectionLifetime` attribute of the `ConnectionString` property sets the maximum number of seconds that a pooled connection can live. If the connections lifetime has exceeded the value of the `PoolConnectionLifetime` attribute, the connection pooling service closes the connection; otherwise, the connection goes back to the connection pool. The connection pooling service enforces the connection lifetime only when a connection is going back to the

connection pool. A `PoolConnectionLifetime` of 0 means that the connection has no lifetime restriction.

The `PoolValidateConnection` 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.

Note:

A `PoolManager` connection cannot be used in the same manner as a normal `RdbConnection` as no database connection is actually made within the `PoolManager` connection context. Attempting an operation requiring a database connection such as executing a query will result in an exception.

3.2 ADO.NET 2.0 Features

Oracle Rdb Data Provider for .NET 7.3.0.2 or later supports Microsoft ADO.NET 2.0 APIs. This section contains the following topics:

- [About ADO.NET 2.0](#)
- [Base Classes and Provider Factory Classes](#)
- [Connection String Builder](#)
- [Support for Schema Discovery](#)
- [User Customization of Metadata](#)

3.2.1 About ADO.NET 2.0

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

Note:

Using ORDP.NET with Microsoft ADO.NET 2.0 requires ADO.NET 2.0- compliant ORDP.NET.

See Also:

ADO.NET in the MSDN Library

3.2.2 Base Classes and Provider Factory Classes

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

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

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

Table 3-3 ORDP.NET Classes that Inherit from ADO.NET 2.0 Base Classes

ORDP.NET Classes	Inherited from ADO.NET 2.0 Base Class
<code>RdbFactory</code>	<code>DbProviderFactory</code>
<code>RdbCommand</code>	<code>DbCommand</code>
<code>RdbCommandBuilder</code>	<code>DbCommandBuilder</code>
<code>RdbConnection</code>	<code>DbConnection</code>
<code>RdbConnectionStringBuilder</code>	<code>DbConnectionStringBuilder</code>
<code>RdbDataAdapter</code>	<code>DbDataAdapter</code>
<code>RdbDataReader</code>	<code>DbDataReader</code>
<code>RdbException</code>	<code>DbException</code>
<code>RdbParameter</code>	<code>DbParameter</code>
<code>RdbParameterCollection</code>	<code>DbParameterCollection</code>
<code>RdbTransaction</code>	<code>DbTransaction</code>

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

See Also:

- [RdbFactory Class](#)
-

3.2.3 Connection String Builder

The `RdbConnectionStringBuilder` 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 through the key/value pairs. One example of a configuration file entry follows:

```
<configuration>
  <connectionStrings>
    <add name="Pers" providerName="Oracle.DataAccess.RdbClient"
      connectionString="Server=node1.mycom.com:1701;Database=mf_personnel" />
  </connectionStrings>
</configuration>
```

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

See Also:

- [Data Provider Pattern in .NET V2.0](#)
-

3.2.4 Support for Schema Discovery

ADO.NET 2.0 exposes five different types of metadata collections through the `RdbConnection.GetSchema` API. This permits application developers to customize metadata retrieval on an individual-application basis, for any Oracle Rdb 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 Rdb SQL query language.

In addition to these standard collections, ORDP.NET also exposes Oracle Rdb-specific collections.

See Also:

- [Rdb Schema Collections](#)
-

3.2.5 User Customization of Metadata

ORDP.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 does not have to contain the entire set of schema configuration information, any collection not found in this customized metadata file will be searched for within the ORDP.NET imbedded metadata file. Developers provide changes that modify the behavior of the schema

retrieval to user-specific requirements. For instance, a developer can filter out database tables they wish to be remain hidden from the `GetSchema` method.

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.rdbclient>
  <settings>
    <add name="MetaDataXml" value="CustomORDPMetaData.xml" />
  </settings>
</oracle.dataaccess.rdbclient>
```

When the `GetSchema` method is called, ORDP.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.rdbclient`. 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 ORDP.NET.

See Also:

- [GetSchema](#)
-

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

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

3.3.1 Use of `FetchSize`

The `FetchSize` property represents the number of rows that ORDP.NET allocates to cache the data fetched from a server round-trip.

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

If the `FetchSize` property is set on the `RdbCommand`, then the newly created `RdbDataReader` inherits the `FetchSize` property of the `RdbCommand`.

This inherited `FetchSize` can be left as is or modified to override the inherited value. The `FetchSize` property of the `RdbDataReader` object can be changed before the first `Read` method invocation, which allocates memory specified by the `FetchSize`. All subsequent fetches from the database use the same cache allocated for that `RdbDataReader`. Therefore, changing the `FetchSize` after the first `Read` method invocation has no effect.

3.3.2 Fine-Tuning `FetchSize`

By fine-tuning the `FetchSize` property, applications can control memory usage and the number of rows fetched in one server round-trip for better performance.

For example, if a query returns 100 rows, then setting `FetchSize` to 100 takes just one server round-trip to fetch the hundred rows.

For the same query, if the `FetchSize` is set to 10, it takes 10 server round-trips to retrieve 100 rows. If the application requires all the rows to be fetched from the result set, the first scenario is faster than the second. However, if the application requires just the first 10 rows from the result set, the second scenario can perform better since it only fetches 10 rows and not 100 rows.

3.3.3 Setting FetchSize Value at Design Time

If the row size for a particular `SELECT` statement is already known from a previous execution, `FetchSize` of the `RdbCommand` can be set at design time to the number of rows the application wishes to fetch for each server round-trip. The `FetchSize` value set on the `RdbCommand` object is inherited by the `RdbDataReader` that is created by the `ExecuteReader` method invocation on the `RdbCommand`. Rather than setting the `FetchSize` on the `RdbCommand`, the `FetchSize` can also be set on the `RdbDataReader` directly.

3.4 Transaction

Transactions may be implicit or explicit.

An implicit transaction is one started for you by ORDP.NET that will be automatically committed at the end of the next executable SQL statement sent down to database system.

An explicit transaction is one created for you when you invoke the [RdbConnection.BeginTransaction](#) method. The returned [RdbTransaction](#) object maintains the context of the transaction within the underlying database.

Explicit transactions must be explicitly committed, explicitly rolled back or disposed. On disposal of an `RdbTransaction` object, an active transaction will be implicitly rolled back.

In addition, in .NET V2.0 you may use `TransactionScope` to define the appropriate transaction boundaries for your operations.

The following sections describe:

- [Implicit Transactions](#)
- [Explicit Transactions](#)
- [TransactionScope](#)

3.4.1 Implicit Transactions

When SQL statements are executed outside the scope of an explicit transaction, an appropriate transaction will be automatically started for you by ORDP.NET.

The type of transaction the ORDP.NET starts up when a transaction is required depends on

- The verb of the SQL statement to be executed
- Whether the connection has been set to `READ_ONLY`

If no specific behaviour has been specified, by default the ORDP.NET will start up a READ_WRITE SERIALIZABLE transaction if the SQL statement requires a read-write transaction, for example, INSERT or UPDATE. If the statement does not require a read-write transaction, a READ_ONLY transaction is started.

If the connection has been set READ_ONLY, ORDP.NET will always start READ_ONLY transactions.

The scope of the transaction is the next executable SQL statement. Once the statement has successfully completed, the transaction will be automatically committed. The execution of the next statement will start a new transaction.

3.4.2 Explicit Transactions

An explicit transaction can only be started in the context of a connection, that is, only by using the appropriate [BeginTransaction](#) method on an RdbConnection object.

Once an explicit transaction starts, all the successive command executions on that connection execute within the context of that transaction, until either the transaction is committed or a rollback is issued.

As well as the standard ability to specify the IsolationLevel when calling the BeginTransaction method, ORDP.NET also allows the use of Oracle Rdb specific transaction specification strings.

Example

```
// C#  
  
.  
.  
.  
conn.Open();  
RdbTransaction tx = conn.BeginTransaction(  
    "READ WRITE RESERVING CANDIDATES FOR EXCLUSIVE WRITE");  
.  
.  
.
```

See your Oracle Rdb documentation for information on transaction declarations.

See Also:

- [RdbConnection Class](#)
- [RdbTransaction Class](#)

3.4.3 TransactionScope

TransactionScope, introduced in .NET V2.0, allows a common transaction mechanism to scope the boundaries of a transaction. Usually used in conjunction with distributed transactions the TransactionScope object allows ease of programming transactions that may involve one or more connections.

Currently ORDP.NET does not support distributed transactions, however `TransactionScope` may still be used in the context of a single `RdbConnection`.

Example

```
// C#
.
.
.
conn.Open();
RdbCommand cmd = new RdbCommand(
    "insert into customers values (999,1091,'FRED')", conn);

try
{
    using (TransactionScope scope = new TransactionScope())
    {
        conn.EnlistTransaction();
        cmd.ExecuteNonQuery();
        scope.Complete();
    }
}
catch (System.Transactions.TransactionException ex)
{
    Console.WriteLine(ex);
}
.
.
.
```

3.5 Guaranteeing Uniqueness in Updating DataSet to Database

This section describes how the `RdbDataAdapter` configures the `PrimaryKey` and `Constraints` properties of the `DataTable` that guarantee uniqueness when the `RdbCommandBuilder` is updating `DataSet` changes to the database.

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

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

To guarantee that each `DataRow` change affects only a single row, there must be a set of `DataColumns` in the `DataTable` for which all rows in the `DataTable` have a unique set of values. The set of `DataColumns` indicated by the properties `DataTable.PrimaryKey` and `DataTable.Constraints` meet this requirement.

The `RdbCommandBuilder` determines uniqueness in the `DataTable` by checking whether the `DataTable.PrimaryKey` is non-null or if there exists a `UniqueConstraint` in the `DataTable.Constraints` collection.

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

This section includes the following topics:

- What Constitutes Uniqueness in `DataRows`?
- Configuring `PrimaryKey` and `Constraints` Properties
- Updating Without `PrimaryKey` and `Constraints` Configuration

3.5.1 What Constitutes Uniqueness in `DataRows`?

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

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

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

Note:

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

3.5.2 Configuring `PrimaryKey` and `Constraints` Properties

If the minimal conditions described in "[What Constitutes Uniqueness in `DataRows`?](#)" are met, then the `DataTable.PrimaryKey` or `DataTable.Constraints` properties can be set.

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

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

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

Criteria:

The set of columns has a unique constraint defined on it or a unique index created on it, with each column having a `NOT NULL` constraint defined on it.

3. If a set of columns that meets the following criteria is returned in the select list, a `UniqueConstraint` is added to the `DataTable.Constraints` collection, but the `DataTable.PrimaryKey` is not set.
Criteria:
 - The set of columns has a unique constraint defined on it or a unique index created on it, with at least one column having a `NOT NULL` constraint defined on it.
4. If a `ROWID` is part of the select list, it is set as the `DataTable.PrimaryKey`.
Additionally, `RdbDataAdapter.FillSchema` exhibits the following behaviors:
 - Setting `DataTable.PrimaryKey` implicitly creates a `UniqueConstraint`.
 - If there are multiple occurrences of a column in the select list and the column is also part of the `DataTable.PrimaryKey` or `UniqueConstraint`, or both, each occurrence of the column is present as part of the `DataTable.PrimaryKey` or `UniqueConstraint`, or both.

3.5.3 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 `RdbDataAdapter.FillSchema`, the `RdbCommandBuilder` directly checks the select list of the `RdbDataAdapter.SelectCommand` to determine if it guarantees uniqueness in the `DataTable`.

However this check results in a server round-trip to retrieve the metadata for the `SELECT` statement of the `RdbDataAdapter.SelectCommand`.

3.6 Data Provider Pattern in .NET V2.0

Introduced in .NET V2.0, the data provider pattern allows the generic coding of ADO.NET connections and operations.

The pattern consists of the following parts:

- A unique string is used to identify each subclass. ADO.NET 2.0 uses the namespace of the subclass as its unique string id
- A configuration file storing the required provider information
- A separate class named `DbProviderFactories` that exposes the `GetFactory` static method. The method takes the unique string id of the desired subclass as its only argument and searches through the `machine.config` file for a subclass with the given unique string id.
- Separate `RdbProviderFactory` classes that expose the data access methods required by the pattern

The following example shows how the generic data provider pattern may be used to access an `Rdb` database using `ORDP.NET`.

Example

```
// C#
.
.
.
String conStr =
```

```

@"User Id=rdb_user;Password=rdb_pw;
  Server=MYNODE:MY_SRV;Database=MY_DBS:MF_PERSONNEL";

DbProviderFactory f =
  DbProviderFactories.GetFactory("Oracle.DataAccess.RdbClient");

DbConnection c = f.CreateConnection();
c.ConnectionString = conStr;
c.Open();
DbCommand cmd = c.CreateCommand();

cmd.CommandText = "select employee_id,last_name,birthday from employees";

IDataReader reader = cmd.ExecuteReader();
while (reader.Read())
{
  Console.Write(reader.GetInt32(0) + "\t");
  Console.Write(reader.GetString(1) + "\t");
  Console.Write(reader.GetDateTime(2));
  Console.WriteLine();
}
reader.Close();
c.Close();
.
.
.

```

The `DbProviderFactories.GetFactory` method returns an [RdbFactory](#) object that may be used to obtain the appropriate data access methods that are available using an `RdbConnection`.

In the above example, the connection string is still not generic as its format is specific to ORDP.NET. This may be made more generic by using the `RdbConnectionStringBuilder` class as returned by the `GetConnectionStringBuilder` method of the `DbProviderFactory`.

Example

```

// C#
.
.
.
DbProviderFactory f =
  DbProviderFactories.GetFactory("Oracle.DataAccess.RdbClient");

DbConnectionStringBuilder sb = f.GetConnectionStringBuilder();
sb.Server = "MyNode:MySQLService";
sb.DataSource = "disk2:[dbs]personnel";
sb.UserId = "testUser";
sb.Password = "mypassword"
DbConnection c = f.CreateConnection();
c.ConnectionString = sb.ConnectionString;
.
.
.

```

The `ConnectionString` returned will be generated from the individual attributes provided to the `DbConnectionStringBuilder` object.

Alternatively, the `web.config` file now supports a new section named `<connectionStrings>` that contains all the connection strings used in an application, for example:

```

<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <connectionStrings>
    <add
      name="MyRdbConnectionString"
      connectionString="Server=MyNode:MyQSService;" +
        "Data Source=disk2:[dbs]personnel "
      providerName="Oracle.DataAccess.RdbClient" />
    </connectionStrings>
  </configuration>

```

The <add> subelement of the <connectionStrings> element exposes the following attributes:

- Name—The friendly name of the connection string
- connectionString—The actual connection string
- providerName—The unique string id of the code provider class

Used in conjunction with the ConnectionStringBuilder:

Example

```

// C#
.
.
.
DbProviderFactory f =
  DbProviderFactories.GetFactory("Oracle.DataAccess.RdbClient");

DbConnectionStringBuilder sb = f.GetConnectionStringBuilder();

Configuration configuration =
  Configuration.GetWebConfiguration("~/");

ConnectionStringsSection section =
  (ConnectionStringsSection)configuration.Sections["connectionStrings"];

sb.ConnectionString =
  section.ConnectionStrings["MyRdbConnectionString"].ConnectionString;

sb.UserId = "testUser";
sb.Password = "mypassword"
DbConnection c = f.CreateConnection();
c.ConnectionString = sb.ConnectionString;
.
.
.

```

For the call to the GetFactory method to work correctly, the ORDP.NET provider factory must be registered as a DbProviderFactory. See the following sections for more details.

The following sections describe:

- [Identification of the Oracle Rdb Data Provider for .NET](#)
- [The DbProviderFactory for ORDP.NET](#)
- [The DbConnectionStringBuilder for ORDP.NET](#)
- [Registration of the DbProviderFactory for ORDP.NET](#) using configuration files

3.6.1 Identification of the Oracle Rdb Data Provider for .NET

ORDP.NET is identified to the NET V2.0 data provider pattern using the following string:

```
"Oracle.DataAccess.RdbClient"
```

3.6.2 The DbDataProviderFactory for ORDP.NET

The new class called `RdbFactory` exposes the standard `DbDataProvider` methods for use when using the Data Provider pattern for .NET V2.0.

Specifically the following methods are exposed:

- `CreateConnection`
- `CreateCommand`
- `CreateConnectionStringBuilder`
- `CreateCommandBuilder`
- `CreateDataAdapter`
- `CreateParameter`

See Also:

- [RdbFactory Class](#)
-

3.6.3 The DbConnectionStringBuilder for ORDP.NET

The new class called `RdbConnectionStringBuilder` provides a standard way of building connection strings for ORDP.NET connections that comply with the Data Provider pattern for .NET V2.0.

See Also:

- [RdbConnectionStringBuilder Class](#)
-

3.6.4 Registration of the DbProviderFactory for ORDP.NET

The Data Provider pattern for .NET V2.0 uses the `machine.config` file to register `DbProviderFactories`.

The `machine.config` file contains settings that apply to the entire computer. There is only one `machine.config` file on a computer and may be found in the "CONFIG" subfolder of your .NET Framework install directory, for example

on Windows 2000 the directory would be:

```
{System Disk}:\WINNT\Microsoft.NET\Framework\{Version  
Number}\CONFIG
```

on Windows XP, Windows Vista and Windows 7, the directory would be:

```
{System Disk}:\WINDOWS\Microsoft.NET\Framework\{Version  
Number}\CONFIG
```

Beginning with release 7.3.2 of ORDP.NET, the ORDP.NET installation step carried out during the ORDT installation will update your system's `machine.config` file to add an entry for ORDP.NET into the `DbProviderFactories` section:

```
<configuration>  
  <system.data>  
    <DbProviderFactories>  
      <add name="Oracle Rdb Data Provider"  
        invariant="Oracle.DataAccess.RdbClient"  
        description=".Net Framework Data Provider for Oracle Rdb"  
        type="Oracle.DataAccess.RdbClient.RdbFactory,Oracle.DataAccess.Rdb,  
        Version=7.3.2.0, Culture=neutral,PublicKeyToken=24caf6849861f483"/>  
      .  
      .  
      .  
    </DbProviderFactories>  
  </system.data>  
</configuration>
```

In addition, the following configuration section type will be added to the `<configSections>` of your system's `machine.config`:

```
<configuration>  
  <configSections>  
    <section name="oracle.dataaccess.rdbclient"  
      type="System.Data.Common.DbProviderConfigurationHandler,  
      System.Data, Version=2.0.0.0, Culture=neutral,  
      PublicKeyToken=b77a5c561934e089"/>  
    .  
    .  
    .  
  </configSections>  
</configuration>
```

See Also:

- [RdbConnectionStringBuilder Class](#)
-

3.7 External Procedures

ORDP.NET allows access to Oracle Rdb External Procedures in much the same way as Stored Procedures.

`RdbCommand` has an Oracle Rdb specific command type, `RdbCommandType.ExternalProcedure`, that may be used to designate that the text of the `RdbCommand` is the name of an Rdb External Procedure.

Once defined, the developer may use an `RdbCommand` with the `RdbCommandType` property value of `RdbCommandTypes.ExternalProcedure` in the same way as `RdbCommand` objects with the `CommandType` property value of `CommandType.StoredProcedure`.

The following example shows how to declare and use an Oracle Rdb External Procedure:

Example

This example assumes the following external procedure has been declared:

```
create procedure LOOKUP_KEY (
  in   :KEY_NAME VARCHAR (100) by descriptor,
  out  :RETURN_STRING VARCHAR (100) by descriptor,
  inout :RETURN_LENGTH SMALLINT by reference)

  language SQL;
  external
    name LOOKUP_MY_KEYS
    location 'MY_SHARES:KEYS.EXE'
    with ALL logical_name translation
    language GENERAL
    GENERAL parameter style
  comment is
    'Return the text value associated with a given key value'
```

```
// C#
.
.
.
RdbCommand cmd = conn.CreateCommand();
cmd.RdbCommandType = RdbCommandTypes.ExternalProcedure;
cmd.CommandText = "LOOKUP_KEY";
RdbCommandBuilder.DeriveParameters(cmd);
RdbParameterCollection coll = cmd.Parameters;
RdbParameter p1 = coll[0]; // in param so need input
RdbParameter p3 = coll[2]; // inout param needs input as well
p1.Value = "KEY1234";
p3.Value = 100;

Console.WriteLine(" proc contains " + coll.Count + " parameters");
for (int i = 0; i < coll.Count; i++)
{
  RdbParameter p2 = coll[i];
  Console.WriteLine(" param " + i + " " + p2.ParameterName);
  Console.WriteLine(" direction of param = " + p2.Direction);
}
IDataReader reader = cmd.ExecuteReader();
while (reader.Read())
{
  Console.WriteLine ("Return Length = " + reader.GetInt32(2) + "\t");
  Console.WriteLine ("Return Value = " + reader.GetString(1) + "\t");
}
reader.Close();
.
.
.
```

See Also:

- [RdbCommandType](#)
-

-
- [RdbCommandTypes Enumeration](#)
 - [RdbCommand Class](#)
-

3.8 SSL and THIN connectivity

ORDP.NET allows access to Oracle JDBC for Rdb servers over a secured SSL TCP/IP socket.

To connect to an Oracle JDBC for Rdb server using SSL, you must provide certain SSL information to the `RdbConnection`.

The simplest way to ask for a SSL style connection is to use the connection string attribute “`SSLMode`” set to the value “`Required`”:

Example

```
// C#  
  
using System;  
RdbConnection conn = new RdbConnection();  
  
class SSLConnectionSample  
{  
    static void Main()  
    {  
        conn.ConnectionString =  
            "Type=THIN;User Id=rdb_user;Password=rdb_pw;" +  
            "Server=MYNODE:1701;Database=MY_DBS:MF_PERSONNEL;" +  
            "SSLMode=Required;";  
  
        conn.Open();  
        Console.WriteLine("Connection successfully made");  
    }  
}
```

In this simplest form of SSL connection, ORDP.NET will try to connect to the server using a SSL secured socket. If the SSL JDBC server is running, ORDP.NET will accept any certificate sent to it without verification, and will then proceed to make the secured connection.

If the server is not using SSL, the connection request will be terminated immediately.

If you try to make a connection to a JDBC server that is using SSL, but fail to provide the `SSLMode` connection attribute, or specify a `SSLMode` attribute set to “`None`”, the connection will eventually time-out as the initial connect handshake will fail.

ORDP.NET allows you to specify various level of verification of certification being sent back from the SSL enabled JDBC server:

- `None` – do not use SSL .
- `VerifyFull` – validate server’s SSL certificate and only allow trusted servers. Deny connection if certificate name is mismatched.
- `VerifyCA` – validate server’s SSL certificate. Still allow connection even if certificate name is mismatched.
- `AllowUntrusted` – validate servers SSL certificate, and allow even if untrusted. Deny connection if certificate name is mismatched.

- `Required` – accept any certificate delivered back from the server. Deny connection if server does not support SSL.

See Also:

- [RdbConnection Properties](#)
-

3.8.1 CertName connection string attribute

Used in conjunction with the “SSLMode” attribute, the “CertName” connection string attribute provides the expected name of the certificate that the thin server will send in reply to the connection request.

Depending on the level of SSLMode requested, the name of the returned certificate may be checked against the name specified by the “CertName” attribute, and if they do not match, the connection may be denied.

The “CertName” attribute is checked when using the following SSLModes:

- [VerifyFull](#) – validate server’s SSL certificate and only allow trusted servers. Deny connection if name is mismatched.
- [AllowUntrusted](#) – validate servers SSL certificate, and allow even if untrusted. Deny connection if name is mismatched.

If the “CertName” connection string attribute is not set, or is set to a blank string, the value “RdbJdbcServer” will be used as the certificate name.

Example

```
// C#
using System;
RdbConnection conn = new RdbConnection();

class SSLConnectionSample
{
    static void Main()
    {
        conn.ConnectionString =
            "Type=THIN;User Id=rdb_user;Password=rdb_pw;" +
            "Server=MYNODE:1701;Database=MY_DB:MF_PERSONNEL;" +
            "SSLMode=AllowUntrusted;CertName=RdbJdbcServer;";

        conn.Open();
        Console.WriteLine("Connection successfully made");
    }
}
```

The certificate name is the name given to the certificate as it was entered in the keystore used by the SSL JDBC server you are connecting to.

If the KEYTOOL application was used to store certificates for the SSL JDBC server use, then the CertName should correspond to the name used within the “CN” parameter of the “-dname” attribute.

Example

```
$ keytool -genkey -alias rdbjdbc-sv -dname "CN=RdbJdbcServer,
OU=My Company, O=My Company, c=US" -keypass "MYSSLWK" -storepass
"MYSSLWK" -KeyStore rdbjdbcsrv.kst
```

See your Oracle JDBC for Rdb documentation for more information on using SSL JDBC servers.

See Also:

- [SSL and THIN connectivity](#)
-

3.8.2 SSLMode - NONE

If the SSLMode is set to “None”, ORDP.NET will not use SSL when trying to connect.

If you try to make a connection to a SSL-enabled JDBC server, the connection will eventually time-out as the initial connect handshake will fail.

See Also:

- [SSL and THIN connectivity](#)
-

3.8.3 SSLMode - VERIFYFULL

If the SSLMode is set to “VerifyFull”, ORDP.NET will use SSL when trying to connect.

“VerifyFull” tells ORDP.NET to carry out the is the highest level of SSL validation.

If the server is not using SSL, the connection request will be terminated immediately.

If the server is using SSL, ORDP.NET will validate the certificate sent to it by the server, and if it is valid, and it is recognized to be from a trusted source, and the certificate name matches, the connection request will proceed.

This SSLMode is used in conjunction with the connection attribute “CertName” which provides the name of the certificate that will be matched when ORDP.NET checks the validity of the certificate sent to it by the server.

If the certificate name returned by the server does not match the name provided within the “CertName” connection string attribute, the connection will be denied.

See Also:

- [SSL and THIN connectivity](#)
-

3.8.4 SSLMode - VERIFYCA

If the SSLMode is set to “VerifyCA”, ORDP.NET will use SSL when trying to connect.

If the server is not using SSL, the connection request will be terminated immediately.

If the server is using SSL, ORDP.NET will validate the certificate sent to it by the server, and if it is valid the connection request will proceed.

The certificate name is not checked.

See Also:

- [SSL and THIN connectivity](#)
-

3.8.5 SSLMode - ALLOWUNTRUSTED

If the SSLMode is set to “AllowUntrusted”, ORDP.NET will use SSL when trying to connect.

If the server is not using SSL, the connection request will be terminated immediately.

If the server is using SSL, ORDP.NET will validate the certificate sent to it by the server, and if it is valid, and the certificate name matches, the connection request will proceed.

This SSLMode is used in conjunction with the connection string attribute “CertName” which provides the name of the certificate that will be matched when ORDP.NET checks the validity of the certificate sent to it by the server.

If the certificate name returned by the server does not match the name provided within the “CertName” attribute, the connection will be denied.

See Also:

- [SSL and THIN connectivity](#)
-

3.8.6 SSLMode - REQUIRED

If the SSLMode is set to “Required”, ORDP.NET will use SSL when trying to connect.

If the server is not using SSL, the connection request will be terminated immediately.

If the server is using SSL, ORDP.NET will accept the certificate sent to it by the server without validation, and the connection request will proceed.

The certificate name is not checked.

See Also:

- [SSL and THIN connectivity](#)
-

3.9 Debug Tracing

ORDP.NET provides debug-tracing support, which allows logging of all the ORDP.NET activities into a trace file.

Tracelevel may be set using:

- `TraceLevel` attribute on the connection string See [Connection String Attributes](#)
- `TraceLevel` registry settings. See [Registry Settings for Tracing Calls](#).

Output from the trace messages will be written to `Console` if no trace file name is specified or is an empty string. The destination for the trace output may be set using:

- `TraceFileName` attribute on the connection string See [Connection String Attributes](#)
- `TraceFileName` registry settings. See [Registry Settings for Tracing Calls](#).

The value passed to trace, as specified in `TraceLevel`, is actually a 32bit flag mask. Each bit set determines what will be traced as shown in the following table.

[Table 3-5](#) lists the valid `TraceLevel` values and their descriptions.

Table 3-5 *TraceLevel values*

Bit	Hexadecimal Value	Decimal Value	Traces
0	0x00000001	1	Standard ORDP.NET methods entry
1	0x00000002	2	Standard ORDP.NET class create/finalize
2	0x00000004	4	SQL statements
4	0x00000010	16	Non-standard ORDP.NET methods entry
5	0x00000020	32	Non-standard ORDP.NET class create/finalize
8	0x00000100	256	Rdb SQS calls
9	0x00000200	512	Network sends
10	0x00000400	1024	Server actions
11	0x00000800	2048	Performance information
14	0x00004000	16384	Dump SQLDA information
15	0x00008000	32768	Connection pooling operations
29	0x20000000	536870912	Memory information
30	0x40000000	1073741824	Full; provides more details on certain flags
(ALL)	0xFFFFFFFF	-1	Trace everything

Caution:

Several of the trace flag values may produce copious output in the trace log file. In addition, setting the `server actions` flag may cause server activity to be logged on the server side.

3.9.2 Registry Settings for Tracing Calls

The following registry settings should be configured under:

HKEYLOCALMACHINE\SOFTWARE\ORACLE\ORDP.NET\HOME

TraceFileName

`TraceFileName` specifies the filename that is to be used for logging trace information.

If no entry exists for this key or the value is null or an empty string, client-side Debug trace output will be written to the Console.

TraceLevel

[Table 3-5](#) lists the valid `TraceLevel` values and their descriptions.

Chapter 4 Oracle.DataAccess.RdbClient Namespace

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

This chapter contains these topics:

- [Overview of Rdb Data Provider Classes](#)
- [Rdb Data Provider Classes](#)

4.1 Overview of Oracle Rdb Data Provider Classes

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

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

Assembly and Namespace

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

Class Inheritance

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

```
Object
  MarshalByRefObject
    Component
      RdbConnection
```

Interface Inheritance

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

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

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

Syntax Used

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

4.2 Oracle Rdb Data Provider Classes

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

- [RdbCommand](#) Class
- [RdbCommandBuilder](#) Class
- [RdbConnection](#) Class
- [RdbConnectionStringBuilder](#) Class
- [RdbDataAdapter](#) Class
- [RdbDataReader](#) Class
- [RdbError](#) Class
- [RdbErrorCollection](#) Class
- [RdbException](#) Class
- [RdbFactory](#) Class
- [RdbInfoMessageEventArgs](#) Class
- [RdbInfoMessageEventHandler](#) Delegate
- [RdbParameter](#) Class
- [RdbParameterCollection](#) Class
- [RdbRowUpdatedEventArgs](#) Class
- [RdbRowUpdatedEventHandler](#) Delegate
- [RdbRowUpdatingEventArgs](#) Class
- [RdbRowUpdatingEventHandler](#) Delegate
- [RdbTransaction](#) Class

4.2.1 RdbCommand Class

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

Class Inheritance

```
Object
  MarshalByRefObject
    Component
```

Declaration

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

Thread Safety

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

Remarks

The execution of any transaction-related statements from an `RdbCommand` is not recommended. The current state of a local transaction (if one exists) is maintained by an `RdbTransaction` object which is used internally to determine the appropriate actions that are required to take place for implicit commit or rollback operations. Transactional operations carried out explicitly within `RdbCommand` statements may fail to updated the `RdbTransaction` state which may result in unexpected implicit transaction problems.

Example

```
// C#
```

```
·
·
```

```

.
string conStr =
    @"User Id=rdb_user;Password=rdb_pw;
      Server=MYNODE:MY_SRV;Database=MY_DBS:MF_PERSONNEL";
// Create the RdbConnection
RdbConnection conn = new RdbConnection(conStr);

conn.Open();
string cmdQuery = "select last_name, employee_id from employees";
// Create the RdbCommand
RdbCommand cmd = new RdbCommand(cmdQuery);
cmd.Connection = conn;
cmd.CommandType = CommandType.Text;

// Execute command, create RdbDataReader object
RdbDataReader reader = cmd.ExecuteReader();
while (reader.Read())
{
    // output Employee Name and Number
    Console.WriteLine("Employee Name : " + reader.GetString(0) +
        " , " + "Employee Number : " + reader.GetString(1));
}

// Dispose RdbDataReader object
reader.Dispose();

// Dispose RdbCommand object
cmd.Dispose();

// Close and Dispose RdbConnection object
conn.Close();
conn.Dispose();
.
.
.

```

Requirements

Namespace: Oracle.DataAccess.RdbClient

Assembly: Oracle.DataAccess.Rdb.dll

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbCommand Members](#)
 - [RdbCommand Constructors](#)
 - [RdbCommand Static Methods](#)
 - [RdbCommand Properties](#)
 - [RdbCommand Public Methods](#)
-

4.2.1.1 RdbCommand Members

RdbCommand members are listed in the following tables:

RdbCommand Constructors

RdbCommand constructors are listed in [Table 4-1](#).

Table 4-1 RdbCommand Constructors

Constructor	Description.
-------------	--------------

RdbCommand Constructors	Instantiates a new instance of <code>RdbCommand</code> class (Overloaded)
---	---

RdbCommand Static Methods

`RdbCommand` static methods are listed in [Table 4-2](#).

Table 4-2 RdbCommand Static Methods

Methods	Description.
<code>Equals</code>	Inherited from <code>Object</code> (Overloaded)

RdbCommand Properties

`RdbCommand` properties are listed in [Table 4-3](#).

Table 4-3 RdbCommand Properties

Name	Description.
CommandText	Specifies the SQL statement or stored procedure to run against the Oracle <code>Rdb</code> database.
<code>CommandTimeout</code>	<i>(Currently Not supported)</i> Specifies the amount of time a command can execute before it will be timed-out.
CommandType	Specifies the command type that indicates how the <code>CommandText</code> property is to be interpreted.
Connection	Specifies the <code>RdbConnection</code> object that is used to identify the connection to execute a command.
<code>Container</code>	Inherited from <code>Component</code>
FetchSize	Specifies the size of the internal cache used by <code>RdbDataReader</code> to store result set data.
Parameters	Specifies the parameters for the SQL statement or stored procedure.
RdbCommandType	Specifies the extended command type that indicates how the <code>CommandText</code> property is to be interpreted.
RowsAffected	Specifies the number of rows affected by the SQL statement execution.
Transaction	Specifies the transaction associated with the command.
UpdatedRowSource	Specifies the value of the row after update.

RdbCommand Public Methods

`RdbCommand` public methods are listed in [Table 4-4](#).

Table 4-4 RdbCommand Public Methods

Public Method	Description.
<code>Cancel</code>	<i>Not Supported</i>
CreateParameter	Creates a new instance of <code>RdbParameter</code> class
Dispose	Dispose of the object after detaching command from the associated <code>RdbConnection</code>
<code>Equals</code>	Inherited from <code>Object</code> (Overloaded)
ExecuteNonQuery	Executes a SQL statement or a command using the <code>CommandText</code> 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
<code>GetHashCode</code>	Inherited from <code>Object</code>
<code>GetType</code>	Inherited from <code>Object</code>

GetHashCode	Inherited from Object
GetType	Inherited from Object
Prepare	<i>This method is a no-op</i>
ToString	Inherited from Object

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbCommand Constructors](#)
 - [RdbCommand Static Methods](#)
 - [RdbCommand Properties](#)
 - [RdbCommand Public Methods](#)
-

4.2.1.2 RdbCommand Constructors

RdbCommand constructors instantiate new instances of RdbCommand class.

Overload List:

- [RdbCommand\(\)](#)
This constructor instantiates a new instance of RdbCommand class.
- [RdbCommand\(string\)](#)
This constructor instantiates a new instance of RdbCommand class using the supplied SQL command or stored procedure.
- [RdbCommand\(RdbConnection\)](#)
This constructor instantiates a new instance of RdbCommand class using the supplied SQL connection to the Oracle Rdb database.
- [RdbCommand\(string, RdbConnection\)](#)
This constructor instantiates a new instance of RdbCommand class using the supplied SQL command or stored procedure, and connection to the Oracle Rdb database.
- [RdbCommand\(string, RdbConnection, RdbTransaction\)](#)
This constructor instantiates a new instance of RdbCommand class using the supplied SQL command or stored procedure, the transaction and connection to the Oracle Rdb database.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbCommand Members](#)
 - [RdbCommand Class](#)
-

RdbCommand()

This constructor instantiates a new instance of RdbCommand class.

Declaration

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

Remarks

Default constructor.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbCommand Members](#)
-

-
- [RdbCommand Class](#)
-

RdbCommand(string)

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

Declaration

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

Parameters

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

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbCommand Members](#)
 - [RdbCommand Class](#)
-

RdbCommand(RdbConnection)

This constructor instantiates a new instance of `RdbCommand` class using the connection to the Oracle Rdb database.

Declaration

```
// C#  
public RdbCommand(RdbConnection rdbConnection);
```

Parameters

- *RdbConnection*
Specifies the connection to the Oracle Rdb database.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbCommand Members](#)
 - [RdbCommand Class](#)
-

RdbCommand(string, RdbConnection)

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

Declaration

```
// C#  
public RdbCommand(string cmdText, RdbConnection rdbConnection);
```

Parameters

- *cmdText*
Specifies the SQL command or stored procedure to be executed.
- *RdbConnection*
Specifies the connection to the Oracle Rdb database.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbCommand Members](#)
 - [RdbCommand Class](#)
-

RdbCommand(string, RdbConnection, RdbTransaction)

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

Declaration

```
// C#  
public RdbCommand(string cmdText, RdbConnection rdbConnection,  
RdbTransaction rdbTransaction);
```

Parameters

- *cmdText*
Specifies the SQL command or stored procedure to be executed.
- *RdbConnection*
Specifies the connection to the Oracle Rdb database.
- *RdbTransaction*
Specifies the transaction to run this command.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbCommand Members](#)
 - [RdbCommand Class](#)
-

4.2.1.3 RdbCommand Static Methods

`RdbCommand` static methods are listed in [Table 4-5](#).

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbCommand Members](#)
 - [RdbCommand Class](#)
-

4.2.1.4 RdbCommand Properties

`RdbCommand` properties are listed in [Table 4-3](#).

CommandText

This property specifies the SQL statement or stored procedure to run against the Oracle Rdb database.

Declaration

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

Property Value

A string.

Implements

`IDbCommand`

Remarks

The default is an empty string.

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

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbCommand Members](#)
 - [RdbCommand Class](#)
-

CommandType

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

Declaration

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

Property Value

A `CommandType`.

Exceptions

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

Remarks

Default = `CommandType.Text`

When the `CommandType` property is set to `Text`, the `CommandText` must be a SQL query. The SQL query should be a select statement.

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.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbCommand Members](#)
 - [RdbCommand Class](#)
-

Connection

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

Declaration

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

Property Value

An `RdbConnection` object.

Implements

`IDbCommand`

Remarks

Default = null

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbCommand Members](#)
 - [RdbCommand Class](#)
-

FetchSize

This property specifies the number of records that may be stored in the `RdbDataReader` internal cache for result set data.

Declaration

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

Property Value

An `int` that specifies the number of records that may be stored in the `RdbDataReader` internal cache.

Exceptions

`ArgumentOutOfRangeException` - The `FetchSize` value specified is invalid, it must be greater than 0.

Remarks

Default = 100.

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

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbCommand Members](#)
 - [RdbCommand Class](#)
-

Parameters

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

Declaration

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

Property Value

`RdbParameterCollection`

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, the values in the `Parameters` property are ignored.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbCommand Members](#)
 - [RdbCommand Class](#)
-

RdbCommandType

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

Declaration

```
// C#
public RdbCommandTypes RdbCommandType
{final get; final set;}
```

Property Value

A `RdbCommandTypes`.

Exceptions

`ArgumentException` - The value is not one of the valid `RdbCommandTypes` such as: `RdbCommandTypes.Text`, `RdbCommandTypes.StoredProcedure` or `RdbCommandTypes.ExternalProcedure`

Remarks

Default = `RdbCommandTypes.Text`

When the `RdbCommandType` property is set to `RdbCommandTypes.Text`, the `CommandText` must be a SQL query. The SQL query should be a select statement.

When the `RdbCommandType` property is set to `RdbCommandTypes.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.

When the `RdbCommandType` property is set to `RdbCommandTypes.ExternalProcedure`, the `CommandText` property is set to the name of the external procedure. The command calls this external procedure when an `Execute` method is called.

`CommandType.Text` and `RdbCommandTypes.Text` are equivalent.

`CommandType.StoredProcedure` and `RdbCommandTypes.StoredProcedure` are equivalent.

Example

```
//C#
.
```

```
.  
RdbCommand cmd = cn.CreateCommand();  
cmd.RdbCommandType = RdbCommandTypes.ExternalProcedure;  
cmd.CommandText = "GET_SYMBOL_WITH_NAMES";  
.br/>.br/>.
```

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbCommand Members](#)
 - [RdbCommand Class](#)
 - [RdbCommandTypes Enumeration](#)
-

RowsAffected

This property specifies the number of rows affected by the execution of this command.

Declaration

```
// C#  
public int RowsAffected {get;}
```

Property Value

RowsAffected

Implements

IDbCommand

Remarks

Default value = none

`RowsAffected` returns the number of rows affected, if the command is UPDATE, INSERT, or DELETE. For all other types of statements, the return value is -1.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbCommand Members](#)
 - [RdbCommand Class](#)
-

Transaction

This property specifies the `RdbTransaction` object in which the `RdbCommand` executes.

Declaration

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

Property Value

RdbTransaction

Implements

IDbCommand

Remarks

Default value = null

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

Note:

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

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbCommand Members](#)
 - [RdbCommand Class](#)
-

UpdatedRowSource

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

Declaration

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

Property Value

An `UpdateRowSource`.

Implements

`IDbCommand`

Exceptions

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

Remarks

Default = `UpdateRowSource.None` if the command is automatically generated.

Default = `UpdateRowSource.Both` if the command is not automatically generated.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbCommand Members](#)
 - [RdbCommand Class](#)
-

4.2.1.5 RdbCommand Public Methods

`RdbCommand` public methods are listed in [Table 4-4](#).

CreateParameter

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

Declaration

```
// C#
```

```
public RdbParameter CreateParameter();
```

Return Value

A new `RdbParameter` with default values.

Implements

`IDbCommand`

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbCommand Members](#)
 - [RdbCommand Class](#)
-

Dispose

This method releases resources allocated for an `RdbCommand` object.

Declaration

```
// C#  
public void Dispose();
```

Implements

`IDisposable`

Remarks

`Dispose` will release resources allocated for an `RdbCommand` object after it has disposed of any associated `RdbParameters` and detached the command from its associated `Connection`.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbCommand Members](#)
 - [RdbCommand Class](#)
-

ExecuteNonQuery

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

Declaration

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

Return Value

The number of rows affected.

Implements

`IDbCommand`

Exceptions

`InvalidOperationException` - The command cannot be executed.

Remarks

`ExecuteNonQuery` returns the number of rows affected, if the command is `UPDATE`, `INSERT`, or `DELETE`. For all other types of statements, the return value is `-1`.

`ExecuteNonQuery` is used for either of the following:

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

Example

```
// C#
.
.
.
RdbConnection conn = new RdbConnection(
    @"User Id=rdb_user;Password=rdb_pw;
    Server=MYNODE:MY_SRV;Database=MY_DBS:MF_PERSONNEL;");

RdbCommand cmd = new RdbCommand(
    @"update salary_history set salary_amount = 33000
    where employee_id='00164' and salary_end is null", conn);

cmd.Connection.Open();
cmd.ExecuteNonQuery();
cmd.Dispose();
.
.
.
```

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbCommand Members](#)
 - [RdbCommand Class](#)
-

ExecuteReader

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

Overload List:

- [ExecuteReader\(\)](#)
This method executes a command specified in the `CommandText` and returns an `RdbDataReader` object.
- [ExecuteReader\(CommandBehavior\)](#)
This method executes a command specified in the `CommandText` and returns an `RdbDataReader` object, using the specified `CommandBehavior` value.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbCommand Members](#)
 - [RdbCommand Class](#)
-

ExecuteReader()

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

Declaration

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

Return Value

An `RdbDataReader`.

Implements

`IDbCommand`

Exceptions

`InvalidOperationException` - The command cannot be executed.

Remarks

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

The command executes this stored procedure when you call `ExecuteReader`

Example

```
// C#
.
.
.
RdbConnection conn = new RdbConnection(
    @"User Id=rdb_user;Password=rdb_pw;
    Server=MYNODE:MY_SRV;Database=MY_DBS:MF_PERSONNEL;");

RdbCommand cmd = new RdbCommand(
    "select last_name from employees", conn);

cmd.Connection.Open();
RdbDataReader reader = cmd.ExecuteReader();
while (reader.Read())
{
    Console.WriteLine("Employee Name : " + reader.GetString(0));
}
reader.Dispose();
cmd.Dispose();
.
.
.
```

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbCommand Members](#)
 - [RdbCommand Class](#)
-

ExecuteReader(CommandBehavior)

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

Declaration

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

Parameters

- *behavior*
Specifies expected behavior.

Return Value

An `RdbDataReader`.

Implements

`IDbCommand`

Exceptions

`InvalidOperationException` - The command cannot be executed.

Remarks

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

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

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

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbCommand Members](#)
 - [RdbCommand Class](#)
-

ExecuteScalar

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

Declaration

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

Return Value

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

Implements

`IDbCommand`

Exceptions

`InvalidOperationException` - The command cannot be executed.

Remarks

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

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

Note:

As `ExecuteScalar` returns an object, any immediate casting operation on this object is treated as an unboxing conversion, which has to be of the exact type of that object. A two-stage cast will need to be done if the type required is not the same as type returned by the `ExecuteScalar` operation.

Example

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

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbCommand Members](#)
 - [RdbCommand Class](#)
-

4.2.2 RdbCommandBuilder Class

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

Class Inheritance

```
Object
  MarshalByRefObject
    Component
```

Declaration

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

Thread Safety

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

Remarks

`RdbCommandBuilder` automatically generates SQL statements for single-table updates when the `SelectCommand` property of the `RdbDataAdapter` is set.

An exception is thrown if the `DataSet` contains multiple tables. The `RdbCommandBuilder` registers itself as a listener for `RowUpdating` events whenever its `DataAdapter` property is set. Only one `RdbDataAdapter` object and one `RdbCommandBuilder` object can be associated with each other at one time.

To generate `INSERT`, `UPDATE`, or `DELETE` statements, the `RdbCommandBuilder` 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.

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

Example

The following example uses the `RdbCommandBuilder` object to create the `UpdateCommand` for the `RdbDataAdapter` object when `RdbDataAdapter.Update()` is called.

```
// C#
public static void BuilderUpdate(String connStr)
{
    string cmdStr = "SELECT EMPLOYEE_ID, LAST_NAME FROM EMPLOYEES";
    //create the adapter with the selectCommand txt and the
    //connection string
    RdbDataAdapter adapter = new RdbDataAdapter(cmdStr, connStr);
    //get the connection from the adapter
    RdbConnection connection = adapter.SelectCommand.Connection;
    //create the builder for the adapter to automatically generate
    //the Command when needed
    RdbCommandBuilder builder = new RdbCommandBuilder(adapter);
    //Create and fill the DataSet using the EMPLOYEES
    DataSet dataset = new DataSet();
    adapter.Fill(dataset, "EMPLOYEES");
    //Get the EMP table from the dataset
    DataTable table = dataset.Tables["EMPLOYEES"];
    //Get the first row from the EMPLOYEES table
    DataRow row0 = table.Rows[0];
    //update the job description in the first row
    row0["LAST_NAME"] = "JONES";
    //Now update the first EMPLOYEES using the adapter, the last name
    //is changed to 'JONES'
    //The RdbCommandBuilder will create the UpdateCommand for the
    //adapter to update the EMPLOYEES table
    adapter.Update(dataset, "EMPLOYEES");
}
```

Requirements

Namespace: `Oracle.DataAccess.RdbClient`

Assembly: `Rdb.DataAccess.Rdb.dll`

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbCommandBuilder Members](#)
 - [RdbCommandBuilder Constructors](#)
 - [RdbCommandBuilder Static Methods](#)
 - [RdbCommandBuilder Properties](#)
 - [RdbCommandBuilder Public Methods](#)
 - [RdbCommandBuilder Events](#)
 - [RdbCommandBuilder Event Delegates](#)
-

4.2.2.1 RdbCommandBuilder Members

`RdbCommandBuilder` members are listed in the following tables:

RdbCommandBuilder Constructors

RdbCommandBuilder constructors are listed in [Table 4-5](#).

Table 4-5 RdbCommandBuilder Constructors

Constructor	Description
RdbCommandBuilder Constructors	Instantiates a new instance of RdbCommandBuilder class (Overloaded).

RdbCommandBuilder Static Methods

RdbCommandBuilder static methods are listed in [Table 4-6](#).

Table 4-6 RdbCommandBuilder Static Methods

Method	Description
Equals	Inherited from Object.
DeriveParameters	Derives the RdbParameterCollection for the specified RdbCommand object.

RdbCommandBuilder Properties

RdbCommandBuilder properties are listed in [Table 4-7](#).

Table 4-7 RdbCommandBuilder Properties

Name	Description
Container	Inherited from Component.
DataAdapter	Indicates the RdbDataAdapter for which the SQL statements are generated.
CaseSensitive	Indicates whether or not double quotes are used around Rdb object names when generating SQL statements.

RdbCommandBuilder Public Methods

RdbCommandBuilder public methods are listed in [Table 4-8](#).

Table 4-8 RdbCommandBuilder Public Methods

Public Method	Description
Dispose	Inherited from Component.
Equals	Inherited from Object (Overloaded).
GetDeleteCommand	Gets the automatically generated RdbCommand object that has the SQL statement (CommandText) to perform deletions on the database.
GetHashCode	Inherited from Object.
GetInsertCommand	Gets the automatically generated RdbCommand object that has the SQL statement (CommandText) to perform insertions on the database.
GetType	Inherited from Object.
GetUpdateCommand	Gets the automatically generated RdbCommand object that has the SQL statement (CommandText) to perform updates on the database.
RefreshSchema	Refreshes the database schema information used to generate INSERT, UPDATE, or DELETE statements.
ToString	Inherited from Object.

RdbCommandBuilder Events

RdbCommandBuilder events are listed in [Table 4-9](#).

Table 4-9 RdbCommandBuilder Events

Event Name	Description
Disposed	Inherited from Component .

RdbCommandBuilder Event Delegates

RdbCommandBuilder event delegates are listed in [Table 4-10](#).

Table 4-10 RdbCommandBuilder Event Delegates

Event Delegate Name	Description
EventHandler	Inherited from Component .

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
- [RdbCommandBuilder Constructors](#)
- [RdbCommandBuilder Static Methods](#)
- [RdbCommandBuilder Properties](#)
- [RdbCommandBuilder Public Methods](#)
- [RdbCommandBuilder Events](#)
- [RdbCommandBuilder Event Delegates](#)

4.2.2.2 RdbCommandBuilder Constructors

RdbCommandBuilder constructors create new instances of the RdbCommandBuilder class.

Overload List:

- [RdbCommandBuilder\(\)](#)
This constructor creates an instance of the RdbCommandBuilder class.
- [RdbCommandBuilder\(RdbDataAdapter\)](#)
This constructor creates an instance of the RdbCommandBuilder class and sets the DataAdapter property to the provided RdbDataAdapter object.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
- [RdbCommandBuilder Members](#)
- [RdbCommandBuilder Class](#)

RdbCommandBuilder()

This constructor creates an instance of the RdbCommandBuilder class.

Declaration

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

Remarks

Default constructor.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)

-
- [RdbCommandBuilder Members](#)
 - [RdbCommandBuilder Class](#)
-

RdbCommandBuilder(RdbDataAdapter)

This constructor creates an instance of the `RdbCommandBuilder` class and sets the `DataAdapter` property to the provided `RdbDataAdapter` object.

Declaration

```
// C#  
public RdbCommandBuilder(RdbDataAdapter da);
```

Parameters

- *da*
The `RdbDataAdapter` object provided.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbCommandBuilder Members](#)
 - [RdbCommandBuilder Class](#)
-

4.2.2.3 RdbCommandBuilder Static Methods

`RdbCommandBuilder` static methods are listed in [Table 4-6](#).

DeriveParameters

This method automatically derives the parameters for an `RdbCommand` object of the type `CommandType.StoredProcedure`.

Declaration

```
// C#  
public static void DeriveParameters(RdbCommand cmd);
```

Return Value

None.

Exceptions

`ObjectDisposedException` - The `RdbCommand` object specified is already disposed.
`InvalidOperationException` - The `RdbCommand` object specified is not of type `CommandType.StoredProcedure`.

Remarks

Information returned by Oracle Rdb is used to create a parameter collection suitable for use with the specified `RdbCommand` object. Once called the [RdbCommand.Parameters](#) property of the specified `RdbCommand` object may be used to get the [RdbParameterCollection](#) generated.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbCommandBuilder Members](#)
 - [RdbCommandBuilder Class](#)
-

4.2.2.4 RdbCommandBuilder Properties

`RdbCommandBuilder` properties are listed in [Table 4-7](#).

DataAdapter

This property indicates the `RdbDataAdapter` for which the SQL statements are generated.

Declaration

```
// C#  
RdbDataAdapter DataAdapter{get; set;}
```

Property Value

`RdbDataAdapter`

Remarks

Default = null

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
- [RdbCommandBuilder Members](#)
- [RdbCommandBuilder Class](#)

CaseSensitive

This property indicates whether or not double quotes are used around `Rdb` object names (for example, tables or columns) when generating SQL statements.

Declaration

```
// C#  
bool CaseSensitive {get; set;}
```

Property Value

A `bool` that indicates whether or not double quotes are used.

Remarks

Default = false

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
- [RdbCommandBuilder Members](#)
- [RdbCommandBuilder Class](#)

4.2.2.5 RdbCommandBuilder Public Methods

`RdbCommandBuilder` public methods are listed in [Table 4-8](#).

GetDeleteCommand

This method gets the automatically generated `RdbCommand` object that has the SQL statement (`CommandText`) perform deletions on the database when an application calls `Update()` on the `RdbDataAdapter`.

Declaration

```
// C#  
public RdbCommand GetDeleteCommand();
```

Return Value

An `RdbCommand`.

Exceptions

`ObjectDisposedException` - The `RdbCommandBuilder` 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 `RdbDataAdapter`.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbCommandBuilder Members](#)
 - [RdbCommandBuilder Class](#)
-

GetInsertCommand

This method gets the automatically generated `RdbCommand` object that has the SQL statement (`CommandText`) perform insertions on the database when an application calls `Update()` on the `RdbDataAdapter`.

Declaration

```
// C#  
public RdbCommand GetInsertCommand();
```

Return Value

An `RdbCommand`.

Exceptions

`ObjectDisposedException` - The `RdbCommandBuilder` 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 `RdbDataAdapter`.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbCommandBuilder Members](#)
 - [RdbCommandBuilder Class](#)
-

GetUpdateCommand

This method gets the automatically generated `RdbCommand` object that has the SQL statement (`CommandText`) perform updates on the database when an application calls `Update()` on the `RdbDataAdapter`.

Declaration

```
// C#  
public RdbCommand GetUpdateCommand();
```

Return Value

An `RdbCommand`.

Exceptions

`ObjectDisposedException` - The `RdbCommandBuilder` 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 `RdbDataAdapter`.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbCommandBuilder Members](#)
 - [RdbCommandBuilder Class](#)
-

RefreshSchema

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

Declaration

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

Remarks

An application should call `RefreshSchema` whenever the `SelectCommand` value of the `RdbDataAdapter` changes.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbCommandBuilder Members](#)
 - [RdbCommandBuilder Class](#)
-

4.2.2.6 RdbCommandBuilder Events

`RdbCommandBuilder` events are listed in [Table 4–9](#).

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbCommandBuilder Members](#)
 - [RdbCommandBuilder Class](#)
-

4.2.2.7 RdbCommandBuilder Event Delegates

`RdbCommandBuilder` event delegates are listed in [Table 4–10](#).

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbCommandBuilder Members](#)
 - [RdbCommandBuilder Class](#)
-

4.2.3 RdbConnection Class

An `RdbConnection` object represents a connection to an Oracle Rdb database.

Class Inheritance

```
Object  
  RdbConnection
```

Declaration

```
// C#  
public sealed class RdbConnection : IDisposable, IDbConnection, ICloneable
```

Thread Safety

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

Example

```
// C#
.
.
.
// Uses connection to create and return an RdbCommand object.

string conStr =
@"Server=node1.oracle.com:GENSRVC;Database=mydb;
  User Id=myname;Password=mypassword;";
RdbConnection conn = new RdbConnection(conStr);
conn.Open();
RdbCommand cmd = conn.CreateCommand();
cmd.CommandText = "insert into mytable values (99, 'foo')";
cmd.CommandType = CommandType.Text;
cmd.ExecuteNonQuery();
.
.
.
```

Requirements

Namespace: Oracle.DataAccess.RdbClient

Assembly: Rdb.DataAccess.Rdb.dll

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
- [RdbConnection Members](#)
- [RdbConnection Constructors](#)
- [RdbConnection Static Methods](#)
- [RdbConnection Properties](#)
- [RdbConnection Public Methods](#)
- [RdbConnection Events](#)
- [RdbConnection Event Delegates](#)

4.2.3.1 RdbConnection Members

`RdbConnection` members are listed in the following tables:

RdbConnection Constructors

`RdbConnection` constructors are listed in [Table 4-11](#).

Table 4-11 RdbConnection Constructors

Constructor	Description
RdbConnection Constructors	Instantiates a new instance of <code>RdbConnection</code> class (Overloaded).

RdbConnection Static Methods

`RdbConnection` static methods are listed in [Table 4-12](#).

Table 4-12 RdbConnection Static Methods

Method	Description
ClearAllPools	Closes all free connections in all connection pools .
Equals	Inherited from <code>Object</code> (Overloaded).
EstablishPoolCriteria	Establish limits for all implicit connection pools.

RdbConnection Properties

`RdbConnection` properties are listed in [Table 4-13](#).

Table 4-13 RdbConnection Properties

Name	Description
ConnectionString	Specifies connection information used to connect to an Rdb database.
ConnectionTimeout	Maximum time (in seconds) to wait for a connection. This attribute specifies the maximum amount of time (in seconds) that the <code>Open ()</code> method can take to obtain a connection before it terminates the request.. If the connection is not made within the specified time, an exception is thrown. A value of zero (0) means wait indefinitely for the connection.
Database	Identifies the database to connect to.
Password	Specifies the password.
ReadOnly	Specifies the READONLY state for this connection.
Server	Specifies the name of the server to use for this connection.
ServerType	Specifies the type of server.
State	Specifies the current state of the connection.
TraceLevel	Specifies the trace level for this connection.
UserId	Specifies the user.

RdbConnection Public Methods

`RdbConnection` public methods are listed in [Table 4-14](#).

Table 4-14 RdbConnection Public Methods

Public Method	Description
BeginTransaction	Begins a local transaction (Overloaded).
ChangeDatabase	Changes the database component of the connection and reconnects to the new database.
Clone	Creates a copy of an <code>RdbConnection</code> object.
Close	Closes the database connection.
CreateCommand	Creates and returns an <code>RdbCommand</code> object.
Dispose	Inherited from <code>IDisposable</code> .
Equals	Inherited from <code>Object</code> (Overloaded).
GetHashCode	Inherited from <code>Object</code> .
GetSchema	Returns connection schema information.
GetType	Inherited from <code>Object</code> .
IsOpen	Returns <code>true</code> if the connection state is not <code>ConnectionState.Closed</code> and is not <code>ConnectionState.Broken</code> .
IsClosed	Returns <code>true</code> if the connection state is <code>ConnectionState.Closed</code> or is <code>ConnectionState.Broken</code> .
IsBroken	Returns <code>true</code> if the connection state is <code>ConnectionState.Broken</code> .
Open	Opens a database connection with the property settings specified by the <code>ConnectionString</code> and other explicitly specified properties.
ToString	Inherited from <code>Object</code> .

RdbConnection Events

RdbConnection events are listed in [Table 4-15](#).

Table 4-15 RdbConnection Events

Event Name	Description
Disposed	Inherited from Component .

RdbConnection Event Delegates

RdbConnection event delegates are listed in [Table 4-16](#).

Table 4-16 RdbConnection Event Delegates

Event Delegate Name	Description
EventHandler	Inherited from Component .

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
- [RdbConnection Members](#)
- [RdbConnection Constructors](#)
- [RdbConnection Static Methods](#)
- [RdbConnection Properties](#)
- [RdbConnection Public Methods](#)
- [RdbConnection Events](#)
- [RdbConnection Event Delegates](#)

4.2.3.2 RdbConnection Constructors

RdbConnection constructors instantiate new instances of the RdbConnection class.

Overload List:

- [RdbConnection\(\)](#)
This constructor instantiates a new instance of the RdbConnection class using default property values.
- [RdbConnection\(String\)](#)
This constructor instantiates a new instance of the RdbConnection class with the provided connection string.

RdbConnection()

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

Declaration

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

Remarks

The properties for RdbConnection are set to the following default values:

- ConnectionString = empty string
- ConnectionTimeout = 0
- Database = empty string

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbConnection Members](#)
 - [RdbConnection Class](#)
-

RdbConnection(String)

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

Declaration

```
// C#  
public RdbConnection(String connectionString);
```

Parameters

- `connectionString`
The connection information used to connect to the Oracle Rdb 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 `RdbConnection` object default to the following values unless the connection string sets them:

- `ConnectionString` = empty string
- `ConnectionTimeout` = 0
- `Database` = empty string

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbConnection Members](#)
 - [RdbConnection Class](#)
-

4.2.3.3 RdbConnection Static Methods

`RdbConnection` static methods are listed in [Table 4-12](#).

ClearAllPools**Declaration**

```
// C#  
public void ClearAllPools();
```

Remarks

Clear all free connections in all the connection pools in an application domain. All free connections will be closed.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbConnection Members](#)
 - [RdbConnection Class](#)
 - [Connection Pool Management](#)
-

EstablishPoolCriteria

Declaration

```
// C#
public void EstablishPoolCriteria(int maxPoolSize, int maxFree,
    int minFree, int cleanerPeriod, int connectionLifeTime,
    int connectionTimeout, bool validateConnection);
```

Parameters

- *maxPoolSize*
The maximum total number of concurrent connections, free or otherwise, allowed in pool.
- *maxFree*
The maximum number of free connections allowed in pool.
- *minFree*
The minimum number of free connections maintained by the pool.
- *cleanerPeriod*
The time in seconds between executions of the pool cleaner.
- *connectionLifeTime*
The maximum time in seconds a connection may exist. This is checked only when a connection is to be returned to the pool.
- *connectionTimeout*
The maximum time in seconds a requester should wait for a free connection.
- *validateConnection*
If *true*, the connection should be checked to ensure that it is still available. This is checked only when a connection is retrieved from or is returned to the pool.

Remarks

The specified criteria are placed on all implicit connections pools created subsequent to calling this method.

Example

```
// C#
.
.
.
string conStr =
    @"Type=POOLEDSQL;Server=node1.oracle.com:GENSRVC;
    Database=mydb;User Id=myname;Password=mypassword;";

int maxPool = 10; // only allow 10 concurrent connections in any pool
int maxFree = 5; // maintain a maximum of 5 free connections in any pool
int minFree = 3; // maintain a minimum of 3 free connections in any pool
int connLife = 0; // allow free connections to exist indefinitely
int cleanerPeriod = 0; // don't run cleaner thread
int connTimeout = 0; // raise an exception immediately if maxPool
    // exceeded and we tried to get a new connection
bool validateServer = false; // true would mean an extra network i/o on
    // the connection being returned from pool

RdbConnection.EstablishPoolCriteria(maxPool, maxFree, minFree,
    connLife, cleanerPeriod, connTimeout, validateServer);

RdbConnection conn = new RdbConnection(conStr);

// open the connection and as this is the first one it will establish the
```

```
// connection pool with limits as specified above  
  
conn.Open();  
// the close will return the connection to the connection pool  
// which will have one free connection now  
conn.Close();  
. . .
```

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbConnection Members](#)
 - [RdbConnection Class](#)
 - [Connection Pool Management](#)
-

4.2.3.4 RdbConnection Properties

RdbConnection properties are listed in [Table 4-13](#)

ConnectionString

This property specifies connection information used to connect to an Oracle Rdb database.

Declaration

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

Property Value

If the connection string is supplied through the constructor, this property is set to that string.

Exceptions

ArgumentException - An invalid syntax is specified for the connection string.

ArgumentNullException - Connection string is null

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 semicolon. If the ConnectionString is not in a proper format, an exception is thrown. All leading and trailing spaces either side of the equals sign ("=") are ignored.

When the ConnectionString property is set, the RdbConnection 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, Server and Database (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.

ORDP.NET supports connections made to an Oracle Rdb database using one of the following types of server connections:

- SQS - Oracle SQL/Services Service

- THIN - Oracle JDBC for Rdb Server
- POOLEDSQS – pooled Oracle SQL/Services Service
- POOLEDTHIN – pooled Oracle JDBC for Rdb Server

Supported Server Attributes.

The type of server to use may be specified either using the `Type` attribute within the connection string, or by using the `ServerType` property. Whichever is set last prior to the `Open` method being called will take precedence.

The `Server` attribute within the connection string, or the `Server` property may be used in conjunction with the `Type` attribute within the connection string, or the `ServerType` property to provide an appropriate server connection.

If the type of server is "SQS" or "POOLEDSQS" then the `Server` must be a valid Oracle SQL/Services service designation of the format:

`Node:Service`

Where:

- `Node`
is a valid TCPIP node specification of an OpenVMS node where SQL/Services is available for Rdb connections.
- `Service`
is the name of a valid SQL/Services universal or database service using protocol "SQLSERVICES" running on the specified `Node`.

If the type of server is "THIN" or "POOLEDTHIN" then the `Server` must be a valid Oracle JDBC for Rdb partial connection URL of the format:

`Node:Port`

Where:

- `Node`
is a valid TCPIP node specification of an OpenVMS node where an Oracle JDBC for Rdb server is available for Rdb connections.
- `Port`
is the TCPIP port number on the specified `Node` that the Oracle JDBC for Rdb server is listening on.

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`. Case is ignored.

Supported connection string attributes:

[Table 4-17](#) lists the supported connection string attributes.

[Table 3-4](#) lists the supported connection string attributes for Pool Manager connections.

Table 4-17 Supported Connection String Attributes

Connection String Attribute	Default value	Description
Certificate Name Or CertName	RdbJdbcServer	Specifies the name of the SSL Certificate expected when attempting a THIN connection using SSL. This attribute is only applicable to THIN connections. This attribute is used in conjunction with the SSLMode connection string attribute. See SSL and THIN connectivity for more details.
ConnectionTimeout or Connection Timeout	0	Maximum time (in seconds) to wait for a connection. This attribute specifies the maximum amount of time (in seconds) that the <code>Open()</code> method can take to obtain a connection before it terminates the request. If the connection is not made within the specified time, an exception is thrown. A value of zero (0) means wait indefinitely for the connection.
Database or Data Source	empty string	Identifies the database associated with the connection. If null or an empty string the default database for the specified server will be used.
Enlist	false	Specifies whether the connection should automatically enlist in the current system transaction.
Keyinfo	false	If true, specifies that extra key information should be collected for all SQL statements processed within this connection. This extra key information may be required for the correct functioning of Entity Framework or DDEX. The extra processing may increase network latency.
National Language Or NlsLang	LATIN1	Specifies the National Language that ORDP should encoded SQL text in. By default characters within SQL text will be interpreted by ORDP as LATIN1 characters. Currently ORDP supports the following character set names in this attribute: <ul style="list-style-type: none"> LATIN1 – default Latin1 characters BIG5 – mixed BIG5 and ASCII characters
Password or Pwd	empty string	Password for the user specified by <code>User Id</code> . This attribute specifies an Rdb user's password. Password is case insensitive. A value of true means the connection should enlist and use the transaction attributes of the current system transaction.
Pooling	false	Specifies whether connection pooling should be enabled for this connection. A value of true means the connection will take part in connection pooling.
PoolValidateConnection	false	NOTE: This attribute is currently ignored by ORDP.NET. If true when used with a POOLEDSQLS or POOLEDTHIN connection, ORDP will carry out connection viability checks prior to releasing it to a new connection request. See Connection Viability Check for more details.
ReadOnly	false	Specifies whether the connection is to be considered READONLY thus preventing update operations from being carried out. A value of true means the connection will be set to a READONLY state.

Connection String Attribute	Default value	Description
Server	empty string	Identifies the server to use for the connection. If <code>Type</code> is specified and is "THIN" the <code>Server</code> must be a valid Oracle JDBC for Rdb connection URL. If <code>Type</code> is not specified or is "SQS" the <code>Server</code> must be a valid Oracle SQL/Services for Rdb connection specification.
SSLMode	NONE	Specifies that the connection to the Oracle JDBC thin server should be made using a secured connection. Valid values are: <ul style="list-style-type: none"> • <code>None</code> – do not use SSL . • <code>Required</code> – accept any certificate delivered back from the server. Deny connection if server does not support SSL. • <code>VerifyCA</code> – validate server’s SSL certificate. Still allow connection even if certificate name is mismatched. • <code>VerifyFull</code> – validate server’s SSL certificate and only allow trusted servers. Deny connection if certificate name is mismatched. • <code>AllowInstall</code> – allow installation of certificate from server into local keystore. • <code>AllowUntrusted</code> – validate servers SSL certificate, and allow even if untrusted. Deny connection if certificate name is mismatched. <p>If <code>SSLMode</code> is not specified, or is specified as “None”, a normal THIN connection will be made.</p> <p>This attribute is only applicable to THIN connections. This attribute is used in conjunction with the <code>CertName</code> connection string attribute. See SSL and THIN connectivity for more details.</p>
Style	empty string	Specifies the <code>style</code> of the connection. Valid <code>styles</code> are: <ul style="list-style-type: none"> • <code>SQS</code> – use SQL/Services style semantics • <code>JDBC</code> – use JDBC semantics • <code>ODBC</code> – use ODBC semantics • <code>SQLSERVER</code> – use SQL Server semantics <p>If not specified or an empty string is specified, the default connection behavior will be used.</p>
TraceFilename or Trace Filename	empty string	Specifies the file to write trace messages to. If not specified or an empty string is specified, trace message will be written to Console.
TraceLevel or Trace	0	Specifies the debug trace level to use on the connection.
Type	SQS	Specifies the type of <code>Server</code> connection. Valid types are: <ul style="list-style-type: none"> • <code>SQS</code> - make an Oracle SQL/Services connection • <code>THIN</code> – make a connection to an Oracle JDBC for Rdb Server

Connection String Attribute	Default value	Description
		<ul style="list-style-type: none"> • POOLEDSQLS - make a pooled Oracle SQL/Services connection • POOLEDTHIN – make a pooled connection to an Oracle JDBC for Rdb Server
		If not specified or an empty string is specified the default type will be used.
User Id or User or Username	empty string	Rdb user name. This attribute specifies the Rdb user name.

Example

```
// C#
.
.
.
RdbConnection conn = new RdbConnection();
conn.ConnectionString =
    @"User Id=MYNAME;Password=MYPASSWORD;
    Server=DBS_SRV:SQSGENERIC;Database=MYDB;Type=SQLS";
.
.
.
```

In addition to the above attributes, the `ConnectionString` property may also contain attributes specific to Pool Manager connections. See [Establishing a Pool Manager](#) for more details.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
- [RdbConnection Members](#)
- [RdbConnection Class](#)
- [Establishing a Pool Manager](#)

CertName

This property specifies the name of certificate to check when using SSL.

Declaration

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

Property Value

The certificate name used to check that SSL certificate source is authentic.

Remarks

The default value is “RdbJdbcServer”.

Setting this property to null or an empty string will cause ORDP to use the default value.

If present, the string value should be the certificate name of the certificate used by the SSL-enable JDBC server that will be connected to.

Used in conjunction with the [SSLMode](#) property.

See [SSL and THIN connectivity](#) for more information

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbConnection Members](#)
 - [RdbConnection Class](#)
 - [SSL and THIN connectivity](#)
-

ConnectionTimeout

This property specifies the maximum amount of time that the `Open()` method can take to obtain a connection before terminating the request.

Declaration

```
// C#  
public int ConnectionTimeout {get;}
```

Property Value

The maximum time allowed for connection request, in seconds.

Remarks

The default value is 0 .

Setting this property to 0 allows the connection request to wait without a time limit.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbConnection Members](#)
 - [RdbConnection Class](#)
-

Database

This property specifies a name or file specification that identifies an Oracle Rdb database instance.

Declaration

```
// C#  
public string Database {get; }
```

Property Value

The Oracle Rdb database file specification.

Remarks

NOTE :

In compliance with the generalized `DbConnection` class defined in .NET V2.0 , the `Database` property can no longer be SET. Use the `SetDatabase` method instead.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbConnection Members](#)
 - [RdbConnection Class](#)
-

Enlist

This property specifies if the connection should automatically take part in the current system transaction.

Declaration

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

Property Value

The Enlist state of the connection.

Exceptions

`InvalidOperationException: Enlist attribute is being set while the connection is open.`

Remarks

The default is `false`.

The connection must be closed to set the `Enlist` property.

When the `Enlist` property is `true`, the connection will attempt to enlist in the current system transaction.

`RdbException: Invalid operation for read only connection`

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbConnection Members](#)
 - [RdbConnection Class](#)
-

Password

This property specifies a password used for the connection.

Declaration

```
// C#  
public string Password { set;}
```

Property Value

The Oracle Rdb database file specification.

Remarks

The default is an empty string.

The connection must be closed to set the `Password` property.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbConnection Members](#)
 - [RdbConnection Class](#)
-

ReadOnly

This property specifies if the connection is to be considered `READONLY`.

Declaration

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

Property Value

The READONLY state of the connection.

Exceptions

`InvalidOperationException: ReadOnly` attribute is being set while the connection is open.

Remarks

The default is `false`.

The connection must be closed to set the `ReadOnly` property.

When the `ReadOnly` property is `true`, any attempt to carry out an update operation using this connection will result in an exception being raised:

```
RdbException: Invalid operation for read only connection
```

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbConnection Members](#)
 - [RdbConnection Class](#)
-

Server

This property identifies a server to use to make the connection.

Declaration

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

Property Value

The server specification.

Exceptions

`InvalidOperationException` - `Server` is being set while the connection is open.
`ArgumentNullException` - `Server` is either null or an empty string

Remarks

The connection must be closed to set the `Server` property.

The `Server` property must be a valid SQL/Services connection string or a valid Oracle JDBC for Rdb partial connection URL.

The `Server` property is used in conjunction with the `Type` attribute and the `ServerType` property to specify the attributes of the server to which to connect.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbConnection Members](#)
 - [RdbConnection Class](#)
 - [Supported Server Attributes](#)
 - [ServerType Property](#)
 - [ConnectionString Property](#)
 - [Supported Connection String Attributes](#)
 - Oracle SQL/Services for Rdb documentation on Service connection strings.
 - Oracle JDBC for Rdb documentation on server connection strings.
-

ServerType

This property specifies the type of server to which the connection will be made.

Declaration

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

Property Value

The server type for the connection.

Exceptions

`InvalidOperationException` - `ServerType` is being set while the connection is open.

`ArgumentOutOfRangeException` - `ServerType` is not one of:

- null or empty string
- "SQS"
- "THIN"
- "POOLEDSQL"
- "POOLEDTHIN"

Remarks

The default is "SQS".

If the `ServerType` is null or an empty string, "SQS" will be used.

Valid types are:

- `SQS` – make an Oracle SQL/Services connection
- `THIN` – make a connection to an Oracle JDBC for Rdb Server
- `POOLEDSQL` – take an Oracle SQL/Services connection from pool
- `POOLEDTHIN` – take a connection to an Oracle JDBC for Rdb Server from pool

The `ServerType` is used in conjunction with the `Server` attribute or property to specify the attributes of the server to connect to.

The connection must be closed to set the `ServerType` property.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbConnection Members](#)
 - [RdbConnection Class](#)
 - [Supported Server Attributes](#)
 - [Server Property](#)
 - [ConnectionString Property](#)
 - [Supported Connection String Attributes](#)
-

SSLMode

This property specifies the mode to use when connecting using SSL with a `THIN` connection.

Declaration

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

Property Value

The `SSLMode` of the connection.

Implements

none

Remarks

The default SSLMode is “None”.

If SSLMode is specified and is not “None”, SSL will be used to connect to the Oracle JDBC for Rdb Server.

Valid values are:

- None – do not use SSL .
- Required – accept any certificate delivered back from the server. Deny connection if server does not support SSL.
- VerifyCA – validate server’s SSL certificate. Still allow connection even if certificate name is mismatched.
- VerifyFull – validate server’s SSL certificate and only allow trusted servers. Deny connection if certificate name is mismatched.
- AllowInstall – allow installation of certificate from server into local keystore.
- AllowUntrusted – validate servers SSL certificate, and allow even if untrusted. Deny connection if certificate name is mismatched.

The SSLMode should be used only in conjunction with a THIN connection.

The SSLMode is used in conjunction with the [CertName](#) property.

See [SSL and THIN connectivity](#) for more information

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbConnection Members](#)
 - [RdbConnection Class](#)
 - [SSL and THIN connectivity](#)
-

State

This property specifies the current state of the connection.

Declaration

```
// C#  
public ConnectionState State {get;}
```

Property Value

The `ConnectionState` of the connection.

Implements

`System.Data.ConnectionState`

Remarks

ORDP.NET supports `ConnectionState.Closed` and `ConnectionState.Open` for this property. The default value is `ConnectionState.Closed`.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbConnection Members](#)
 - [RdbConnection Class](#)
-

Style

This property specifies the current style or behavior of the connection.

Declaration

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

Property Value

The `Style` of the connection.

Remarks

Valid styles are:

- null or empty string – use DEFAULT behavior
- "SQS" – use SQL/Services style semantics
- "JDBC" – use JDBC semantic
- "ODBC" – use ODBC semantics
- "SQLSERVER" – use SQL Server semantics

The behavior of operations carried out in the connection may be affected.

Currently this attribute only affects the behavior when the `RdbConnection.AttachDataReader` method is called when a datareader is already attached to the connection.

- DEFAULT style - the new reader will be added to the list of current readers
- SQS style – if the `RdbConnection.FetchSize` is > 1 then the existing reader will be silently closed otherwise the new reader will be added to the list of current readers
- JDBC style - the old reader will be silently closed
- ODBC – An `InvalidOperationException` exception will be thrown
- SQLSERVER - the old reader will be silently closed

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbConnection Members](#)
 - [RdbConnection Class](#)
-

TraceFilename

This property specifies the filename where trace messages will be written.

Declaration

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

Property Value

The trace filename used to write trace messages to for debugging purposes.

Remarks

The default value is an empty string.

Setting this property to null or an empty string will cause ORDP.NET to send trace messages to the Console.
If present, the string value should be a valid file specification.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbConnection Members](#)
 - [RdbConnection Class](#)
-

TraceLevel

This property specifies the trace level for debugging purposes.

Declaration

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

Property Value

The trace level for debugging.

Remarks

The default value is 0 .

Setting this property to 0 disables debug tracing.

See [Debug Tracing](#) for more information about the tracelevel values allowed.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbConnection Members](#)
 - [RdbConnection Class](#)
 - [Debug Tracing](#)
-

UserId

This property specifies the user name to connect with.

Declaration

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

Property Value

The username for the connection.

Remarks

The default is an empty string.

The connection must be closed to set the `UserId` property.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbConnection Members](#)
 - [RdbConnection Class](#)
-

4.2.3.5 RdbConnection Public Methods

`RdbConnection` public methods are listed in [Table 4–14](#).

BeginTransaction

`BeginTransaction` methods begin local transactions.

Overload List

- [BeginTransaction\(\)](#)
This method begins a local transaction.
- [BeginTransaction\(IsolationLevel\)](#)
This method begins a local transaction with the specified isolation level.
- [BeginTransaction\(String\)](#)
This method begins a local transaction with the specified transaction type information.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbConnection Members](#)
 - [RdbConnection Class](#)
-

BeginTransaction()

This method begins a local transaction.

Declaration

```
// C#  
public RdbTransaction BeginTransaction();
```

Return Value

An `RdbTransaction` 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 `System.Data.IsolationLevel.ReadCommitted`. All further operations related to the transaction must be performed on the returned `RdbTransaction` object.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbConnection Members](#)
 - [RdbConnection Class](#)
-

BeginTransaction(IsolationLevel)

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

Declaration

```
// C#  
public RdbTransaction BeginTransaction(System.Data.IsolationLevel  
isolationLevel);
```

Parameters

- *isolationLevel*
The isolation level for the new transaction.

Return Value

An `RdbTransaction` object representing the new transaction.

Implements

`IDbConnection`

Exceptions

`InvalidOperationException` - A transaction has already been started.

`ArgumentException` - The `isolationLevel` specified is invalid.

Remarks

The following two isolation levels are supported:

- `System.Data.IsolationLevel.ReadCommitted`
- `System.Data.IsolationLevel.Serializable`

Requesting other isolation levels causes an exception.

Example

```
// C#
// Starts a transaction and inserts one record. If insert fails, rolls
// back the transaction. Otherwise, commits the transaction.
.
.
.
//Create an RdbCommand object using the connection object
RdbCommand cmd = new RdbCommand("", conn);
// Start a transaction
RdbTransaction txn = conn.BeginTransaction(IsolationLevel.ReadCommitted);
try
{
    cmd.CommandText = "insert into mytable values (99, 'foo')";
    cmd.CommandType = CommandType.Text;
    cmd.ExecuteNonQuery();
    txn.Commit();
    Console.WriteLine(
        "Record inserted into the database table.");
}
catch(Exception e)
{
    try
    {
        txn.Rollback();
    }
    catch(Exception e2)
    {
        Console.WriteLine("Problem with rollback: " + e2.ToString());
    }
    Console.WriteLine(
        "Record was NOT inserted into the database table.");
}
.
.
.
```

Note:

Try/Catch exception handling should always be used when rolling back a transaction. A **Rollback** generates an **InvalidOperationException** if the connection is terminated or if the transaction has already been rolled back on the server.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbConnection Members](#)
 - [RdbConnection Class](#)
-

BeginTransaction(String)

This method begins a local transaction with the specified transaction type.

Declaration

```
// C#
public RdbTransaction BeginTransaction(String transactionInfo);
```

Parameters

- *transactionInfo*
The transaction type and other information for the new transaction

Return Value

An `RdbTransaction` object representing the new transaction.

Implements

`IDbConnection`

Exceptions

`InvalidOperationException` - A transaction has already been started.

`ArgumentException` - The *transactionInfo* specified is invalid.

Remarks

This *transactionInfo* string should follow the format of the transaction type specification used by Oracle Rdb SQL SET TRANSACTION statement.

Example

```
// C#
// Starts a transaction and inserts two records. If insert fails, rolls
// back the transaction. Otherwise, commits the transaction.
.
.
.
string ConStr =
    @"User Id=rdb_user;Password=rdb_pw;
    Server=MYNODE:MY_SRV;Database=MY_DBS:MF_PERSONNEL";
RdbConnection conn = new RdbConnection(ConStr);
conn.Open();
//Create an RdbCommand object using the connection object
RdbCommand cmd = new RdbCommand("", conn);

// Start a transaction
RdbTransaction txn = conn.BeginTransaction(
```

```

        "READ WRITE RESERVING MYTABLE FOR EXCLUSIVE WRITE");
try
{
    cmd.CommandText = "insert into mytable values (98, 'foo')";
    cmd.CommandType = CommandType.Text;
    cmd.ExecuteNonQuery();
    cmd.CommandText = "insert into mytable values (99, 'foo2')";
    cmd.ExecuteNonQuery();
    txn.Commit();
    Console.WriteLine(
        "Both records are inserted into the database table.");
}
catch(Exception e)
{
    try
    {
        txn.Rollback();
    }
    catch(Exception e2)
    {
        Console.WriteLine("Problem with rollback: " + e2.ToString());
    }
    Console.WriteLine(
        "Neither record was inserted into the database table.");
}
.
.
.

```

Note:

Try/Catch exception handling should always be used when rolling back a transaction. A **Rollback** generates an **InvalidOperationException** if the connection is terminated or if the transaction has already been rolled back on the server.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbConnection Members](#)
 - [RdbConnection Class](#)
-

ChangeDatabase

This method changes the database that the connection is made to.

Declaration

```
// C#
public void ChangeDatabase(string newDB);
```

Parameters

- *newDB*
The specification of the database to connect to.

Return Value

None.

Implements

IDbConnection

Remarks

Performs the following:

- Rolls back any pending transactions.
- Closes any existing connection to the old database.
- Reopens the connection using the new database specification along with the rest of the current connection properties.

Example

```
// C#
.
.
.
string ConStr =
    @"User Id=rdb_user;Password=rdb_pw;
    Server=MYNODE:MY_SRV;Database=MY_DBS:SUPPORT_PERSONNEL";

RdbConnection conn = new RdbConnection(ConStr);
conn.Open();
.
.
.
//The ChangeDatabase will close the old connection and open a new one
//using the same connection criteria as above but with the Database
//altered
conn.ChangeDatabase("MY_DBS:SALES_PERSONNEL");
.
.
.
conn.Close();
.
.
.
```

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbConnection Members](#)
 - [RdbConnection Class](#)
-

Clone

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

Declaration

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

Return Value

An `RdbConnection` object.

Implements

`Icloneable`

Remarks

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

Example

```
// C#
.
```

```
.  
.br/>RdbConnection conn = new RdbConnection(ConStr);  
conn.Open();  
.br/>.br/>.br/>//Need a proper casting for the return value when cloned  
RdbConnection concloned = (RdbConnection) conn.Clone();  
.br/>.br/>.
```

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbConnection Members](#)
 - [RdbConnection Class](#)
-

Close

If this connection is a Pool Manager, this method releases local resources held by the Pool Manager, otherwise this method closes the connection to the database.

Declaration

```
// C#  
public void Close();
```

Implements

IDbConnection

Remarks

If the connection is a Pool Manager, resources are released, otherwise it performs the following:

- Rolls back any pending transactions.
- Closes the connection to the database. The connection can be reopened using `Open()`.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbConnection Members](#)
 - [RdbConnection Class](#)
-

CreateCommand

This method creates and returns an `RdbCommand` object associated with the `RdbConnection` object.

Declaration

```
// C#  
public RdbCommand CreateCommand();
```

Return Value

The `RdbCommand` object.

Implements

IDbConnection

Example

```
// C#
// Uses connection to create and return an RdbCommand object.
.
.
.
string ConStr =
    @"User Id=rdb_user;Password=rdb_pw;
      Server=MYNODE:MY_SRV;Database=MY_DBS:MF_PERSONNEL;";
RdbConnection conn = new RdbConnection(ConStr);
conn.Open();
RdbCommand cmd = conn.CreateCommand();
cmd.CommandText = "insert into mytable values (99, 'foo')";
cmd.CommandType = CommandType.Text;
cmd.ExecuteNonQuery();
.
.
.
```

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbConnection Members](#)
 - [RdbConnection Class](#)
-

GetSchema

GetSchema methods return schema information for the data source of the RdbConnection. Supported Only in ADO.NET 2.0-Compliant ORDP.NET

Overload List

- [GetSchema\(\)](#)
This method returns schema information for the data source of the RdbConnection.
- [GetSchema \(string collectionName\)](#)
This method returns schema information for the data source of the RdbConnection using the specified string for the collection name.
- [GetSchema \(string collectionName, string\[\] restrictions\)](#)
This method returns schema information for the data source of the RdbConnection using the specified string for the collection name and the specified string array for the restriction values.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbConnection Members](#)
 - [RdbConnection Class](#)
 - [Oracle CorporationRdb Schema Collections](#)
 - [Support for Schema Discovery](#)
 - [User Customization of Metadata](#)
-

GetSchema()

This method returns schema information for the data source of the RdbConnection.

Declaration

```
// C#
public override DataTable GetSchema();
```

Return Value

A DataTable object.

Implements

IDbConnection

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.RdbClient;  
  
class GetSchemaSample  
{  
    static void Main(string[] args)  
    {  
        string constr =  
            @"User Id=rdb_user;Password=rdb_pw;  
            Server=MYNODE:MY_SRV;Database=MY_DBS:MF_PERSONNEL;";  
        string ProviderName = "Oracle.DataAccess.RdbClient";  
  
        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.RdbClient_Schema.xml  
                DataTable dtSchema = conn.GetSchema();  
                dtSchema.WriteXml(ProviderName + "_Schema.xml");  
            }  
            catch (Exception ex)  
            {  
                Console.WriteLine(ex.Message);  
                Console.WriteLine(ex.StackTrace);  
            }  
        }  
    }  
}
```

See Also:

-
- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbConnection Members](#)
 - [RdbConnection Class](#)
 - [Oracle Corporation Rdb Schema Collections](#)
 - [Support for Schema Discovery](#)
 - [User Customization of Metadata](#)
-

GetSchema (string collectionName)

This method returns schema information for the data source of the `RdbConnection` 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.

Implements

`IDbConnection`

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.RdbClient;  
  
class GetSchemaSample  
{  
    static void Main(string[] args)  
    {  
        string constr =  
            @"User Id=rdb_user;Password=rdb_pw;  
            Server=MYNODE:MY_SRV;Database=MY_DBS:MF_PERSONNEL;";  
        string ProviderName = "Oracle.DataAccess.RdbClient";  
        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);
}
}
}
}

```

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbConnection Members](#)
 - [RdbConnection Class](#)
 - [Oracle Corporation Rdb Schema Collections](#)
 - [Support for Schema Discovery](#)
 - [User Customization of Metadata](#)
-

GetSchema (string collectionName, string[] restrictions)

This method returns schema information for the data source of the `RdbConnection` 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*
Name of the collection for which metadata is required.
- *restrictions*
An array of restrictions that apply to the metadata being retrieved.

Return Value

A `DataTable` object.

Implements

`IDbConnection`

Exceptions

`ArgumentException` – The requested collection is not defined.

`InvalidOperationException` – The connection is closed.

`InvalidOperationException` – The requested collection is not supported by current version of the Oracle Rdb database.

`InvalidOperationException` – No population string is specified for requested collection.

`InvalidOperationException` – More restrictions were provided than the requested collection supports.

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 method returns a `DataTable` that contains only rows from the specified metadata collection that match the specified restrictions.

The restrictions depend on the collection you are trying to retrieve. The .NET standard collection restrictions are documented in your .NET Framework documentation. Valid restrictions for collections specific to ORDP may be found under the appropriate collection sub-headings within [ORDP.NET-Specific Schema Collection](#).

Valid restrictions may also be retrieved from the `DbMetaDataCollectionNames.Restrictions` collection.

```
DataTable dtRestrictions =  
    conn.GetSchema (DbMetaDataCollectionNames.Restrictions);
```

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.

The `collectionName` parameter is not case-sensitive, but `restrictions` (string values) are.

Note:

Oracle Rdb SQL wildcard characters ("% and "_") may be used in string parameter values. ORDP.NET will use the SQL `LIKE` operator to search for the values provided.

Example

```
// C#

using System;
using System.Data;
using System.Data.Common;
using Oracle.DataAccess.RdbClient;

class GetSchemaSample
{
    static void Main(string[] args)
    {
        string constr =
            @"User Id=rdb_user;Password=rdb_pw;
            Server=MYNODE:MY_SRV;Database=MY_DBS:MF_PERSONNEL;";
        string ProviderName = "Oracle.DataAccess.RdbClient";

        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;

                // now filter out just the 'Columns' restrictions
                dv.RowFilter = "CollectionName = 'Columns'";
                dv.Sort = "RestrictionNumber";

                // save the new sorted filtered table
                dtRestrictions = dv.ToTable();

                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[2];

                // clear collection
                for (int i = 0; i < 2; i++)
                    restrictions[i] = null;

                //Get all columns from all tables with names starting with "EMP"
                restrictions[0] = "EMP%";
                DataTable dtAllEmpCols = conn.GetSchema("Columns", restrictions);

                // clear collection
                for (int i = 0; i < 2; i++)
                    restrictions[i] = null;
            }
        }
    }
}
```


-
- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbConnection Members](#)
 - [RdbConnection Class](#)
-

IsBroken

This method returns `true` if the connection state of the database is `ConnectionState.Broken` otherwise it returns `false`.

Declaration

```
// C#  
public bool IsBroken();
```

Implements

IDbConnection

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbConnection Members](#)
 - [RdbConnection Class](#)
-

Open

If this is a Pool Manager connection, this method establishes the criteria for connection pooling. Otherwise, this method opens a connection to an Rdb database.

Declaration

```
// C#  
public void Open();
```

Implements

IDbConnection

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - The connection is already opened or the connection string is null or empty.

Remarks

If the attributes within the `ConnectionString` property specify that the connection is a Pool Manager, the attributes for this connection will be used to establish a connection pool. Otherwise, a new connection is established.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbConnection Members](#)
 - [RdbConnection Class](#)
-

EnlistTransaction()

This method enlists the connection to the current system transaction.

Declaration

```
// C#  
public void EnlistTransaction();
```

Implements

IDbConnection

Exceptions

NotSupportedException – A local transaction has already been started.

Remarks

Subsequent database operations will be carried out within the scope of the current system transaction. As ORDP.NET does not currently support distributed transactions, a local default transaction will be declared.

EnlistTransaction may be used in conjunction with TransactionScope.

The current transaction may be completed by:

- Issuing an explicit Commit or Rollback SQL statement on the connection or
- The Dispose of the current TransactionScope or
- Calling the Complete() method of the current TransactionScope object
- A commit or rollback notification issued to the associate enlistment

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbConnection Members](#)
 - [RdbConnection Class](#)
-

EnlistTransaction(transaction)

This method enlists the connection to the specified transaction.

Declaration

```
// C#  
public void EnlistTransaction(System.Transactions.Transaction  
transaction);
```

Implements

IDbConnection

Exceptions

NotSupportedException – A local transaction has already been started.

Remarks

Subsequent database operations will be carried out within the scope of the specified transaction. As ORDP.NET does not currently support distributed transactions, a local default transaction will be declared.

The current transaction may be completed by:

- A commit or rollback notification issued to the associate enlistment

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbConnection Members](#)
 - [RdbConnection Class](#)
-

SetDatabase

This Method sets a name or file specification that identifies an Oracle Rdb database instance.

Declaration

```
// C#  
public void SetDatabase ( string DBspec )
```

Property Value

The Oracle Rdb database file specification.

Exceptions

`InvalidOperationException` - `ConnectionString` is being set while the connection is open.

Remarks

The default is an empty string.

The connection must be closed prior to calling this method.

If the database is an empty string, the default database for the specified server will be used. If no default database is associated with the server, an exception will be raised.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbConnection Members](#)
 - [RdbConnection Class](#)
-

4.2.3.6 RdbConnection Events

`RdbConnection` events are listed in [Table 4-15](#).

InfoMessage

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

Declaration

```
// C#  
public event RdbInfoMessageEventHandler InfoMessage;
```

Event Data

The event handler receives an `RdbInfoMessageEventArgs` 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 `RdbInfoMessageEventHandler` delegate to listen to this event.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbConnection Members](#)
 - [RdbConnection Class](#)
-

StateChange

This event is triggered when the connection state changes.

Declaration

```
// C#  
public event StateChangeEventHandlers StateChange;
```

Event Data

The event handler receives a `StateChangeEventArgs` object, which exposes the following properties containing information about the event.

- `CurrentState`
The new state of the connection.
- `OriginalState`
The original state of the connection.

Remarks

The `StateChange` event is raised after a connection changes state, whenever an explicit call is made to `Open`, `Close` or `Dispose`.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbConnection Members](#)
 - [RdbConnection Class](#)
-

4.2.3.7 RdbConnection Event Delegates

`RdbConnection` event delegates are listed in [Table 4–16](#).

RdbInfoMessageEventHandler

This event delegate handles the `InfoMessage` event.

StateChangeEventHandler

This event delegate handles the `StateChange` event.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbConnection Members](#)
 - [RdbConnection Class](#)
 - Microsoft ADO.NET documentation for a description of `StateChangeEventHandler`
-

4.2.4 RdbDataAdapter Class

An `RdbDataAdapter` object represents a data provider object that populates the `DataSet` and updates changes in the `DataSet` to the Oracle Rdb database.

Class Inheritance

```
Object
  MarshalByRefObject
    Component
      DataAdapter
        DbDataAdapter
          RdbDataAdapter
```

Declaration

```
// C#
public sealed class RdbDataAdapter : DbDataAdapter, IDbDataAdapter
```

Thread Safety

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

Example

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

```
CREATE TABLE empInfo (
  empno NUMBER(4) PRIMARY KEY,
  empName VARCHAR(20) NOT NULL,
  hiredate DATE ANSI,
  salary NUMBER(7,2)
);
```

The `EMPINFO` table has the following values:

EMPNO	EMPNAME	HIREDATE	SALARY
1	KING	01-MAY-81	12345.67
2	SCOTT	01-SEP-75	34567.89
3	BLAKE	01-OCT-90	9999.12
4	SMITH	NULL	NULL

The following example uses the `RdbDataAdapter` and the dataset to update the `EMPINFO` table:

```
// C#
public static void AdapterUpdate(string connStr)
{
  string cmdStr = "SELECT EMPNO, EMPNAME, SALARY FROM EMPINFO";
  //create the adapter with the selectCommand txt and the
  //connection string
  RdbDataAdapter adapter = new RdbDataAdapter(cmdStr, connStr);
  //get the connection from the adapter
  RdbConnection connection = adapter.SelectCommand.Connection;
  //create the UpdateCommand object for updating the EMPINFO table
  //from the dataset
  adapter.UpdateCommand = new RdbCommand(
    "UPDATE EMPINFO SET SALARY = :iSALARY where EMPNO = :iEMPNO",
    connection);
  adapter.UpdateCommand.Parameters.Add(":iSALARY", DbType.Double,
    0, "SALARY");
```

```

adapter.UpdateCommand.Parameters.Add(":iEMPNO", DbType.Int16,
    0, "EMPNO");
//Create and fill the DataSet using the EMPINFO
DataSet dataset = new DataSet();
adapter.Fill(dataset, "EMPINFO");
//Get the EMPINFO table from the dataset
DataTable table = dataset.Tables["EMPINFO"];
//Get the first row from the EMPINFO table
DataRow row0 = table.Rows[0];
//update the salary in the first row
row0["SALARY"] = 99999.99;
//Now update the EMPINFO using the adapter, the salary
//of 'KING' is changed to 99999.99
adapter.Update(dataset, "EMPINFO");
}

```

Requirements

Namespace: Oracle.DataAccess.RdbClient

Assembly: Rdb.DataAccess.Rdb.dll

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
- [RdbDataAdapter Members](#)
- [RdbDataAdapter Constructors](#)
- [RdbDataAdapter Static Methods](#)
- [RdbDataAdapter Properties](#)
- [RdbDataAdapter Public Methods](#)
- [RdbDataAdapter Events](#)
- [RdbDataAdapter Event Delegates](#)

4.2.4.1.1 RdbDataAdapter Members

`RdbDataAdapter` members are listed in the following tables:

RdbDataAdapter Constructors

`RdbDataAdapter` constructors are listed in [Table 4-18](#).

Table 4-18 RdbDataAdapter Constructors

Constructor	Description
RdbDataAdapter Constructors	Instantiates a new instance of <code>RdbDataAdapter</code> class (Overloaded).

RdbDataAdapter Static Methods

`RdbDataAdapter` static methods are listed in [Table 4-19](#).

Table 4-19 RdbDataAdapter Static Methods

Constructor	Description
<code>Equals</code>	Inherited from <code>Object</code> (Overloaded).

RdbDataAdapter Properties

`RdbDataAdapter` properties are listed in [Table 4-20](#).

Table 4-20 RdbDataAdapter Properties

Name	Description
AcceptChangesDuringFill	Inherited from DataAdapter .
ContinueUpdateOnError	Inherited from DataAdapter .
DeleteCommand	A SQL statement or stored procedure to delete rows from an Rdb database.
InsertCommand	A SQL statement or stored procedure to insert new rows into an Rdb database .
MissingMappingAction	Inherited from DataAdapter .
MissingSchemaAction	Inherited from DataAdapter .
SelectCommand	A SQL statement or stored procedure that returns a single or multiple result set.
TableMappings	Inherited from DataAdapter .
UpdateCommand	A SQL statement or stored procedure to update rows from the DataSet to an Rdb database.

RdbDataAdapter Public Methods

RdbDataAdapter public methods are listed in [Table 4-21](#).

Table 4-21 RdbDataAdapter Public Methods

Public Method	Description
Dispose	Inherited from Component .
Equals	Inherited from Object (Overloaded).
Fill	Inherited from DbDataAdapter .
FillSchema	Inherited from DbDataAdapter .
GetFillParameters	Inherited from DbDataAdapter .
GetHashCode	Inherited from Object .
GetType	Inherited from Object .
ToString	Inherited from Object .
Update	Inherited from DbDataAdapter .

RdbDataAdapter Events

RdbDataAdapter events are listed in [Table 4-22](#).

Table 4-22 RdbDataAdapter Events

Event	Description
Disposed	Inherited from Component .
FillError	Inherited from DbDataAdapter .
RdbRowUpdated	This event is raised when row(s) have been updated by the Update () .
RdbRowUpdating	This event is raised when row data are about to be updated to the database.

RdbDataAdapter Event Delegates

RdbDataAdapter event delegates are listed in [Table 4-23](#).

Table 4-23 RdbDataAdapter Events Delegates

Event Delegate Name	Description
---------------------	-------------

<code>EventHandler</code>	Inherited from <code>Component</code> .
<code>FillErrorEventHandler</code>	Inherited from <code>DbDataAdapter</code> .
RdbRowUpdatedEventHandler	Event Delegate for the <code>RowUpdated</code> Event .
RdbRowUpdatingEventHandler	Event Delegate for the <code>RowUpdating</code> Event.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbDataAdapter Members](#)
 - [RdbDataAdapter Constructors](#)
 - [RdbDataAdapter Static Methods](#)
 - [RdbDataAdapter Properties](#)
 - [RdbDataAdapter Public Methods](#)
 - [RdbDataAdapter Events](#)
 - [RdbDataAdapter Event Delegates](#)
-

4.2.4.1.2 RdbDataAdapter Constructors

`RdbDataAdapter` constructors create new instances of an `RdbDataAdapter` class.

Overload List:

- [RdbDataAdapter\(\)](#)
This constructor creates an instance of an `RdbDataAdapter` class.
- [RdbDataAdapter\(RdbCommand\)](#)
This constructor creates an instance of an `RdbDataAdapter` class with the provided `RdbCommand` as the `SelectCommand`.
- [RdbDataAdapter\(string, RdbConnection\)](#)
This constructor creates an instance of an `RdbDataAdapter` class with the provided `RdbConnection` object and the command text for the `SelectCommand`.
- [RdbDataAdapter\(string, string\)](#)
This constructor creates an instance of an `RdbDataAdapter` class with the provided connection string and the command text for the `SelectCommand`.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbDataAdapter Members](#)
 - [RdbDataAdapter Class](#)
-

RdbDataAdapter()

This constructor creates an instance of an `RdbDataAdapter` class with no arguments.

Declaration

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

Remarks

Initial values are set for the following `RdbDataAdapter` properties as indicated:

- `MissingMappingAction` = `MissingMappingAction.Passthrough`
- `MissingSchemaAction` = `MissingSchemaAction.Add`

See Also:

-
- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbDataAdapter Members](#)
 - [RdbDataAdapter Class](#)
-

RdbDataAdapter(RdbCommand)

This constructor creates an instance of an `RdbDataAdapter` class with the provided `RdbCommand` as the `SelectCommand`.

Declaration

```
// C#  
public RdbDataAdapter(RdbCommand selectCommand);
```

Parameters

- `selectCommand`
The `RdbCommand` that is to be set as the `SelectCommand` property.

Remarks

Initial values are set for the following `RdbDataAdapter` properties as indicated:

- `MissingMappingAction` = `MissingMappingAction.Passthrough`
- `MissingSchemaAction` = `MissingSchemaAction.Add`

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbDataAdapter Members](#)
 - [RdbDataAdapter Class](#)
-

RdbDataAdapter(string, RdbConnection)

This constructor creates an instance of an `RdbDataAdapter` class with the provided `RdbConnection` object and the command text for the `SelectCommand`.

Declaration

```
// C#  
public RdbDataAdapter(string selectCommandText, RdbConnection  
selectConnection);
```

Parameters

- `selectCommandText`
The string that is set as the `CommandText` of the `SelectCommand` property of the `RdbDataAdapter`.
- `selectConnection`
The `RdbConnection` to connect to the Rdb database.

Remarks

The `RdbDataAdapter` 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 `RdbDataAdapter` properties as indicated:

- `MissingMappingAction` = `MissingMappingAction.Passthrough`
- `MissingSchemaAction` = `MissingSchemaAction.Add`

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbDataAdapter Members](#)
 - [RdbDataAdapter Class](#)
-

RdbDataAdapter(string, string)

This constructor creates an instance of an `RdbDataAdapter` class with the provided connection string and the command text for the `SelectCommand`.

Declaration

```
// C#
public RdbDataAdapter(string selectCommandText, string
selectConnectionString);
```

Parameters

- `selectCommandText`
The string that is set as the `CommandText` of the `SelectCommand` property of the `RdbDataAdapter`.
- `selectConnectionString`
The connection string.

Remarks

Initial values are set for the following `RdbDataAdapter` properties as indicated:

- `MissingMappingAction` = `MissingMappingAction.Passthrough`
- `MissingSchemaAction` = `MissingSchemaAction.Add`

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbDataAdapter Members](#)
 - [RdbDataAdapter Class](#)
-

4.2.4.1.3 RdbDataAdapter Static Methods

`RdbDataAdapter` static methods are listed in [Table 4-19](#).

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbDataAdapter Members](#)
 - [RdbDataAdapter Class](#)
-

4.2.4.1.4 RdbDataAdapter Properties

`RdbDataAdapter` properties are listed in [Table 4-20](#).

DeleteCommand

This property is a SQL statement or stored procedure to delete rows from an Rdb database.

Declaration

```
// C#
public RdbCommand DeleteCommand {get; set;}
```

Property Value

An `RdbCommand` used during the `Update` call to delete rows from tables in the Rdb 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 `RdbCommandBuilder`, if no command is provided for this.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbDataAdapter Members](#)
 - [RdbDataAdapter Class](#)
-

InsertCommand

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

Declaration

```
// C#  
public RdbCommand InsertCommand {get; set;}
```

Property Value

An `RdbCommand` 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 `RdbCommandBuilder`, if no command is provided for this property.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbDataAdapter Members](#)
 - [RdbDataAdapter Class](#)
-

SelectCommand

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

Declaration

```
// C#  
public RdbCommand SelectCommand {get; set;}
```

Property Value

An `RdbCommand` used during the `Fill` call to populate the selected rows to the `DataSet`.

Remarks

Default = null

If the `SelectCommand` does not return any rows, no tables are added to the dataset and no exception is raised.

If the `SELECT` statement selects from a `VIEW`, no key information is retrieved when a `FillSchema()` or a `Fill()` with `MissingSchemaAction.AddWithKey` is invoked.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbDataAdapter Members](#)
 - [RdbDataAdapter Class](#)
-

UpdateCommand

This property is a SQL statement or stored procedure to update rows from the `DataSet` to an Rdb database.

Declaration

```
// C#  
public RdbCommand UpdateCommand {get; set;}
```

Property Value

An `RdbCommand` used during the `Update` call to update rows in the Rdb 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 `RdbCommandBuilder`, if no command is provided for this property.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbDataAdapter Members](#)
 - [RdbDataAdapter Class](#)
-

4.2.4.1.5 RdbDataAdapter Public Methods

`RdbDataAdapter` public methods are listed in [Table 4-21](#).

4.2.4.1.6 RdbDataAdapter Events

`RdbDataAdapter` events are listed in [Table 4-22](#).

RowUpdated

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

Declaration

```
// C#  
public event RdbRowUpdatedEventHandler RowUpdated;
```

EventData

The event handler receives an `RdbRowUpdatedEventArgs` object that exposes the following properties containing information about the event.

- `Command`
The `RdbCommand` executed during the `Update`.
- `Errors` (inherited from `RowUpdatedEventArgs`)
The exception, if any, is generated during the `Update`.
- `RecordsAffected` (inherited from `RowUpdatedEventArgs`)
The number of rows modified, inserted, or deleted by the execution of the `Command`.
- `Row` (inherited from `RowUpdatedEventArgs`)
The `DataRow` sent for `Update`.

- `StatementType` (inherited from `RowUpdatedEventArgs`)
The type of SQL statement executed.
- `Status` (inherited from `RowUpdatedEventArgs`)
The `UpdateStatus` of the Command.
- `TableMapping` (inherited from `RowUpdatedEventArgs`)
The `DataTableMapping` used during the Update.

Example

The following example shows how to use the `RowUpdating` and `RowUpdated` events.

```
// C#
// create the event handler for RowUpdating event
protected static void OnRowUpdating(object sender,
RdbRowUpdatingEventArgs e)
{
    Console.WriteLine("Row updating....");
    Console.WriteLine("Event arguments:");
    Console.WriteLine("Command Text: " + e.Command.CommandText);
    Console.WriteLine("Command Type: " + e.StatementType);
    Console.WriteLine("Status: " + e.Status);
}

// create the event handler for RowUpdated event
protected static void OnRowUpdated(object sender,
RdbRowUpdatedEventArgs e)
{
    Console.WriteLine("Row updated....");
    Console.WriteLine("Event arguments:");
    Console.WriteLine("Command Text: " + e.Command.CommandText);
    Console.WriteLine("Command Type: " + e.StatementType);
    Console.WriteLine("Status: " + e.Status);
}

public static void AdapterEvents(string connStr)
{
    string cmdStr = "SELECT EMPNO, EMPNAME, SALARY FROM EMPINFO";
    //create the adapter with the selectCommand txt and the
    //connection string
    RdbDataAdapter adapter = new RdbDataAdapter(cmdStr, connStr);
    //get the connection from the adapter
    RdbConnection connection = adapter.SelectCommand.Connection;
    //create the UpdateCommand object for updating the EMPINFO table
    //from the dataset
    adapter.UpdateCommand = new RdbCommand(
        "UPDATE EMPINFO SET SALARY = :iSALARY where EMPNO = :iEMPNO",
        connection);
    adapter.UpdateCommand.Parameters.Add(":iSALARY", DbType.Double,
0, "SALARY");
    adapter.UpdateCommand.Parameters.Add(":iEMPNO", DbType.Int16,
0, "EMPNO");
    //Create and fill the DataSet using the EMPINFO
    DataSet dataset = new DataSet();
    adapter.Fill(dataset, "EMPINFO");
    //Get the EMPINFO table from the dataset
    DataTable table = dataset.Tables["EMPINFO"];

    //Get the first row from the EMPINFO table
    DataRow row0 = table.Rows[0];
    //update the salary in the first row
    row0["SALARY"] = 99999.99;
    //set the event handlers for the RowUpdated and the RowUpdating event
```

```

//the OnRowUpdating() method will be triggered before the update, and
//the OnRowUpdated() method will be triggered after the update
adapter.RowUpdating += new RdbRowUpdatingEventHandler(OnRowUpdating);
adapter.RowUpdated += new RdbRowUpdatedEventHandler(OnRowUpdated);
//Now update the EMPINFO using the adapter, the salary
//of 'KING' is changed to 99999.99
//The OnRowUpdating() and the OnRowUpdated() methods
//will be triggered
adapter.Update(dataset, "EMPINFO");
}

```

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
- [RdbDataAdapter Members](#)
- [RdbDataAdapter Class](#)

RowUpdating

This event is raised when row data are about to be updated to the database.

Declaration

```

// C#
public event RdbRowUpdatingEventHandler RowUpdating;

```

Event Data

The event handler receives an `RdbRowUpdatingEventArgs` object, which exposes the following properties containing information about the event.

- `Command`
The `RdbCommand` 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](#)".

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
- [RdbDataAdapter Members](#)
- [RdbDataAdapter Class](#)

4.2.4.1.7 RdbDataAdapter Event Delegates

`RdbDataAdapter` event delegates are listed in [Table 4-23](#).

RdbRowUpdatedEventHandler

This event delegate handles the `RowUpdated` Event.

RdbRowUpdatingEventHandler

This event delegate handles the `RowUpdating` Event.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbDataAdapter Members](#)
 - [RdbDataAdapter Class](#)
-

4.2.5 RdbDataReader Class

An `RdbDataReader` object represents a forward-only, read-only, in-memory result set. Unlike the `DataSet`, the `RdbDataReader` stays connected and fetches one row at a time.

Class Inheritance

```
Object
  MarshalByRefObject
    RdbDataReader
```

Declaration

```
// C#
public sealed class RdbDataReader : IDataReader
```

Thread Safety

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

Remarks

An `RdbDataReader` instance is constructed by a call to the `ExecuteReader` method of the `RdbCommand` object.

When the `DataReader` is closed or has been disposed, the only properties that can be accessed are `IsClosed` and `RecordsAffected`.

Example

The `RdbDataReader` examples in this section are based on the `CURRENT_INFO` view from `MF_PERSONNEL`.

The following example retrieves the data from the `CURRENT_INFO` view:

```
//C#
//This method retrieves data from CURRENT_INFO view:
public void ReadEmpInfo(string connStr)
{
    string cmdStr = "SELECT * FROM CURRENT_INFO LIMIT TO 10 ROWS";
    RdbConnection connection = new RdbConnection(connStr);
    RdbCommand cmd = new RdbCommand(cmdStr, connection);
    connection.Open();
    RdbDataReader reader = cmd.ExecuteReader();
    //declare the variables to retrieve the data in CURRENT_INFO view
    short empNo;
    string empName;
    DateTime jobDate;
    double salary;
    string dept;
    int idx;
    //read the next row until end of row
    while (reader.Read())
    {
```

```

// note the automatic conversion from string to numeric
empNo = reader.GetInt16(reader.GetOrdinal("ID"));
Console.WriteLine("Employee number: " + empNo);
empName = reader.GetString(reader.GetOrdinal("LAST_NAME"));
Console.WriteLine("Employee name: " + empName);
//the following columns can have NULL value, so it
//is important to call IsDBNull before getting the column data
idx = reader.GetOrdinal("JSTART");
if (!reader.IsDBNull(idx))
{
    jobDate = reader.GetDateTime(idx);
    Console.WriteLine("Job Start date: " + jobDate);
}
idx = reader.GetOrdinal("SALARY");
if (!reader.IsDBNull(idx))
{
    salary = reader.GetDouble(idx);
    Console.WriteLine("Salary: " + salary);
}
idx = reader.GetOrdinal("DEPARTMENT");
if (!reader.IsDBNull(idx))
{
    dept = reader.GetString(idx);
    Console.WriteLine("Department: " + dept);
}
Console.WriteLine();
//done reading one row
} //Done Reading view
//Close the reader
reader.Close();
// Dispose of the command
cmd.Dispose();
// Close the connection
connection.Close();
}

```

Requirements

Namespace: Oracle.DataAccess.RdbClient

Assembly: Rdb.DataAccess.Rdb.dll

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
- [RdbDataReader Members](#)
- [RdbDataReader Static Methods](#)
- [RdbDataReader Properties](#)
- [RdbDataReader Public Methods](#)
- [RdbDataReader SchemaTable](#)

4.2.5.1 RdbDataReader Members

RdbDataReader members are listed in the following tables:

RdbDataReader Static Methods

RdbDataReader static methods are listed in [Table 4-24](#).

Table 4-24 RdbDataReader Static Methods

Methods	Description
Equals	Inherited from Object (Overloaded)

RdbDataReader Properties

RdbDataReader properties are listed in [Table 4-25](#).

Table 4-25 RdbDataReader Properties

Property	Description
FetchSize	Specifies the size of the RdbDataReader internal cache.
FieldCount	Gets the number of columns in the result set.
IsClosed	Indicates whether the data reader is closed.
Item	Gets the value of the column (Overloaded).
RecordsAffected	Gets the number of rows changed, inserted, or deleted by execution of the SQL statement.

RdbDataReader Public Methods

RdbDataReader public methods are listed in [Table 4-26](#).

Table 4-26 RdbDataReader Public Methods

Public Method	Description
Close	Closes the RdbDataReader.
CreateObjRef	Inherited from MarshalByRefObject.
Dispose	Releases any resources or memory allocated by the object.
Equals	Inherited from Object (Overloaded).
GetBoolean	<i>Not Supported.</i>
GetBlobAsString	Return the String value of the specified Blob column (Overloaded).
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	<i>Not Supported.</i>
GetChars	Populates the provided character array with up to the maximum number of characters, from the specified offset (in characters) of the column.
GetData	<i>Not Supported.</i>
GetDataTypeName	Returns the .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 BINARYDOUBLE column.
GetFieldType	Returns the Type of the specified column.
GetFloat	Returns the float value of the specified NUMBER column or BINARYFLOAT column.
GetGuid	<i>Not Supported.</i>
GetHashCode	Inherited from Object.
GetInt16	Returns the Int16 value of the specified NUMBER column.
GetInt32	Returns the Int32 value of the specified NUMBER column.
GetInt64	Returns the Int64 value of the specified NUMBER column.
GetName	Returns the name of the specified column.
GetOrdinal	Returns the 0-based ordinal (or index) of the specified column name.
GetSchemaTable	Returns a DataTable that describes the column metadata of the RdbDataReader.

GetString	Returns the string value of the specified column.
GetType	Inherited from <code>Object</code> class.
GetValue	Returns the column value as a .NET type.
GetValues	Gets all the column values as .NET types.
IsDBNull	Indicates whether the column value is null.
NextResult	Advances the data reader to the next result set when reading the results.
Read	Advances the data reader to the next record when reading the results.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbDataReader Members](#)
 - [RdbDataReader Static Methods](#)
 - [RdbDataReader Properties](#)
 - [RdbDataReader Public Methods](#)
 - [RdbDataReader SchemaTable](#)
-

4.2.5.2 RdbDataReader Static Methods

`RdbDataReader` static methods are listed in [Table 4-24](#).

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbDataReader Class](#)
 - [RdbDataReader Members](#)
-

4.2.5.3 RdbDataReader Properties

`RdbDataReader` public methods are listed in [Table 4-25](#).

FetchSize

This property specifies the number of records to be stored in the `RdbDataReader` internal cache.

Declaration

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

Property Value

An `int` that specifies the number of records that the `RdbDataReader` will store in its internal cache.

Exceptions

`ArgumentOutOfRangeException` - The `FetchSize` value specified is invalid; it must be greater than 0.

Remarks

Default = The `RdbCommand` `FetchSize` property value.

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

See Also:

-
- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbDataReader Class](#)
 - [RdbDataReader Members](#)
-

FieldCount

This property gets the number of columns in the result set.

Declaration

```
// C#  
public int FieldCount {get;}
```

Property Value

The number of columns in the result set if one exists, otherwise 0.

Implements

IDataRecord

Exceptions

InvalidOperationException - The reader is closed.

Remarks

Default = 0

This property has a value of 0 for queries that do not return result sets.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbDataReader Class](#)
 - [RdbDataReader Members](#)
-

IsClosed

This property indicates whether the data reader is closed.

Declaration

```
// C#  
public bool IsClosed {get;}
```

Property Value

If the `RdbDataReader` 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 `RdbDataReader` is closed.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbDataReader Class](#)
 - [RdbDataReader Members](#)
-

Item

This property gets the value of the column in .NET datatype.

Overload List:

- [Item \[index\]](#)
This property gets the .NET Value of the column specified by the column index.
- [Item \[string\]](#)
This property gets the .NET Value of the column specified by the column name.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbDataReader Class](#)
 - [RdbDataReader Members](#)
-

Item [index]

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

Declaration

```
// C#  
public object this[int index] {get;}
```

Parameters

- *index*
The zero-based index of the column.

Property Value

The .NET value of the specified column.

Implements

IDataRecord

Remarks

Default = Not Applicable
In C#, this property is the indexer for this class.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbDataReader Class](#)
 - [RdbDataReader Members](#)
-

Item [string]

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

Declaration

```
// C#  
public object this[string columnName] {get;}
```

Parameters

- *columnName*
The name of the column.

Property Value

The .NET Value of the specified column.

Implements

IDataRecord

Remarks

Default = Not Applicable

A case-sensitive search is made to locate the specified column by its name. If this fails, then a case-insensitive search is made.

In C#, this property is the indexer for this class.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbDataReader Class](#)
 - [RdbDataReader Members](#)
-

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 RdbDataReader is closed.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbDataReader Class](#)
 - [RdbDataReader Members](#)
-

4.2.5.4 RdbDataReader Public Methods

RdbDataReader public methods are listed in [Table 4-26](#).

Close

This method closes the RdbDataReader.

Declaration

```
// C#  
public void Close();
```

Implements

IDataReader

Remarks

The `Close` method frees all resources associated with the `RdbDataReader`.

Example

The code example for the `RdbDataReader` class includes the `Close` method. See "[Example](#)" in the `RdbDataReader` class section.

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 `RdbDataReader`.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbDataReader Class](#)
 - [RdbDataReader Members](#)
-

GetBlobAsString

`GetBlobAsString` returns the specified `Blob` column's value as a `String`. If the column value is not a `Blob`, the returned value will be `value.ToString()`.

Overload List:

- [GetBlobAsString\(int index\)](#)
This method returns the specified `Blob` column's value as a `String`; no separator will be inserted between segments.
- [GetBlobAsString\(int index, String separator\)](#)
This method returns the specified `Blob` column's value as a `String` taking into account the separator value.

GetBlobAsString(int index)

This method returns the specified `Blob` column's value as a `String`.

Declaration

```
// C#  
public String GetBlobAsString(int index);
```

Parameters

- *index*
The zero-based column index.

Return Value

The value of the column as a `String`.

Implements

`IDataRecord`

Exceptions

`InvalidOperationException` - The connection is closed, the reader is closed, `Read()` has not been called, or all rows have been read.
`IndexOutOfRangeException` - The column index is invalid.
`InvalidCastException` - The accessor method is invalid for this column type or the column value is `NULL`.

Remarks

`IsDBNull` should be called to check for `NULL` values before calling this method.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbDataReader Class](#)
 - [RdbDataReader Members](#)
-

GetBlobAsString(int index, String separator)

This method returns the specified `Blob` column's value as a `String`. If the original `Rdb Segmented String` data has multiple segments the separator value will be inserted between each segment prior to building the resultant `String`.

Declaration

```
// C#  
public String GetBlobAsString(int index, String separator);
```

Parameters

- *index*
The zero-based column index.
- *separator*
The separator `String` value to use between `Rdb Segmented String` segments.

Return Value

The value of the column as a `String` after inserting the provided separator between each segment.

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.

Example

This example uses a separator to place a new line between each segmented string segment.

```
// C#  
.  
.  
.
```

```

//get the reader
RdbDataReader reader = cmd.ExecuteReader();
while (reader.Read())
{
    if (!reader.IsDBNull(0))
    {
        Console.WriteLine(reader.GetString(0));
        Console.WriteLine("Now read with separator ...");
        Console.WriteLine(reader.GetBlobAsString(0, "\n"));
    }
}
reader.Close();
.
.
.

```

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbDataReader Class](#)
 - [RdbDataReader Members](#)
-

GetByte

This method returns the byte value of the specified column.

Declaration

```

// C#
public byte GetByte(int index);

```

Parameters

- *index*
The zero-based column index.

Return Value

The value of the column as a byte.

Implements

IDataRecord

Exceptions

InvalidOperationException - The connection is closed, the reader is closed, Read() has not been called, or all rows have been read.

IndexOutOfRangeException - The column index is invalid.

InvalidCastException - The accessor method is invalid for this column type or the column value is NULL.

Remarks

IsDBNull should be called to check for NULL values before calling this method.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbDataReader Class](#)
 - [RdbDataReader Members](#)
-

GetBytes

This method populates the provided byte array with up to the maximum number of bytes, from the specified offset (in bytes) of the column.

Declaration

```
// C#  
public long GetBytes(int index, long fieldOffset, byte[] buffer, int  
bufferOffset, int length);
```

Parameters

- *index*
The zero-based column index.
- *fieldOffset*
The offset within the column from which reading begins (in bytes).
- *buffer*
The byte array that the data is read into.
- *bufferOffset*
The offset within the buffer to begin reading data into (in bytes).
- *length*
The maximum number of bytes to read (in bytes).

Return Value

The number of bytes read.

Implements

IDataRecord

Exceptions

InvalidOperationException - The connection is closed, the reader is closed, `Read()` has not been called, or all rows have been read.

IndexOutOfRangeException - The column index is invalid.

InvalidCastException - The accessor method is invalid for this column type or the column value is NULL.

Remarks

This method returns the number of bytes read into the buffer. This may be less than the actual length of the field if the method has been called previously for the same column.

If a null reference is passed for `buffer`, the length of the field in bytes is returned.

`IsDBNull` should be called to check for NULL values before calling this method.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbDataReader Class](#)
 - [RdbDataReader Members](#)
-

GetChars

This method populates the provided character array with up to the maximum number of characters, from the specified offset (in characters) of the column.

Declaration

```
// C#
```

```
public long GetChars(int index, long fieldOffset, char[] buffer, int
bufferOffset, int length);
```

Parameters

- *index*
The zero based column index.
- *fieldOffset*
The index within the column from which to begin reading (in characters).
- *buffer*
The character array that the data is read into.
- *bufferOffset*
The index within the buffer to begin reading data into (in characters).
- *length*
The maximum number of characters to read (in characters).

Return Value

The number of characters read.

Implements

IDataRecord

Exceptions

InvalidOperationException - The connection is closed, the reader is closed, `Read()` has not been called, or all rows have been read.

IndexOutOfRangeException - The column index is invalid.

InvalidCastException - The accessor method is invalid for this column type or the column value is NULL.

Remarks

This method returns the number of characters read into the buffer. This may be less than the actual length of the field, if the method has been called previously for the same column.

If a null reference is passed for `buffer`, the length of the field in characters is returned.

`IsDBNull` should be called to check for NULL values before calling this method.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbDataReader Class](#)
 - [RdbDataReader Members](#)
-

GetDataTypeName

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

Declaration

```
// C#
public string GetDataTypeName(int index);
```

Parameters

- *index*
The zero-based column index.

Return Value

The name of the .NET type of the column.

Implements

IDataRecord

Exceptions

`InvalidOperationException` - The reader is closed.

`IndexOutOfRangeException` - The column index is invalid.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbDataReader Class](#)
 - [RdbDataReader Members](#)
-

GetDateTime

This method returns the `DateTime` value of the specified column.

Declaration

```
// C#  
public DateTime GetDateTime(int index);
```

Parameters

- `index`
The zero-based column index.

Return Value

The `DateTime` value of the column.

Implements

`IDataRecord`

Exceptions

`InvalidOperationException` - The connection is closed, the reader is closed, `Read()` has not been called, or all rows have been read.

`IndexOutOfRangeException` - The column index is invalid.

`InvalidCastException` - The accessor method is invalid for this column type or the column value is `NULL`.

Remarks

`IsDBNull` should be called to check for `NULL` values before calling this method.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbDataReader Class](#)
 - [RdbDataReader Members](#)
-

GetDecimal

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

Declaration

```
// C#  
public decimal GetDecimal(int index);
```

Parameters

- `index`

The zero-based column index.

Return Value

The `decimal` value of the column.

Implements

`IDataRecord`

Exceptions

`InvalidOperationException` - The connection is closed, the reader is closed, `Read()` has not been called, or all rows have been read.

`IndexOutOfRangeException` - The column index is invalid.

`InvalidCastException` - The accessor method is invalid for this column type or the column value is `NULL`.

Remarks

`IsDBNull` should be called to check for `NULL` values before calling this method.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbDataReader Class](#)
 - [RdbDataReader Members](#)
-

GetDouble

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

Declaration

```
// C#  
public double GetDouble(int index);
```

Parameters

- `index`
The zero-based column index.

Return Value

The `double` value of the column.

Implements

`IDataRecord`

Exceptions

`InvalidOperationException` - The connection is closed, the reader is closed, `Read()` has not been called, or all rows have been read.

`IndexOutOfRangeException` - The column index is invalid.

`InvalidCastException` - The accessor method is invalid for this column type or the column value is `NULL`.

Remarks

`IsDBNull` should be called to check for `NULL` values before calling this method.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
-

-
- [RdbDataReader Class](#)
 - [RdbDataReader Members](#)
-

GetFieldType

This method returns the `Type` of the specified column.

Declaration

```
// C#  
public Type GetFieldType(int index);
```

Parameters

- `index`
The zero-based column index.

Return Value

The `Type` of the default .NET type of the column.

Implements

`IDataRecord`

Exceptions

`InvalidOperationException` - The reader is closed.

`IndexOutOfRangeException` - The column index is invalid.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbDataReader Class](#)
 - [RdbDataReader Members](#)
-

GetFloat

This method returns the `float` value of the specified `NUMBER` column or `BINARY FLOAT` column.

Declaration

```
// C#  
public float GetFloat(int index);
```

Parameters

- `index`
The zero-based column index.

Return Value

The `float` value of the column.

Implements

`IDataRecord`

Exceptions

`InvalidOperationException` - The connection is closed, the reader is closed, `Read()` has not been called, or all rows have been read.

`IndexOutOfRangeException` - The column index is invalid.

`InvalidCastException` - The accessor method is invalid for this column type or the column value is NULL.

Remarks

`IsDBNull` should be called to check for NULL values before calling this method.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbDataReader Class](#)
 - [RdbDataReader Members](#)
-

GetInt16

This method returns the `Int16` value of the specified `NUMBER` column.

Declaration

```
// C#  
public short GetInt16(int index);
```

Parameters

- `index`
The zero-based column index.

Return Value

The `Int16` value of the column.

Implements

`IDataRecord`

Exceptions

`InvalidOperationException` - The connection is closed, the reader is closed, `Read()` has not been called, or all rows have been read.

`IndexOutOfRangeException` - The column index is invalid.

`InvalidCastException` - The accessor method is invalid for this column type or the column value is NULL.

Remarks

`IsDBNull` should be called to check for NULL values before calling this method.

Note:

`short` is equivalent to `Int16`.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbDataReader Class](#)
 - [RdbDataReader Members](#)
-

GetInt32

This method returns the `Int32` value of the specified `NUMBER` column.

Declaration

```
// C#
public int GetInt32(int index);
```

Parameters

- *index*
The zero-based column index.

Return Value

The `Int32` value of the column.

Implements

`IDataRecord`

Exceptions

`InvalidOperationException` - The connection is closed, the reader is closed, `Read()` has not been called, or all rows have been read.

`IndexOutOfRangeException` - The column index is invalid.

`InvalidCastException` - The accessor method is invalid for this column type or the column value is `NULL`.

Remarks

`IsDBNull` should be called to check for `NULL` values before calling this method.

Note:

`int` is equivalent to `Int32`.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbDataReader Class](#)
 - [RdbDataReader Members](#)
-

GetInt64

This method returns the `Int64` value of the specified `NUMBER` column.

Declaration

```
// C#
public long GetInt64(int index);
```

Parameters

- *index*
The zero-based column index.

Return Value

The `Int64` value of the column.

Implements

`IDataRecord`

Exceptions

`InvalidOperationException` - The connection is closed, the reader is closed, `Read()` has not been called, or all rows have been read.

`IndexOutOfRangeException` - The column index is invalid.

`InvalidCastException` - The accessor method is invalid for this column type or the column value is NULL.

Remarks

`IsDBNull` should be called to check for NULL values before calling this method.

Note:

`long` is equivalent to `Int64`.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbDataReader Class](#)
 - [RdbDataReader Members](#)
-

GetName

This method returns the name of the specified column.

Declaration

```
// C#  
public string GetName(int index);
```

Parameters

- `index`
The zero-based column index.

Return Value

The name of the column.

Implements

`IDataRecord`

Exceptions

`InvalidOperationException` - The reader is closed.

`IndexOutOfRangeException` - The column index is invalid.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbDataReader Class](#)
 - [RdbDataReader Members](#)
-

GetDateTime

This method returns a `System.Date` structure of the specified DATE column.

Declaration

```
// C#  
public System.Date GetDateTime(int index);
```

Parameters

- `index`
The zero-based column index.

Return Value

The `Date` value of the column.

Exceptions

`InvalidOperationException` - The connection is closed, the reader is closed, `Read()` has not been called, or all rows have been read.

`IndexOutOfRangeException` - The column index is invalid.

`InvalidCastException` - The accessor method is invalid for this column type or the column value is `NULL`.

Remarks

`IsDBNull` should be called to check for `NULL` values before calling this method.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbDataReader Class](#)
 - [RdbDataReader Members](#)
-

GetDecimal

This method returns a `Decimal` structure of the specified `NUMBER` column.

Declaration

```
// C#  
public Decimal GetDecimal(int index);
```

Parameters

- *index*
The zero-based column index.

Return Value

The `System.Decimal` value of the column.

Exceptions

`InvalidOperationException` - The connection is closed, the reader is closed, `Read()` has not been called, or all rows have been read.

`IndexOutOfRangeException` - The column index is invalid.

`InvalidCastException` - The accessor method is invalid for this column type or the column value is `NULL`.

Remarks

`IsDBNull` should be called to check for `NULL` values before calling this method.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbDataReader Class](#)
 - [RdbDataReader Members](#)
-

GetOrdinal

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

Declaration

```
// C#
public int GetOrdinal(string name);
```

Parameters

- *name*
The specified column name.

Return Value

The index of the column.

Implements

`IDataRecord`

Exceptions

`InvalidOperationException` - The reader is closed.

`IndexOutOfRangeException` - The column index is invalid.

Remarks

A case-sensitive search is made to locate the specified column by its name. If this fails, then a case-insensitive search is made.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbDataReader Class](#)
 - [RdbDataReader Members](#)
-

GetSchemaTable

This method returns a `DataTable` that describes the column metadata of the `RdbDataReader`.

Declaration

```
// C#
public DataTable GetSchemaTable();
```

Return Value

A `DataTable` that contains the metadata of the result set.

Implements

`IDataReader`

Exceptions

`InvalidOperationException` - The connection is closed or the reader is closed.

Remarks

`RdbDataReader.GetSchemaTable()` returns the `SchemaTable`.

RdbDataReader SchemaTable

The `RdbDataReader SchemaTable` is a `DataTable` that describes the column metadata of the `RdbDataReader`.

The columns of the `SchemaTable` are in the order shown.

Table 4-27 RdbDataReader SchemaTable

Name	Name Type	Description.
ColumnName	System.String	The name of the column.
ColumnOrdinal	System.Int32	The 0-based ordinal of the column.
ColumnSize	System.Int64	The maximum possible length of a value in the column (in octets).
NumericPrecision	System.Int16	The maximum precision of the column, if the column is a numeric datatype.
NumericScale	System.Int16	The scale of the column.
IsUnique	System.Boolean	Indicates whether the column is unique. <code>true</code> 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 <code>BaseTableName</code> . <code>IsUnique</code> is guaranteed to be <code>true</code> if one of the following applies: <ul style="list-style-type: none"> the column constitutes a key by itself there is a unique constraint or a unique index that applies only to this column and a <code>NOT NULL</code> constraint has been defined on the column the column is an explicitly selected <code>ROWID</code> <code>IsUnique</code> is false if the column can contain duplicate values in the base table. The default is <code>false</code> . The value of this property is the same for each occurrence of the base table column in the select list.
IsKey	System.Boolean	Indicates whether the column is a key column. <code>true</code> 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 <code>IsKey</code> set to <code>true</code> 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: <ul style="list-style-type: none"> A base table primary key. Any of the unique constraints or unique indexes with the following condition: A <code>NOT NULL</code> constraint must be defined on the column or on all of the columns, in the case of a composite unique constraint or composite unique index. Any of the composite unique constraints or composite unique indexes with the following condition: A <code>NULL</code> constraint must be defined on at least one, but not all, of the columns. An explicitly selected <code>ROWID</code>. <code>False</code> 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.

Name	Name Type	Description.
IsRowID	System.Boolean	true if the column is the DbKey (ROWID) for the row, otherwise false.
BaseColumnName	System.String	The name of the column in the database if an alias is used for the column.
BaseCatalogName	System.String	The name of the catalog in the database that contains 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	Type	The database column type (DbType) of the column.
ProviderType	Int32	The Oracle Rdb database column type of the column.
AllowDBNull	System.Boolean	true if null values are allowed, otherwise false.
IsExpression	System.Boolean	true if the column is an expression; otherwise false.
IsLong	System.Boolean	true if the column is a BLOB; otherwise false.
IsReadOnly	System.Boolean	true if the column is read-only; otherwise false.
IsAutoIncrement	System.Boolean	true if the column is auto-increment; otherwise false.
Cast	System.String	SQL cast statement used with this field type.

Example

This example creates and uses the `SchemaTable` from the reader.

```
// C#
public static void ReadSchemaTable(string connStr)
{
    .
    .
    .
    //get the reader
    RdbDataReader reader = cmd.ExecuteReader();
    //get the schema table
    DataTable schemaTable = reader.GetSchemaTable();
    //retrieve the first column info.
    DataRow col0 = schemaTable.Rows[0];
    //print out the column info
    Console.WriteLine("Column name: " + col0["COLUMNNAME"]);
    Console.WriteLine("Precision: " + col0["NUMERICPRECISION"]);
    Console.WriteLine("Scale: " + col0["NUMERICSCALE"]);
    .
    .
    .
}
```

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
- [RdbDataReader Class](#)
- [RdbDataReader Members](#)

GetString

This method returns the `string` value of the specified column.

Declaration

```
// C#
public string GetString(int index);
```

Parameters

- *index*
The zero-based column index.

Return Value

The `string` value of the column.

Implements

`IDataRecord`

Exceptions

`InvalidOperationException` - The connection is closed, the reader is closed, `Read()` has not been called, or all rows have been read.

`IndexOutOfRangeException` - The column index is invalid.

`InvalidCastException` - The accessor method is invalid for this column type or the column value is `NULL`.

Remarks

`IsDBNull` should be called to check for `NULL` values before calling this method.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbDataReader Class](#)
 - [RdbDataReader Members](#)
-

GetValue

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

Declaration

```
// C#
public object GetValue(int index);
```

Parameters

- *index*
The zero-based column index.

Return Value

The value of the column as a .NET type.

Implements

`IDataRecord`

Exceptions

`InvalidOperationException` - The connection is closed, the reader is closed, `Read()` has not been called, or all rows have been read.

`IndexOutOfRangeException` - The column index is invalid.

Remarks

When this method is invoked for a `NUMBER` column, the .NET type returned depends on the precision and scale of the column. For example, if a column is defined as `NUMBER(4,0)` then values in this column are retrieved as a `System.Int16`.

If the precision and scale is such that no .NET type can represent all the possible values that could exist in that column, the value is returned as a `System.Decimal`, if possible. If the value cannot be represented by a `System.Decimal`, an exception is raised. For example, if a column is defined as `NUMBER(20,10)` then a value in this column is retrieved as a `System.Decimal`.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbDataReader Class](#)
 - [RdbDataReader Members](#)
-

GetValues

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

Declaration

```
// C#  
public int GetValues(object[] values);
```

Parameters

- *values*
An array of objects to hold the .NET types as the column values.

Return Value

The number of objects in the *values* array.

Implements

`IDataRecord`

Exceptions

`InvalidOperationException` - The connection is closed, the reader is closed, `Read()` has not been called, or all rows have been read.

Remarks

This method provides a way to retrieve all column values rather than retrieving each column value individually.

The number of column values retrieved is the minimum of the length of the *values* array and the number of columns in the result set.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbDataReader Class](#)
 - [RdbDataReader Members](#)
-

IsDBNull

This method indicates whether the column value is NULL.

Declaration

```
// C#  
public bool IsDBNull(int index);
```

Parameters

- *index*
The zero-based column index.

Return Value

Returns `true` if the column is a `NULL` value; otherwise, returns `false`.

Implements

`IDataRecord`

Exceptions

`InvalidOperationException` - The reader is closed, `Read()` has not been called, or all rows have been read.

`IndexOutOfRangeException` - The column index is invalid.

Remarks

This method should be called to check for `NULL` values before calling the other accessor methods.

Example

The code example for the `RdbDataReader` class includes the `IsDBNull` method.

See "[Example](#)" in the `RdbDataReader` class section.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbDataReader Class](#)
 - [RdbDataReader Members](#)
-

NextResult

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

Declaration

```
// C#  
public bool NextResult();
```

Return Value

Returns `true` if another result set exists; otherwise, returns `false`.

Implements

`IDataReader`

Exceptions

`InvalidOperationException` - The connection is closed or the reader is closed.

Remarks

`NextResult` is used when reading results from stored procedure execution that return more than one result set.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbDataReader Class](#)
 - [RdbDataReader Members](#)
-

Read

This method reads the next row in the result set.

Declaration

```
// C#  
public bool Read();
```

Return Value

Returns `true` if another row exists; otherwise, returns `false`.

Implements

`IDataReader`

Exceptions

`InvalidOperationException` - The connection is closed or the reader is closed.

Remarks

The initial position of the data reader is before the first row. Therefore, the `Read` method must be called to fetch the first row. The row that was just read is considered the *current row*. If the `RdbDataReader` has no more rows to read, it returns `false`.

Example

The code example for the `RdbDataReader` class includes the `Read` method. See "[Example](#)" in the `RdbDataReader` class section.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbDataReader Class](#)
 - [RdbDataReader Members](#)
-

4.2.6 RdbError Class

The `RdbError` class represents an error reported by `Rdb`.

Class Inheritance

```
Object  
  RdbError
```

Declaration

```
// C#  
public sealed class RdbError
```

Thread Safety

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

Remarks

The `RdbError` class represents a warning or an error reported by `Rdb`.

Example

```
// C#  
. . .  
try  
{  
    cmd.ExecuteNonQuery()  
}  
catch ( RdbException e )
```

```

{
    RdbError err1 = e.Errors[0];
    RdbError err2 = e.Errors[1];
    Console.WriteLine("Error 1 Message:", err1.Message);
    Console.WriteLine("Error 2 Source:", err2.Source);
}

```

Requirements

Namespace: Oracle.DataAccess.RdbClient

Assembly: Rdb.DataAccess.Rdb.dll

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
- [RdbError Members](#)
- [RdbError Static Methods](#)
- [RdbError Properties](#)
- [RdbError Methods](#)

4.2.6.1 RdbError Members

`RdbError` members are listed in the following tables:

RdbError Static Methods

`RdbError` static methods are listed in [Table 4-28](#).

Table 4-28 RdbError Static Methods

Methods	Description
<code>Equals</code>	Inherited from <code>Object</code> (Overloaded).

RdbError Properties

`RdbError` properties are listed in [Table 4-29](#).

Table 4-29 RdbError Properties

Property	Description
Message	Specifies the message describing the error.
Number	Specifies the Rdb error number.
Procedure	Specifies the stored procedure that causes the error.
Source	Specifies the name of the data provider that generates the error.
SqlState	Specifies the SQL State value associated with this error.
StackTrace	Specifies the stack trace for this error.

RdbError Methods

`RdbError` methods are listed in [Table 4-30](#).

Table 4-30 RdbError Methods

Method	Description
<code>Equals</code>	Inherited from <code>Object</code> (Overloaded).
<code>GetHashCode</code>	Inherited from <code>Object</code> .

GetType	Inherited from Object .
ToString	Returns a string representation of the RdbError .

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbError Members](#)
 - [RdbError Static Methods](#)
 - [RdbError Properties](#)
 - [RdbError Methods](#)
-

4.2.6.2 RdbError Static Methods

RdbError static methods are listed in [Table 4-28](#).

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbError Members](#)
 - [RdbError Class](#)
-

4.2.6.3 RdbError Properties

RdbError properties are listed in [Table 4-29](#).

Message

This property specifies the message describing the error.

Declaration

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

Property Value

A string.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbError Members](#)
 - [RdbError Class](#)
-

Number

This property specifies the Rdb error number.

Declaration

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

Property Value

An int.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbError Members](#)
-

-
- [RdbError Class](#)
-

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 `RdbError` object.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbError Members](#)
 - [RdbError Class](#)
-

Source

This property specifies the name of the data provider that generates the error.

Declaration

```
// C#  
public string Source {get;}
```

Property Value

A string.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbError Members](#)
 - [RdbError Class](#)
-

SQLState

This property specifies the SQLSTATE (if any) associated with the error.

Declaration

```
// C#  
public string SQLState {get;}
```

Property Value

A string.

Remarks

See your Oracle Rdb SQL documentation for the possible values and descriptions of SQLSTATE.

See Also:

-
- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbError Members](#)
 - [RdbError Class](#)
-

StackTrace

This property specifies the stack trace for the underlying exception associated with this `error`.

Declaration

```
// C#  
public string StackTrace {get;}
```

Property Value

A string.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbError Members](#)
 - [RdbError Class](#)
-

4.2.6.4 RdbError Methods

`RdbError` methods are listed in [Table 4-30](#).

ToString

Overrides `Object`

This method returns a string representation of the `RdbError`.

Declaration

```
// C#  
public override string ToString();
```

Return Value

Returns a string with the format:

RDB-error number: error message
or
SQL- SQLState: error message

Example

```
RDB-99009:Failed to connect
```

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbError Members](#)
 - [RdbError Class](#)
-

4.2.7 RdbErrorCollection Class

An `RdbErrorCollection` class represents a collection of all errors that are thrown by the Oracle Rdb Data Provider for .NET.

Class Inheritance

```
Object
  CollectionBase
    RdbErrorCollection
```

Declaration

```
// C#
public sealed class RdbErrorCollection : CollectionBase
```

Thread Safety

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

Remarks

A simple `CollectionBase` that holds a list of `RdbErrors`.

Example

```
// C#
// The following example demonstrates how to access an
// individual RdbErrors from an RdbException

public void DisplayErrors(RdbException myException)
{
    for (int i=0; i < myException.Errors.Count; i++;)
    {
        Console.WriteLine("Index #" + i + "\n" +
            "Error: " + myException.Errors[i].ToString() + "\n");
    }
}
```

Requirements

Namespace: `Oracle.DataAccess.RdbClient`

Assembly: `Rdb.DataAccess.Rdb.dll`

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
- [RdbErrorCollection Members](#)
- [RdbErrorCollection Static Methods](#)
- [RdbErrorCollection Properties](#)
- [RdbErrorCollection Public Methods](#)

4.2.7.1 RdbErrorCollection Members

`RdbErrorCollection` members are listed in the following tables:

RdbErrorCollection Static Methods

`RdbErrorCollection` static methods are listed in [Table 4-31](#).

Table 4-31 RdbErrorCollection Static Methods

Methods	Description
<code>Equals</code>	Inherited from <code>Object</code> (Overloaded).

RdbErrorCollection Properties

`RdbErrorCollection` properties are listed in [Table 4–32](#).

Table 4-32 RdbErrorCollection Properties

Property	Description
Capacity	Inherited from <code>CollectionBase</code> .
Count	Inherited from <code>CollectionBase</code> .
IsReadOnly	Inherited from <code>CollectionBase</code> .
IsSynchronized	Inherited from <code>CollectionBase</code> .
Item	Inherited from <code>CollectionBase</code> .

RdbErrorCollection Public Methods

`RdbError` methods are listed in [Table 4–33](#).

Table 4-33 RdbErrorCollection Public Methods

Public Method	Description
<code>CopyTo</code>	Inherited from <code>CollectionBase</code> .
<code>Equals</code>	Inherited from <code>Object</code> (Overloaded).
<code>GetHashCode</code>	Inherited from <code>Object</code> .
<code>GetType</code>	Inherited from <code>Object</code> .
<code>ToString</code>	Inherited from <code>Object</code> .

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
- [RdbErrorCollection Members](#)
- [RdbErrorCollection Static Methods](#)
- [RdbErrorCollection Properties](#)
- [RdbErrorCollection Public Methods](#)

4.2.7.2 RdbErrorCollection Static Methods

`RdbErrorCollection` static methods are listed in [Table 4–31](#).

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
- [RdbErrorCollection Members](#)
- [RdbErrorCollection Class](#)

4.2.7.3 RdbErrorCollection Properties

`RdbErrorCollection` properties are listed in [Table 4–32](#).

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
- [RdbErrorCollection Members](#)
- [RdbErrorCollection Class](#)

4.2.7.4 RdbErrorCollection Public Methods

`RdbError` methods are listed in [Table 4–33](#).

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbErrorCollection Members](#)
 - [RdbErrorCollection Class](#)
-

4.2.8 RdbException Class

The `RdbException` class represents an exception that is thrown when the Oracle Rdb Data Provider for .NET encounters an error. Each `RdbException` object contains at least one `RdbError` object in the `Error` property that describes the error or warning.

Class Inheritance

```
Object
  Exception
    SystemException
      RdbException
```

Declaration

```
// C#
public sealed class RdbException : SystemException
```

Thread Safety

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

Example

```
// C#
// The following example generates an RdbException due to
// bad SQL syntax, (that is the missing keyword "from")
// and then displays the exception message and source property.
.
.
.
try
{
    .
    .
    .
    // select * emp will cause an error
    RdbCommand cmd = new RdbCommand("select * emp", con);
}
catch ( RdbException e )
{
    Console.WriteLine("{0} throws {1}",e.Source, e.Message);
}
.
.
.
```

Requirements

Namespace: `Oracle.DataAccess.RdbClient`

Assembly: `Rdb.DataAccess.Rdb.dll`

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbException Members](#)
 - [RdbException Static Methods](#)
 - [RdbException Properties](#)
 - [RdbException Methods](#)
-

4.2.8.1 RdbException Members

`RdbException` members are listed in the following tables:

RdbException Static Methods

`RdbException` static methods are listed in [Table 4-34](#).

Table 4-34 RdbException Static Methods

Methods	Description
<code>Equals</code>	Inherited from <code>Object</code> (Overloaded).

RdbException Properties

`RdbException` properties are listed in [Table 4-35](#).

Table 4-35 RdbException Properties

Property	Description
Errors	Specifies a collection of one or more <code>RdbError</code> objects that contain information about exceptions generated by the Rdb database.
<code>HelpLink</code>	Inherited from <code>Exception</code> .
<code>InnerException</code>	Inherited from <code>Exception</code> .
Message	Specifies the error messages that occur in the exception.
Number	Specifies the Rdb error number.
Procedure	Specifies the stored procedure that caused the exception.
Source	Specifies the name of the data provider that generates the error.
<code>StackTrace</code>	Inherited from <code>Exception</code> .
<code>TargetSite</code>	Inherited from <code>Exception</code> .

RdbException Methods

`RdbException` methods are listed in [Table 4-36](#).

Table 4-36 RdbException Methods

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

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
- [RdbException Members](#)
- [RdbException Static Methods](#)
- [RdbException Properties](#)
- [RdbException Methods](#)

4.2.8.2 RdbException Static Methods

`RdbException` static methods are listed in [Table 4–34](#).

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbException Members](#)
 - [RdbException Class](#)
-

4.2.8.3 RdbException Properties

`RdbException` properties are listed in [Table 4–35](#).

Errors

This property specifies a collection of one or more `RdbError` objects that contain information about exceptions generated by the Rdb database.

Declaration

```
// C#  
public RdbErrorCollection Errors {get;}
```

Property Value

An `RdbErrorCollection`.

Remarks

The `Errors` property contains at least one instance of `RdbError` objects.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbException Members](#)
 - [RdbException Class](#)
-

Message

Overrides `Exception`

This property specifies the error messages that occur in the exception.

Declaration

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

Property Value

A `string`.

Remarks

`Message` is a concatenation of all errors in the `Errors` collection. Each error message is concatenated and is followed by a carriage return, except the last one.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbException Members](#)
 - [RdbException Class](#)
-

Number

This property specifies the Rdb 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 Rdb and can be a provider-specific error number.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbException Members](#)
 - [RdbException Class](#)
-

Procedure

This property specifies the stored procedure that caused the exception.

Declaration

```
// C#
public string Procedure {get;}
```

Property Value

The stored procedure name.

Source

Overrides `Exception`

This property specifies the name of the data provider that generates the error.

Declaration

```
// C#
public override string Source {get;}
```

Property Value

The name of the data provider.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbException Members](#)
 - [RdbException Class](#)
-

4.2.8.4 RdbException Methods

`RdbException` methods are listed in [Table 4–36](#).

GetObjectData

Overrides `Exception`

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

Declaration

```
// C#
public override void GetObjectData(SerializationInfo info,
StreamingContext context);
```

Parameters

- *info*
A `SerializationInfo` object.
- *context*
A `StreamingContext` object.

Remarks

The information includes `DataSource`, `Message`, `Number`, `Procedure`, `Source`, and `StackTrace`.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbException Members](#)
 - [RdbException Class](#)
-

ToString

Overrides `Exception`

This method returns the fully qualified name of this exception, the `error` message in the `Message` property, the `InnerException.ToString()` message, and the stack trace.

Declaration

```
// C#  
public override string ToString();
```

Return Value

The string representation of the exception.

Example

```
// C#  
. . .  
try  
{  
    // incorrect spelling of "from" will cause an exception  
    RdbCommand cmd = new RdbCommand("select * form emp", con);  
}  
catch ( RdbException e )  
{  
    Console.WriteLine("{0}", e.ToString());  
}  
. . .
```

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbException Members](#)
 - [RdbException Class](#)
-

4.2.9 RdbInfoMessageEventArgs Class

The `RdbInfoMessageEventArgs` class provides event data for the `RdbConnection.InfoMessage` event. When any warning occurs in the database, the `RdbConnection.InfoMessage` event is triggered along with the `RdbInfoMessageEventArgs` object that stores the event data.

Class Inheritance

```
Object
  EventArgs
    RdbInfoMessageEventArgs
```

Declaration

```
// C#
public sealed class RdbInfoMessageEventArgs
```

Thread Safety

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

Example

```
// C#
public void WarningHandler(object src, RdbInfoMessageEventArgs args)
{
    LogOutput("Source object is: " + src.GetType().Name);
    LogOutput("InfoMessageArgs.Message is " + args.Message);
    LogOutput("InfoMessageArgs.Errors is " + args.Errors);
    LogOutput("InfoMessageArgs.Source is " + args.Source);
}

public bool MyFunc()
{
    .
    .
    .
    conn.Open();
    RdbCommand cmd = conn.CreateCommand();
    //Register to the InfoMessageHandler
    cmd.Connection.InfoMessage +=
    new RdbInfoMessageEventHandler(WarningHandler);
    cmd.CommandText = CmdStr;
    cmd.CommandType = CommandType.Text;
    //If CmdStr causes warning(s), it will be handled.
    cmd.ExecuteNonQuery();
    .
    .
    .
}
```

Requirements

Namespace: `Oracle.DataAccess.RdbClient`

Assembly: `Rdb.DataAccess.Rdb.dll`

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbInfoMessageEventArgs Members](#)
 - [RdbInfoMessageEventArgs Static Methods](#)
 - [RdbInfoMessageEventArgs Properties](#)
 - [RdbInfoMessageEventArgs Public Methods](#)
-

4.2.9.1 RdbInfoMessageEventArgs Members

RdbInfoMessageEventArgs members are listed in the following tables:

RdbInfoMessageEventArgs Static Methods

The RdbInfoMessageEventArgs static methods are listed in [Table 4-37](#).

Table 4-37 RdbInfoMessageEventArgs Static Methods

Methods	Description
Equals	Inherited from <code>Object</code> (Overloaded).

RdbInfoMessageEventArgs Properties

The RdbInfoMessageEventArgs properties are listed in [Table 4-38](#).

Table 4-38 RdbInfoMessageEventArgs Properties

Name	Description
Errors	Specifies the collection of errors generated by the data source.
Message	Specifies the error text generated by the data source.
Source	Specifies the name of the object that generated the error.

RdbInfoMessageEventArgs Public Methods

The RdbInfoMessageEventArgs methods are listed in [Table 4-39](#).

Table 4-39 RdbInfoMessageEventArgs Public Methods

Name	Description
Equals	Inherited from <code>Object</code> (Overloaded).
GetHashCode	Inherited from <code>Object</code> .
GetType	Inherited from <code>Object</code> .
ToString	Inherited from <code>Object</code> .

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
- [RdbInfoMessageEventArgs Members](#)
- [RdbInfoMessageEventArgs Static Methods](#)
- [RdbInfoMessageEventArgs Properties](#)
- [RdbInfoMessageEventArgs Public Methods](#)

4.2.9.2 RdbInfoMessageEventArgs Static Methods

The RdbInfoMessageEventArgs static methods are listed in [Table 4-37](#).

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
- [RdbInfoMessageEventArgs Members](#)
- [RdbInfoMessageEventArgs Class](#)

4.2.9.3 RdbInfoMessageEventArgs Properties

The `RdbInfoMessageEventArgs` properties are listed in [Table 4–38](#).

Errors

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

Declaration

```
// C#  
public RdbErrorCollection Errors {get;}
```

Property Value

The collection of errors.

Message

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

Declaration

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

Property Value

The error text.

Source

This property specifies the name of the object that generated the error.

Declaration

```
// C#  
public string Source {get;}
```

Property Value

The object that generated the error.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbInfoMessageEventArgs Members](#)
 - [RdbInfoMessageEventArgs Class](#)
-

4.2.9.4 RdbInfoMessageEventArgs Public Methods

The `RdbInfoMessageEventArgs` methods are listed in [Table 4–39](#).

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbInfoMessageEventArgs Members](#)
 - [RdbInfoMessageEventArgs Class](#)
-

4.2.10 RdbInfoMessageEventHandler Delegate

The `RdbInfoMessageEventHandler` represents the signature of the method that handles the `RdbConnection.InfoMessage` event.

Declaration

```
// C#  
public delegate void RdbInfoMessageEventHandler(object sender,
```

```
RdbInfoMessageEventArgs eventArgs);
```

Parameter

- *sender*
The source of the event.
- *eventArgs*
The `RdbInfoMessageEventArgs` object that contains the event data.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
- [RdbInfoMessageEventArgs Members](#)
- [RdbInfoMessageEventArgs Class](#)

4.2.11 RdbParameter Class

An `RdbParameter` object represents a parameter for an `RdbCommand` or a `DataSet` column.

Class Inheritance

```
Object
  MarshalByRefObject
    RdbParameter
```

Declaration

```
// C#
public sealed class RdbParameter : MarshalByRefObject, IDataParameter,
IDisposable, ICloneable
```

Thread Safety

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

Exceptions

`ArgumentException` - The type binding is invalid.

Example

```
// C#
.
.
.
RdbParameter [] prm = new RdbParameter[3];
// Create RdbParameter objects through RdbParameterCollection
prm[0] = cmd.Parameters.Add("paramEmpno", DbType.Decimal,
    1234, ParameterDirection.Input);
prm[1] = cmd.Parameters.Add("paramEname", DbType.String,
    "Client", ParameterDirection.Input);
prm[2] = cmd.Parameters.Add("paramDeptNo", DbType.Decimal,
    10, ParameterDirection.Input);
cmd.CommandText =
    "insert into emp(empno, ename, deptno) values (:1, :2, :3)";
cmd.CommandType = CommandType.CommandText;
cmd.ExecuteNonQuery();
.
.
.
```

Requirements

Namespace: `Oracle.DataAccess.RdbClient`

Assembly: `Rdb.DataAccess.Rdb.dll`

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbParameter Members](#)
 - [RdbParameter Constructors](#)
 - [RdbParameter Static Methods](#)
 - [RdbParameter Properties](#)
 - [RdbParameter Public Methods](#)
-

4.2.11.1 RdbParameter Members

`RdbParameter` members are listed in the following tables:

RdbParameter Constructors

`RdbParameter` constructors are listed in [Table 4-40](#).

Table 4-40 RdbParameter Constructors

Constructor	Description
RdbParameter Constructors	Instantiates a new instance of <code>RdbParameter</code> class (Overloaded).

RdbParameter Static Methods

`RdbParameter` static methods are listed in [Table 4-41](#).

Table 4-41 RdbParameter Static Methods

Methods	Description
<code>Equals</code>	Inherited from <code>Object</code> (Overloaded).

RdbParameter Properties

`RdbParameter` properties are listed in [Table 4-42](#).

Table 4-42 RdbParameter Properties

Name	Description
DbType	Specifies the datatype of the parameter using the <code>Data.DbType</code> enumeration type.
Direction	Specifies whether the parameter is input-only, output-only, bi-directional, or a stored function return value parameter.
<code>IsNull</code>	<i>This method is a no-op.</i>
ParameterName	Specifies the name of the parameter.
Precision	Specifies the maximum number of digits used to represent the <code>Value</code> property.
Scale	Specifies the number of decimal places to which <code>Value</code> property is resolved.
Size	Specifies the maximum size, in bytes or characters, of the data transmitted to or from the server.
SourceColumn	Specifies the name of the <code>DataTable</code> Column of the <code>DataSet</code> .
SourceVersion	Specifies the <code>DataRowVersion</code> value to use when loading the <code>Value</code> property of the parameter.
Value	Specifies the value of the <code>Parameter</code> .

RdbParameter Public Methods

`RdbParameter` public methods are listed in [Table 4-43](#).

Table 4-43 `RdbParameter` Public Methods

Public Method	Description
Clone	Creates a shallow copy of an <code>RdbParameter</code> object.
Dispose	Releases allocated resources.
<code>Equals</code>	Inherited from <code>Object</code> (Overloaded).
<code>GetHashCode</code>	Inherited from <code>Object</code> .
<code>GetType</code>	Inherited from <code>Object</code> .
<code>ToString</code>	Inherited from <code>Object</code> (Overloaded).

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
- [RdbParameter Members](#)
- [RdbParameter Constructors](#)
- [RdbParameter Static Methods](#)
- [RdbParameter Properties](#)
- [RdbParameter Public Methods](#)

4.2.11.2 `RdbParameter` Constructors

`RdbParameter` constructors instantiate new instances of the `RdbParameter` class.

Overload List:

- [RdbParameter\(\)](#)
This constructor instantiates a new instance of `RdbParameter` class.
- [RdbParameter\(string, DbType\)](#)
This constructor instantiates a new instance of `RdbParameter` class using the supplied parameter name and datatype.
- [RdbParameter\(string, object\)](#)
This constructor instantiates a new instance of the `RdbParameter` class using the supplied parameter name and parameter value.
- [RdbParameter\(string, DbType, ParameterDirection\)](#)
This constructor instantiates a new instance of the `RdbParameter` class using the supplied parameter name, datatype, and parameter direction.
- [RdbParameter\(string, DbType, object, ParameterDirection\)](#)
This constructor instantiates a new instance of the `RdbParameter` class using the supplied parameter name, datatype, value, and direction.
- [RdbParameter\(string, DbType, int\)](#)
This constructor instantiates a new instance of the `RdbParameter` class using the supplied parameter name, datatype, and size.
- [RdbParameter\(string, DbType, int, string\)](#)
This constructor instantiates a new instance of the `RdbParameter` class using the supplied parameter name, datatype, size, and source column.
- [RdbParameter\(string, DbType, int, ParameterDirection, bool, byte, byte, string, DataRowVersion, object\)](#)
This constructor instantiates a new instance of the `RdbParameter` class using the supplied parameter name, datatype, size, direction, null indicator, precision, scale, source column, source version and parameter value.
- [RdbParameter\(string, DbType, int, object, ParameterDirection\)](#)
This constructor instantiates a new instance of the `RdbParameter` class using the supplied parameter name, datatype, size, value, and direction.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbParameter Members](#)
 - [RdbParameter Class](#)
-

RdbParameter()

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

Declaration

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

Remarks**Default Values:**

`DbType` - String
`ParameterDirection` - Input
`isNullable` - true
`offset` - 0
`ParameterName` - Empty string
`Precision` - 0
`Size` - 0
`SourceColumn` - Empty string
`Value` - null

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbParameter Members](#)
 - [RdbParameter Class](#)
-

RdbParameter (string, DbType)

This constructor instantiates a new instance of `RdbParameter` class using the supplied parameter name and Rdb datatype.

Declaration

```
// C#  
public RdbParameter(string parameterName, DbType dbType);
```

Parameters

- *parameterName*
Specifies the parameter name.
- *dbType*
Specifies the datatype of the `RdbParameter`

Remarks

Unless explicitly set in the constructor, all the properties have the default values.

Default Values:

`DbType` - String
`ParameterDirection` - Input
`isNullable` - true
`offset` - 0
`ParameterName` - Empty string
`Precision` - 0

Size - 0
SourceColumn - Empty string
SourceVersion - Current
Value - null

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbParameter Members](#)
 - [RdbParameter Class](#)
-

RdbParameter(string, object)

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

Declaration

```
// C#  
public RdbParameter(string parameterName, object obj);
```

Parameters

- *parameterName*
Specifies parameter name.
- *obj*
Specifies value of the `RdbParameter`.

Remarks

Unless explicitly set in the constructor, all the properties have the default values.

Default Values:

DbType - String
ParameterDirection - Input
isNullable - true
offset - 0
ParameterName - Empty string
Precision - 0
Size - 0
SourceColumn - Empty string
SourceVersion - Current
Value - null

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbParameter Members](#)
 - [RdbParameter Class](#)
-

RdbParameter(string, DbType, ParameterDirection)

This constructor instantiates a new instance of the `RdbParameter` class using the supplied parameter name, datatype, and parameter direction.

Declaration

```
// C#  
public RdbParameter(string parameterName, DbType type,  
ParameterDirection direction);
```

Parameters

- *parameterName*
Specifies the parameter name.
- *type*
Specifies the datatype of the `RdbParameter`.
- *direction*
Specifies the direction of the `RdbParameter`.

Remarks

Unless explicitly set in the constructor, all the properties have the default values.

Default Values:

`DbType` - String
`ParameterDirection` - Input
`isNullable` - true
`offset` - 0
`ParameterName` - Empty string
`Precision` - 0
`Size` - 0
`SourceColumn` - Empty string
`SourceVersion` - Current
`Value` - null

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbParameter Members](#)
 - [RdbParameter Class](#)
-

RdbParameter(string, DbType, object, ParameterDirection)

This constructor instantiates a new instance of the `RdbParameter` class using the supplied parameter name, datatype, value, and direction.

Declaration

```
// C#  
public RdbParameter(string parameterName, DbType type, object obj,  
ParameterDirection direction);
```

Parameters

- *parameterName*
Specifies the parameter name.
- *type*
Specifies the datatype of the `RdbParameter`.
- *obj*
Specifies the value of the `RdbParameter`.
- *direction*
Specifies one of the `ParameterDirection` values.

Remarks

Unless explicitly set in the constructor, all the properties have the default values.

Default Values:

`DbType` - String

ParameterDirection - Input
isNullable - true
offset - 0
ParameterName - Empty string
Precision - 0
Size - 0
SourceColumn - Empty string
SourceVersion - Current
Value - null

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbParameter Members](#)
 - [RdbParameter Class](#)
-

RdbParameter(string, DbType, int)

This constructor instantiates a new instance of the `RdbParameter` class using the supplied parameter name, datatype, and size.

Declaration

```
// C#  
public RdbParameter(string parameterName, DbType type, int size);
```

Parameters

- *parameterName*
Specifies the parameter name.
- *type*
Specifies the datatype of the `RdbParameter`.
- *size*
Specifies the size of the `RdbParameter` 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
ParameterName - Empty string
Precision - 0
Size - 0
SourceColumn - Empty string
SourceVersion - Current
Value - null

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbParameter Members](#)
 - [RdbParameter Class](#)
-

RdbParameter(string, DbType, int, string)

This constructor instantiates a new instance of the `RdbParameter` class using the supplied parameter name, datatype, size, and source column.

Declaration

```
// C#
public RdbParameter(string parameterName, DbType type, int size,
string srcColumn);
```

Parameters

- *parameterName*
Specifies the parameter name.
- *type*
Specifies the datatype of the `RdbParameter`.
- *size*
Specifies the size of the `RdbParameter` value.
- *srcColumn*
Specifies the name of the source column.

Remarks

Unless explicitly set in the constructor, all the properties have the default values.

Default Values:

`DbType` - String
`ParameterDirection` - Input
`isNullable` - true
`offset` - 0
`ParameterName` - Empty string
`Precision` - 0
`Size` - 0
`SourceColumn` - Empty string
`SourceVersion` - Current
`Value` - null

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbParameter Members](#)
 - [RdbParameter Class](#)
-

RdbParameter(string, DbType, int, ParameterDirection, bool, byte, byte, string, DataRowVersion, object)

This constructor instantiates a new instance of the `RdbParameter` class using the supplied parameter name, datatype, size, direction, null indicator, precision, scale, source column, source version and parameter value.

Declaration

```
// C#
public RdbParameter(string parameterName, DbType dbType, int size,
ParameterDirection direction, bool isNullable, byte precision,
byte scale, string srcColumn, DataRowVersion srcVersion, object obj);
```

Parameters

- *parameterName*
Specifies the parameter name.

- *dbType*
Specifies the datatype of the `RdbParameter`.
- *size*
Specifies the size of the `RdbParameter` value.
- *direction*
Specifies `ParameterDirection` value.
- *isNullable*
Specifies if the parameter value can be null.
- *precision*
Specifies the precision of the parameter value.
- *scale*
Specifies the scale of the parameter value.
- *srcColumn*
Specifies the name of the source column.
- *srcVersion*
Specifies one of the `DataRowVersion` values.
- *obj*
Specifies the parameter value.

Exceptions

`ArgumentException` - The supplied value does not belong to the type of `Value` property in any of the `Types`.

Remarks

Unless explicitly set in the constructor, all the properties have the default values.

Default Values:

`DbType` - String
`ParameterDirection` - Input
`isNullable` - true
`offset` - 0
`ParameterName` - Empty string
`Precision` - 0
`Size` - 0
`SourceColumn` - Empty string
`SourceVersion` - Current
`Value` - null

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbParameter Members](#)
 - [RdbParameter Class](#)
-

RdbParameter(string, DbType, int, object, ParameterDirection)

This constructor instantiates a new instance of the `RdbParameter` class using the supplied parameter name, datatype, size, value, and direction.

Declaration

```
// C#
public RdbParameter(string parameterName, DbType type, int size, object
obj, ParameterDirection direction);
```

Parameters

- *parameterName*
Specifies the parameter name.
- *type*
Specifies the datatype of the `RdbParameter`.
- *size*
Specifies the size of the `RdbParameter` value.
- *obj*
Specifies the value of the `RdbParameter`.
- *direction*
Specifies one of the `ParameterDirection` values.

Remarks

Changing the `DbType` implicitly changes the `DbType`.
Unless explicitly set in the constructor, all the properties have the default values.

Default Values:

`DbType` - String
`ParameterDirection` - Input
`isNullable` - true
`offset` - 0
`ParameterName` - Empty string
`Precision` - 0
`Size` - 0
`SourceColumn` - Empty string
`SourceVersion` - Current
`ArrayBindStatus` - Success
`Value` - null

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbParameter Members](#)
 - [RdbParameter Class](#)
-

4.2.11.3 RdbParameter Static Methods

`RdbParameter` static methods are listed in [Table 4-41](#).

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbParameter Members](#)
 - [RdbParameter Class](#)
-

4.2.11.4 RdbParameter Properties

`RdbParameter` properties are listed in [Table 4-42](#).

DbType

This property specifies the datatype of the parameter using the `Data.DbType` enumeration type.

Declaration

```
// C#
```

```
public DbType DbType { get; set; }
```

Property Value

A `DbType` enumerated value.

Implements

`IDataParameter`

Exceptions

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

Remarks

Default = `DbType.String`

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
- [RdbParameter Members](#)
- [RdbParameter Class](#)

Direction

This property specifies whether the parameter is input-only, output-only, bi-directional, or a stored function return value parameter.

Declaration

```
// C#  
public ParameterDirection Direction { get; set; }
```

Property Value

A `ParameterDirection` enumerated value.

Implements

`IDataParameter`

Exceptions

`ArgumentOutOfRangeException` - The `ParameterDirection` value specified is invalid.

Remarks

Default = `ParameterDirection.Input`

Possible values: `Input`, `InputOutput`, `Output`, and `ReturnValue`.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
- [RdbParameter Members](#)
- [RdbParameter Class](#)

Null

This property indicates that the `Value` property is `DBNull`, the database `NULL` value.

Declaration

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

Property Value

A `bool` that specifies that the value is `DBNull`.

Remarks

Default = `false`.

This property may be used to set the `NULL` indicator for this parameter.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbParameter Members](#)
 - [RdbParameter Class](#)
-

ParameterName

This property specifies the name of the parameter.

Declaration

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

Property Value

String

Implements

`IDataParameter`

Remarks

Default = `null`

`Rdb` supports `ParameterName` up to 30 characters.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbParameter Members](#)
 - [RdbParameter Class](#)
-

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 `DbType.Decimal`.

`Rdb` supports `Precision` range from 0 to 38.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbParameter Members](#)
-

-
- [RdbParameter Class](#)
-

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 `DbType.Decimal`.

`Rdb` supports `Scale` between -84 and 127.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbParameter Members](#)
 - [RdbParameter Class](#)
-

Size

This property specifies the maximum size, in bytes or characters, of the data transmitted to or from the server.

Declaration

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

Property Value

int

Exceptions

`ArgumentOutOfRangeException` - The `Size` value specified is invalid.

Remarks

The default value is 0.

Before execution, this property specifies the maximum size to be bound in the `Value` property.

After execution, it contains the size of the type in the `Value` property.

Currently `Size` is only used for parameters of type `String`:

The value of `Size` is handled as follows:

- Fixed length datatypes: ignored
- Variable length datatypes: describes the maximum amount of data transmitted to or from the server. For character data, `Size` is in number of characters and for binary data, it is in number of bytes. If the `Size` is not explicitly set, it is inferred from the actual size of the specified parameter value when binding.

Note:

`Size` does not include the null terminating character for the string data.

See Also:

-
- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbParameter Members](#)
 - [RdbParameter Class](#)
-

SourceColumn

This property specifies the name of the DataTable Column of the DataSet.

Declaration

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

Property Value

A string.

Implements

IDataParameter

Remarks

Default = empty string

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbParameter Members](#)
 - [RdbParameter Class](#)
-

SourceVersion

This property specifies the DataRowVersion value to use when loading the Value property of the parameter.

Declaration

```
// C#  
public DataRowVersion SourceVersion { get; set; }
```

Property Value

DataRowVersion

Implements

IDataParameter

Exceptions

ArgumentOutOfRangeException - The DataRowVersion value specified is invalid.

Remarks

Default = DataRowVersion.Current

SourceVersion is used by the RdbDataAdapter.UpdateCommand() during the RdbDataAdapter.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 RdbDataAdapter.InsertCommand() and the RdbDataAdapter.DeleteCommand().

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbParameter Members](#)
-

-
- [RdbParameter Class](#)
-

Value

This property specifies the value of the `Parameter`.

Declaration

```
// C#  
public object Value { get; set; }
```

Property Value

An object.

Implements

`IDataParameter`

Exceptions

`ArgumentException` - The `Value` property specified is invalid.

`InvalidArgumentException`- The `Value` property specified is invalid.

Remarks

Default = `null`

The `Value` property can be overwritten by `RdbDataAdapter.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 `System.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 `Null` property to `true`. In this case, the provider sends a `null` value to the database.

If `DbType` is not set, its value can be inferred by `Value`.

For input parameters the value is:

- Bound to the `RdbCommand` that is sent to the server.
- Converted to the datatype specified in `DbType` when the provider sends the data to the server.

For output parameters the value is:

- Set on completion of the `RdbCommand` (true for return value parameters also).
- Set to the data from the server, to the datatype specified in `DbType`.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbParameter Members](#)
 - [RdbParameter Class](#)
-

4.2.11.5 RdbParameter Public Methods

`RdbParameter` public methods are listed in [Table 4-43](#).

Clone

This method creates a shallow copy of an `RdbParameter` object.

Declaration

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

Return Value

An `RdbParameter` object.

Implements

`ICloneable`

Remarks

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

Example

```
// C#  
.  
.  
.  
//Need a proper casting for the return value when cloned  
RdbParameter paramcloned = (RdbParameter) param.Clone();  
.  
.  
.
```

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbParameter Members](#)
 - [RdbParameter Class](#)
-

Dispose

This method releases resources allocated for an `RdbParameter` object.

Declaration

```
// C#  
public void Dispose();
```

Implements

`IDisposable`

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbParameter Members](#)
 - [RdbParameter Class](#)
-

4.2.12 RdbParameterCollection Class

An `RdbParameterCollection` class represents a collection of all parameters relevant to an `RdbCommand` object and their mappings to `DataSet` columns.

Class Inheritance

```
Object  
  MarshalByRefObject  
    RdbParameterCollection
```

Declaration

```
// C#
public sealed class RdbParameterCollection : IDataParameterCollection,
    IList, ICollection, IEnumerable
```

Thread Safety

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

Remarks

The position of an `RdbParameter` added into the `RdbParameterCollection` 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 `RdbParameter` in the `RdbParameterCollection` is ignored.

Example

```
// C#
.
.
.
string conStr =
    @"User Id=rdb_user;Password=rdb_pw;
    Server=MYNODE:MY_SRV;Database=MY_DBS:MF_PERSONNEL;";
// Create the RdbConnection
RdbConnection conn = new RdbConnection(conStr);
conn.Open();
// Create the RdbCommand
RdbCommand cmd = new RdbCommand();
cmd.Connection = conn;
// Create RdbParameter
RdbParameter [] prm = new RdbParameter[3];
// Bind parameters
prm[0] = cmd.Parameters.Add("paramEmpno", DbType.Decimal, 1234,
    ParameterDirection.Input);
prm[1] = cmd.Parameters.Add("paramEname", DbType.String,
    "Client", ParameterDirection.Input);
prm[2] = cmd.Parameters.Add("paramDeptNo", DbType.Decimal, 10,
    ParameterDirection.Input);
cmd.CommandText =
    "insert into emp(empno, ename, deptno) values(:1, :2, :3)";
cmd.ExecuteNonQuery();
// Remove RdbParameter objects from the collection
cmd.Parameters.Clear();
// Dispose RdbCommand object
cmd.Dispose();
// Close and Dispose RdbConnection object
conn.Close();
conn.Dispose();
.
.
.
```

Requirements

Namespace: `Oracle.DataAccess.RdbClient`

Assembly: `Rdb.DataAccess.Rdb.dll`

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbParameterCollection Members](#)
 - [RdbParameterCollection Static Methods](#)
-

- [RdbParameterCollection Properties](#)
- [RdbParameterCollection Public Methods](#)

4.2.12.1 RdbParameterCollection Members

`RdbParameterCollection` members are listed in the following tables:

RdbParameterCollection Static Methods

`RdbParameterCollection` static methods are listed in [Table 4-44](#).

Table 4-44 RdbParameterCollection Static Methods

Methods	Description
<code>Equals</code>	Inherited from <code>Object</code> (Overloaded).

RdbParameterCollection Properties

`RdbParameterCollection` properties are listed in [Table 4-45](#).

Table 4-45 RdbParameterCollection Properties

Name	Description
Count	Specifies the number of <code>RdbParameters</code> in the collection.
Item	Gets and sets the <code>RdbParameter</code> object (Overloaded).

RdbParameterCollection Public Methods

`RdbParameterCollection` public methods are listed in [Table 4-46](#).

Table 4-46 RdbParameterCollection Public Methods

Public Method	Description
Add	Adds objects to the collection (Overloaded).
Clear	Removes all the <code>RdbParameter</code> objects from the collection.
Contains	Indicates whether objects exist in the collection (Overloaded).
CopyTo	Copies <code>RdbParameter</code> objects from the collection, starting with the supplied <code>index</code> to the supplied array.
<code>Equals</code>	Inherited from <code>Object</code> (Overloaded).
<code>GetHashCode</code>	Inherited from <code>Object</code> .
<code>GetType</code>	Inherited from <code>Object</code> .
IndexOf	Returns the <code>index</code> of the objects in the collection (Overloaded).
Insert	Inserts the supplied <code>RdbParameter</code> to the collection at the specified <code>index</code> .
Remove	Removes objects from the collection.
RemoveAt	Removes objects from the collection by location (Overloaded).
<code>ToString</code>	Inherited from <code>Object</code> .

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
- [RdbParameterCollection Members](#)
- [RdbParameterCollection Static Methods](#)
- [RdbParameterCollection Properties](#)
- [RdbParameterCollection Public Methods](#)

4.2.12.2 RdbParameterCollection Static Methods

`RdbParameterCollection` static methods are listed in [Table 4–44](#).

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbParameterCollection Members](#)
 - [RdbParameterCollection Class](#)
-

4.2.12.3 RdbParameterCollection Properties

`RdbParameterCollection` properties are listed in [Table 4–45](#).

Count

This property specifies the number of `RdbParameter` objects in the collection.

Declaration

```
// C#  
public int Count {get;}
```

Property Value

The number of `RdbParameter` objects.

Implements

`ICollection`

Remarks

Default = 0

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbParameterCollection Members](#)
 - [RdbParameterCollection Class](#)
-

Item

`Item` gets and sets the `RdbParameter` object.

Overload List:

- [Item\[int\]](#)
This property gets and sets the `RdbParameter` object at the index specified by the supplied `parameterIndex`.
- [Item\[string\]](#)
This property gets and sets the `RdbParameter` object using the parameter name specified by the supplied `parameterName`.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbParameterCollection Members](#)
 - [RdbParameterCollection Class](#)
-

Item[int]

This property gets and sets the `RdbParameter` 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 `RdbParameterCollection` class is a zero-based index.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
- [RdbParameterCollection Members](#)
- [RdbParameterCollection Class](#)

Item[string]

This property gets and sets the `RdbParameter` object using the parameter name specified by the supplied `parameterName`.

Declaration

```
// C#  
public RdbParameter Item[string parameterName] {get; set;};
```

Property Value

An `RdbParameter`.

Implements

`IDataParameterCollection`

Exceptions

`IndexOutOfRangeException` - The supplied parameter name does not exist.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
- [RdbParameterCollection Members](#)
- [RdbParameterCollection Class](#)

4.2.12.4 RdbParameterCollection Public Methods

`RdbParameterCollection` public methods are listed in [Table 4-46](#).

Add

`Add` adds objects to the collection.

Overload List:

- [Add\(object\)](#)
This method adds the supplied object to the collection.
- [Add\(RdbParameter\)](#)
This method adds the supplied `RdbParameter` object to the collection.
- [Add\(string, object\)](#)

This method adds an `RdbParameter` object to the collection using the supplied name and object value.

- [Add\(string, DbType\)](#)
This method adds an `RdbParameter` object to the collection using the supplied name and database type.
- [Add\(string, DbType, ParameterDirection\)](#)
This method adds an `RdbParameter` object to the collection using the supplied name, database type, and direction.
- [Add\(string, DbType, object, ParameterDirection\)](#)
This method adds an `RdbParameter` object to the collection using the supplied name, database type, parameter value, and direction.
- [Add\(string, DbType, int, object, ParameterDirection\)](#)
This method adds an `RdbParameter` object to the collection using the supplied name, database type, size, parameter value, and direction.
- [Add\(string, DbType, int\)](#)
This method adds an `RdbParameter` object to the collection using the supplied name, database type, and size.
- [Add\(string, DbType, int, string\)](#)
This method adds an `RdbParameter` object to the collection using the supplied name, database type, size, and source column.
- [Add\(string, DbType, int, ParameterDirection, bool, byte, byte, string, DataRowVersion, object\)](#)
This method adds an `RdbParameter` object to the collection using the supplied name, database type, size, direction, nullability indicator, precision, scale, source column, source version, and parameter value.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbParameterCollection Members](#)
 - [RdbParameterCollection Class](#)
-

Add(object)

This method adds the supplied object to the collection.

Declaration

```
// C#  
public int Add(object obj);
```

Parameters

- *obj*
Specifies the supplied object value.

Return Value

The index at which the new [RdbParameter](#) is added.

Implements

`IList`

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbParameterCollection Members](#)
 - [RdbParameterCollection Class](#)
-

Add(RdbParameter)

This method adds the supplied [RdbParameter](#) object to the collection.

Declaration

```
// C#  
public RdbParameter Add(RdbParameter paramObj);
```

Parameters

- *paramObj*
Specifies the supplied `RdbParameter` object.

Return Value

The newly created `RdbParameter` object which was added to the collection.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
- [RdbParameterCollection Members](#)
- [RdbParameterCollection Class](#)
- [RdbParameter](#) for the default values of any unspecified `RdbParameter` properties

Add(string, object)

This method adds [RdbParameter](#) object to the collection using the supplied name and object value

Declaration

```
// C#  
public RdbParameter Add(string name, object val);
```

Parameters

- *name*
Specifies the parameter name.
- *val*
Specifies the `RdbParameter` value.

Return Value

The newly created `RdbParameter` object, which was added to the collection.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
- [RdbParameterCollection Members](#)
- [RdbParameterCollection Class](#)
- [RdbParameter](#) for the default values of any unspecified `RdbParameter` properties

Add(string, DbType)

This method adds an [RdbParameter](#) object to the collection using the supplied name and database type.

Declaration

```
// C#  
public RdbParameter Add(string name, DbType dbType);
```

Parameters

- *name*
Specifies the parameter name.

- *dbType*
Specifies the datatype of the [RdbParameter](#).

Return Value

The newly created `RdbParameter` object, which was added to the collection.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
- [RdbParameterCollection Members](#)
- [RdbParameterCollection Class](#)
- [RdbParameter](#) for the default values of any unspecified `RdbParameter` properties

Add(string, DbType, ParameterDirection)

This method adds an `RdbParameter` object to the collection using the supplied name, database type, and direction.

Declaration

```
// C#
public RdbParameter Add(string name, DbType dbType, ParameterDirection
direction);
```

Parameters

- *name*
Specifies the parameter name.
- *dbType*
Specifies the datatype of the [RdbParameter](#).
- *direction*
Specifies the `RdbParameter` direction.

Return Value

The newly created `RdbParameter` object, which was added to the collection.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
- [RdbParameterCollection Members](#)
- [RdbParameterCollection Class](#)
- [RdbParameter](#) for the default values of any unspecified `RdbParameter` properties.

Add(string, DbType, object, ParameterDirection)

This method adds an [RdbParameter](#) object to the collection using the supplied name, database type, parameter value, and direction.

Declaration

```
// C#
public RdbParameter Add(string name, DbType dbType, object val,
ParameterDirection dir);
```

Parameters

- *name*
Specifies the parameter name.
- *dbType*

- Specifies the datatype of the `RdbParameter`.
- *val*
Specifies the `RdbParameter` value.
- *dir*
Specifies one of the `ParameterDirection` values.

Return Value

The newly created `RdbParameter` object, which was added to the collection.

Example

```
// C#
.
.
.
RdbParameter prm = new RdbParameter();
prm = cmd.Parameters.Add("paramEmpno", DbType.Decimal, 1234,
ParameterDirection.Input);
cmd.CommandText = "insert into NumTable(numcol) values(?)";
cmd.ExecuteNonQuery();
.
.
.
```

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
- [RdbParameterCollection Members](#)
- [RdbParameterCollection Class](#)
- [RdbParameter](#) for the default values of any unspecified `RdbParameter` properties.

Add(string, DbType, int, object, ParameterDirection)

This method adds an [RdbParameter](#) object to the collection using the supplied name, database type, size, parameter value, and direction.

Declaration

```
// C#
public RdbParameter Add(string name, DbType dbType, int size,
object val, ParameterDirection dir;
```

Parameters

- *name*
Specifies the parameter name.
- *dbType*
Specifies the datatype of the `RdbParameter`.
- *size*
Specifies the size of `RdbParameter`.
- *val*
Specifies the `RdbParameter` value.
- *dir*
Specifies one of the `ParameterDirection` values.

Return Value

The newly created `RdbParameter` object, which was added to the collection.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbParameterCollection Members](#)
 - [RdbParameterCollection Class](#)
 - [RdbParameter](#) for the default values of any unspecified `RdbParameter` properties.
-

Add(string, DbType, int)

This method adds an [RdbParameter](#) object to the collection using the supplied name, database type, and size.

Declaration

```
// C#  
public RdbParameter Add(string name, DbType dbType, int size);
```

Parameters

- *name*
Specifies the parameter name.
- *dbType*
Specifies the datatype of the `RdbParameter`.
- *size*
Specifies the size of `RdbParameter`.

Return Value

The newly created `RdbParameter` object, which was added to the collection.

Example

```
// C#  
. . .  
RdbParameter prm = new RdbParameter();  
prm = cmd.Parameters.Add("param1", DbType.Decimal, 10);  
prm.Direction = ParameterDirection.Input;  
prm.Value = 1111;  
cmd.CommandText = "insert into NumTable(numcol) values(?)";  
cmd.ExecuteNonQuery();  
. . .
```

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbParameterCollection Members](#)
 - [RdbParameterCollection Class](#)
 - [RdbParameter](#) for the default values of any unspecified `RdbParameter` properties.
-

Add (string, DbType, int, string)

This method adds an [RdbParameter](#) object to the collection using the supplied name, database type, size, and source column.

Declaration

```
// C#
```

```
public RdbParameter Add(string name, DbType dbType, int size, string srcColumn);
```

Parameters

- *name*
Specifies the parameter name.
- *dbType*
Specifies the datatype of the RdbParameter.
- *size*
Specifies the size of RdbParameter.
- *srcColumn*
Specifies the name of the source column.

Return Value

The newly created RdbParameter object, which was added to the collection.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
- [RdbParameterCollection Members](#)
- [RdbParameterCollection Class](#)
- [RdbParameter](#) for the default values of any unspecified RdbParameter properties.

Add(string, DbType, int, ParameterDirection, bool, byte, byte, string, DataRowVersion, object)

This method adds an [RdbParameter](#) 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 RdbParameter Add(string name, DbType dbType, int size, ParameterDirection dir, bool isNullable, byte precision, byte scale, string srcColumn, DataRowVersion version, object val);
```

Parameters

- *name*
Specifies the parameter name.
- *dbType*
Specifies the datatype of the RdbParameter.
- *size*
Specifies the size of RdbParameter.
- *dir*
Specifies one of the ParameterDirection values.
- *isNullable*
Specifies if the parameter value can be null. This value is silently discarded as all columns are deemed nullable.
- *precision*
Specifies the precision of the parameter value.
- *scale*
Specifies the scale of the parameter value.
- *srcColumn*
Specifies the name of the source column.

- *version*
Specifies one of the `DataRowVersion` values.
- *val*
Specifies the parameter value.

Return Value

The newly created `RdbParameter` 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 `DbType`s.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbParameterCollection Members](#)
 - [RdbParameterCollection Class](#)
-

Clear

This method removes all the `RdbParameter` objects from the collection.

Declaration

```
// C#  
public void Clear();
```

Implements

`IList`

Example

```
// C#  
.  
.  
.  
RdbCommand cmd = new RdbCommand(conn);  
RdbParameter [] prm = new RdbParameter[3];  
prm[0] = cmd.Parameters.Add("paramEmpno", DbType.Decimal,  
    1234, ParameterDirection.Input);  
prm[1] = cmd.Parameters.Add("paramEname", DbType.String,  
    "Client", ParameterDirection.Input);  
prm[2] = cmd.Parameters.Add("paramDeptNo", DbType.Decimal,  
    10, ParameterDirection.Input);  
cmd.CommandText =  
    "insert into emp(empno, ename, deptno) values (:1, :2, :3)";  
cmd.ExecuteNonQuery();  
// This method removes all the parameters  
// from the parameter collection.  
cmd.Parameters.Clear();  
.  
.  
.
```

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbParameterCollection Members](#)
 - [RdbParameterCollection Class](#)
-

Contains

`Contains` indicates whether the supplied object exists in the collection.

Overload List:

- [Contains\(object\)](#)
This method indicates whether the supplied object exists as a `Value` in an [RdbParameter](#) in the collection.
- [Contains\(RdbParameter\)](#)
This method indicates whether the supplied [RdbParameter](#) exists in the collection.
- [Contains\(string\)](#)
This method indicates whether an [RdbParameter](#) object exists in the collection with the name matching the supplied string.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbParameterCollection Members](#)
 - [RdbParameterCollection Class](#)
-

Contains(object)

This method indicates whether an [RdbParameter](#) with the specified `Value` exists in the collection.

Declaration

```
// C#  
public bool Contains(object obj)
```

Parameters

- `obj`
Specifies the value to look for.

Return Value

A `bool` that indicates whether or not the [RdbParameter](#) with the specified `Value` is contained in the collection.

Implements

`IList`

Remarks

Returns `true` if the collection contains the [RdbParameter](#) object; otherwise, returns `false`.

Example

```
//C#  
.  
.  
.  
if (cmd.Parameters.Contains(P1))  
.  
.  
.
```

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbParameterCollection Members](#)
-

-
- [RdbParameterCollection Class](#)
-

Contains(RdbParameter)

This method indicates whether the supplied [RdbParameter](#) exists in the collection.

Declaration

```
// C#  
public bool Contains(RdbParameter param)
```

Parameters

- *param*
Specifies the RdbParameter.

Return Value

A `bool` that indicates whether or not the [RdbParameter](#) specified is inside the collection.

Implements

`IList`

Exceptions

`InvalidCastException` - The supplied *param* is not an `RdbParameter` object.

Remarks

Returns `true` if the collection contains the `RdbParameter` object; otherwise, returns `false`.

Example

```
//C#  
  
// This method removes a particular parameter  
// from the parameter collection.  
RdbParameter prm;  
.  
.  
.  
if (cmd.Parameters.Contains(prm) )  
    cmd.Parameters.Remove(prm);  
.  
.  
.
```

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbParameterCollection Members](#)
 - [RdbParameterCollection Class](#)
-

Contains(string)

This method indicates whether an [RdbParameter](#) object exists in the collection using the supplied string as the [ParameterName](#) of the [RdbParameter](#) object.

Declaration

```
// C#  
public bool Contains(string name);
```

Parameters

- *name*

Specifies the name of `RdbParameter` object.

Return Value

Returns `true` if the collection contains the `RdbParameter` object with the specified parameter name; otherwise, returns `false`.

Implements

`IDataParameterCollection`

Example

```
// C#
.
.
.
// This method removes a particular parameter
// from the parameter collection.

if (cmd.Parameters.Contains("param1"))
    cmd.Parameters.Remove(prm);
.
.
.
```

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbParameterCollection Members](#)
 - [RdbParameterCollection Class](#)
-

CopyTo

This method copies `RdbParameter` objects from the collection, starting with the supplied `index` to the supplied array.

Declaration

```
// C#
public void CopyTo(Array array, int index);
```

Parameters

- `array`
Specifies the array.
- `index`
Specifies the index to array.

Implements

`Icollection`

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbParameterCollection Members](#)
 - [RdbParameterCollection Class](#)
-

IndexOf

`IndexOf` returns the index of the `RdbParameter` object in the collection.

Overload List:

- [IndexOf\(object\)](#)

This method returns the index of the `RdbParameter` object in the collection.

- [IndexOf\(String\)](#)

This method returns the `index` of the `RdbParameter` object with the specified name in the collection.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbParameterCollection Members](#)
 - [RdbParameterCollection Class](#)
-

IndexOf(object)

This method returns the index of the `RdbParameter` object in the collection.

Declaration

```
// C#  
public int IndexOf(object obj);
```

Parameters

- *obj*
Specifies the object.

Return Value

Returns the index of the `RdbParameter` object in the collection.

Implements

`IList`

Exceptions

`InvalidCastException` - The supplied *obj* cannot be cast to an `RdbParameter` object.

Remarks

Returns the `index` of the supplied `RdbParameter` *obj* in the collection.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbParameterCollection Members](#)
 - [RdbParameterCollection Class](#)
-

IndexOf(String)

This method returns the `index` of the `RdbParameter` object with the specified name in the collection.

Declaration

```
// C#  
public int IndexOf(String name);
```

Parameters

name
Specifies the name of parameter.

Return Value

Returns the `index` of the supplied `RdbParameter` in the collection.

Implements

IdataParameterCollection

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbParameterCollection Members](#)
 - [RdbParameterCollection Class](#)
-

Insert

This method inserts the supplied `RdbParameter` object to the collection at the specified index.

Declaration

```
// C#  
public void Insert(int index, object obj);
```

Parameters

- `index`
Specifies the index.
- `obj`
Specifies the `RdbParameter` object.

Implements

`IList`

Remarks

An `InvalidCastException` is thrown if the supplied `obj` cannot be cast to an `RdbParameter` object.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbParameterCollection Members](#)
 - [RdbParameterCollection Class](#)
-

Remove

This method removes the supplied `RdbParameter` from the collection.

Declaration

```
// C#  
public void Remove(object obj);
```

Parameters

- `obj`
Specifies the object to remove.

Implements

`IList`

Exceptions

`InvalidCastException` - The supplied `obj` cannot be cast to an `RdbParameter` object.

Example

```
// C#
```

-
-
-

```
prm = cmd.Parameters.Add("param1", DbType.Decimal, 1234,
ParameterDirection.Input);
if (cmd.Parameters.Contains((Object)prm))
{
    // This method removes a particular parameter from
    // the parameter collection.
    cmd.Parameters.Remove((Object) prm);
}
.
.
.
```

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbParameterCollection Members](#)
 - [RdbParameterCollection Class](#)
-

RemoveAt

`RemoveAt` removes the `RdbParameter` object from the collection by location or by name.

Overload List:

- [RemoveAt\(int\)](#)
This method removes from the collection the `RdbParameter` object located at the index specified by the supplied index.
- [RemoveAt\(String\)](#)
This method removes from the collection the `RdbParameter` object specified by the supplied name.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
- [RdbParameterCollection Class](#)
- [RdbParameterCollection Members](#)

RemoveAt(int)

This method removes from the collection the `RdbParameter` object located at the index specified by the supplied index.

Declaration

```
// C#
public void RemoveAt(int index);
```

Parameters

- `index`
Specifies the index from which the `RdbParameter` is to be removed.

Implements

`IList`

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbParameterCollection Members](#)
 - [RdbParameterCollection Class](#)
-

RemoveAt(String)

This method removes from the collection the `RdbParameter` object specified by the supplied name.

Declaration

```
// C#  
public void RemoveAt(String name);
```

Parameters

- *name*
The name of the `RdbParameter` object to be removed from the collection.

Implements

`IDataParameterCollection`

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbParameterCollection Members](#)
 - [RdbParameterCollection Class](#)
-

4.2.13 RdbRowUpdatedEventHandler Delegate

The `RdbRowUpdatedEventHandler` delegate represents the signature of the method that handles the `RdbDataAdapter.RowUpdated` event.

Declaration

```
// C#  
public delegate void RdbRowUpdatedEventHandler(object sender,  
RdbRowUpdatedEventArgs eventArgs);
```

Parameters

- *sender*
The source of the event.
- *eventArgs*
The `RdbRowUpdatedEventArgs` 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.RdbClient Namespace](#)
-

4.2.14 RdbRowUpdatedEventArgs Class

The `RdbRowUpdatedEventArgs` class provides event data for the `RdbDataAdapter.RowUpdated` event.

Class Inheritance

```
Object  
  EventArgs  
    RowUpdatedEventArgs  
      RdbRowUpdatedEventArgs
```

Declaration

```
// C#
```

```
public sealed class RdbRowUpdatedEventArgs : RowUpdatedEventArgs
```

Thread Safety

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

Example

The example for the `RowUpdated` event shows how to use `RdbRowUpdatedEventArgs`. See `RowUpdated` event "[Example](#)".

Requirements

Namespace: `Oracle.DataAccess.RdbClient`

Assembly: `Rdb.DataAccess.Rdb.dll`

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
- [RdbRowUpdatedEventArgs Members](#)
- [RdbRowUpdatedEventArgs Constructor](#)
- [RdbRowUpdatedEventArgs Static Methods](#)
- [RdbRowUpdatedEventArgs Properties](#)
- [RdbRowUpdatedEventArgs Public Methods](#)

4.2.14.1 RdbRowUpdatedEventArgs Members

`RdbRowUpdatedEventArgs` members are listed in the following tables:

RdbRowUpdatedEventArgs Constructors

`RdbRowUpdatedEventArgs` constructors are listed in [Table 4-47](#).

Table 4-47 RdbRowUpdatedEventArgs Constructors

Constructor	Description
RdbRowUpdatedEventArgs Constructor	Instantiates a new instance of <code>RdbRowUpdatedEventArgs</code> class.

RdbRowUpdatedEventArgs Static Methods

The `RdbRowUpdatedEventArgs` static methods are listed in [Table 4-48](#).

Table 4-48 RdbRowUpdatedEventArgs Static Methods

Methods	Description
<code>Equals</code>	Inherited from <code>Object</code> (Overloaded).

RdbRowUpdatedEventArgs Properties

The `RdbRowUpdatedEventArgs` properties are listed in [Table 4-49](#).

Table 4-49 RdbRowUpdatedEventArgs Properties

Name	Description
Command	Specifies the <code>RdbCommand</code> that is used when <code>RdbDataAdapter.Update()</code> is called.
<code>Errors</code>	Inherited from <code>RowUpdatedEventArgs</code> .
<code>RecordsAffected</code>	Inherited from <code>RowUpdatedEventArgs</code> .
<code>Row</code>	Inherited from <code>RowUpdatedEventArgs</code> .
<code>StatementType</code>	Inherited from <code>RowUpdatedEventArgs</code> .
<code>Status</code>	Inherited from <code>RowUpdatedEventArgs</code> .

Name	Description
TableMapping	Inherited from RowUpdatedEventArgs.

RdbRowUpdatedEventArgs Public Methods

The RdbRowUpdatedEventArgs properties are listed in [Table 4-50](#).

Table 4-50 RdbRowUpdatedEventArgs Public Methods

Public Method	Description
Equals	Inherited from Object (Overloaded).
GetHashCode	Inherited from Object.
GetType	Inherited from Object.
ToString	Inherited from Object.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
- [RdbRowUpdatedEventArgs Members](#)
- [RdbRowUpdatedEventArgs Constructor](#)
- [RdbRowUpdatedEventArgs Static Methods](#)
- [RdbRowUpdatedEventArgs Properties](#)
- [RdbRowUpdatedEventArgs Public Methods](#)

4.2.14.2 RdbRowUpdatedEventArgs Constructor

The RdbRowUpdatedEventArgs constructor creates a new RdbRowUpdatedEventArgs instance.

Declaration

```
// C#
public RdbRowUpdatedEventArgs(DataRow row, IDbCommand command,
StatementType statementType, DataTableMapping tableMapping);
```

Parameters

- *row*
The DataRow sent for Update.
- *command*
The IDbCommand executed during the Update.
- *statementType*
The StatementType enumeration value indicating the type of SQL statement executed.
- *tableMapping*
The DataTableMapping used for the Update.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
- [RdbRowUpdatedEventArgs Members](#)
- [RdbRowUpdatedEventArgs Class](#)

4.2.14.3 RdbRowUpdatedEventArgs Static Methods

The RdbRowUpdatedEventArgs static methods are listed in [Table 4-48](#).

See Also:

-
- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbRowUpdatedEventArgs Members](#)
 - [RdbRowUpdatedEventArgs Class](#)
-

4.2.14.4 RdbRowUpdatedEventArgs Properties

The `RdbRowUpdatedEventArgs` properties are listed in [Table 4-49](#).

Command

This property specifies the `RdbCommand` that is used when `RdbDataAdapter.Update()` is called.

Declaration

```
// C#  
public new RdbCommand Command {get;}
```

Property Value

The `RdbCommand` executed when `Update` is called.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbRowUpdatedEventArgs Members](#)
 - [RdbRowUpdatedEventArgs Class](#)
-

4.2.14.5 RdbRowUpdatedEventArgs Public Methods

The `RdbRowUpdatedEventArgs` properties are listed in [Table 4-50](#).

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbRowUpdatedEventArgs Members](#)
 - [RdbRowUpdatedEventArgs Class](#)
-

4.2.15 RdbRowUpdatingEventArgs Class

The `RdbRowUpdatingEventArgs` class provides event data for the `RdbDataAdapter.RowUpdating` event.

Class Inheritance

```
Object  
  EventArgs  
    RowUpdatingEventArgs  
      RdbRowUpdatingEventArgs
```

Declaration

```
// C#  
public sealed class RdbRowUpdatingEventArgs : RowUpdatingEventArgs
```

Thread Safety

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

Example

The example for the `RowUpdated` event shows how to use `RdbRowUpdatingEventArgs`. See `RowUpdated` event "[Example](#)".

Requirements

Namespace: Oracle.DataAccess.RdbClient

Assembly: Rdb.DataAccess.Rdb.dll

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
- [RdbRowUpdatingEventArgs Members](#)
- [RdbRowUpdatingEventArgs Constructor](#)
- [RdbRowUpdatingEventArgs Static Methods](#)
- [RdbRowUpdatingEventArgs Properties](#)
- [RdbRowUpdatingEventArgs Public Methods](#)

4.2.15.1 RdbRowUpdatingEventArgs Members

RdbRowUpdatingEventArgs members are listed in the following tables:

RdbRowUpdatingEventArgs Constructors

RdbRowUpdatingEventArgs constructors are listed in [Table 4-51](#).

Table 4-51 RdbRowUpdatingEventArgs Constructors

Constructor	Description
RdbRowUpdatingEventArgs Constructor	Instantiates a new instance of RdbRowUpdatingEventArgs class (Overloaded).

RdbRowUpdatingEventArgs Static Methods

The RdbRowUpdatingEventArgs static methods are listed in [Table 4-52](#).

Table 4-52 RdbRowUpdatingEventArgs Static Methods

Methods	Description
Equals	Inherited from Object (Overloaded).

RdbRowUpdatingEventArgs Properties

The RdbRowUpdatingEventArgs properties are listed in [Table 4-53](#).

Table 4-53 RdbRowUpdatingEventArgs Properties

Name	Description
Command	Specifies the RdbCommand that is used when RdbDataAdapter.Update() is called.
Errors	Inherited from RowUpdatingEventArgs.
Row	Inherited from RowUpdatingEventArgs.
StatementType	Inherited from RowUpdatingEventArgs.
Status	Inherited from RowUpdatingEventArgs.
TableMapping	Inherited from RowUpdatingEventArgs.

RdbRowUpdatingEventArgs Public Methods

The RdbRowUpdatingEventArgs public methods are listed in [Table 4-54](#).

Table 4-54 RdbRowUpdatingEventArgs Public Methods

Public Method	Description
Equals	Inherited from Object (Overloaded).

Public Method	Description
GetHashCode	Inherited from <code>Object</code> .
GetType	Inherited from <code>Object</code> .
ToString	Inherited from <code>Object</code> .

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
- [RdbRowUpdatingEventArgs Members](#)
- [RdbRowUpdatingEventArgs Constructor](#)
- [RdbRowUpdatingEventArgs Static Methods](#)
- [RdbRowUpdatingEventArgs Properties](#)
- [RdbRowUpdatingEventArgs Public Methods](#)

4.2.15.2 RdbRowUpdatingEventArgs Constructor

The `RdbRowUpdatingEventArgs` constructor creates a new instance of the `RdbRowUpdatingEventArgs` class using the supplied data row, `IDbCommand`, type of SQL statement, and table mapping.

Declaration

```
// C#
public RdbRowUpdatingEventArgs(DataRow row, IDbCommand command,
StatementType statementType, DataTableMapping tableMapping);
```

Parameters

- *row*
The `DataRow` sent for Update.
- *command*
The `IDbCommand` executed during the Update.
- *statementType*
The `StatementType` enumeration value indicating the type of SQL statement executed.
- *tableMapping*
The `DataTableMapping` used for the Update.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
- [RdbRowUpdatingEventArgs Members](#)
- [RdbRowUpdatingEventArgs Class](#)

4.2.15.3 RdbRowUpdatingEventArgs Static Methods

The `RdbRowUpdatingEventArgs` static methods are listed in [Table 4-52](#).

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
- [RdbRowUpdatingEventArgs Members](#)
- [RdbRowUpdatingEventArgs Class](#)

4.2.15.4 RdbRowUpdatingEventArgs Properties

The `RdbRowUpdatingEventArgs` properties are listed in [Table 4-53](#).

Command

This property specifies the `RdbCommand` that is used when the `RdbDataAdapter.Update()` is called.

Declaration

```
// C#  
public new RdbCommand Command {get; set;}
```

Property Value

The `RdbCommand` executed when `Update` is called.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbRowUpdatingEventArgs Members](#)
 - [RdbRowUpdatingEventArgs Class](#)
-

4.2.15.5 RdbRowUpdatingEventArgs Public Methods

The `RdbRowUpdatingEventArgs` public methods are listed in [Table 4-54](#).

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbRowUpdatingEventArgs Members](#)
 - [RdbRowUpdatingEventArgs Class](#)
-

4.2.16 RdbRowUpdatingEventHandler Delegate

The `RdbRowUpdatingEventHandler` delegate represents the signature of the method that handles the `RdbDataAdapter.RowUpdating` event.

Declaration

```
// C#  
public delegate void RdbRowUpdatingEventHandler (object sender,  
RdbRowUpdatingEventArgs eventArgs);
```

Parameters

- *sender*
The source of the event.
- *eventArgs*
The `RdbRowUpdatingEventArgs` 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.RdbClient Namespace](#)
 - [RdbRowUpdatingEventArgs Members](#)
 - [RdbRowUpdatingEventArgs Class](#)
-

4.2.17 RdbTransaction Class

An `RdbTransaction` object represents a local transaction.

Class Inheritance

```
Object
  MarshalByRefObject
    RdbTransaction
```

Declaration

```
// C#
public sealed class RdbTransaction : MarshalByRefObject,
IDbTransaction, IDisposable
```

Thread Safety

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

Remarks

The application calls `BeginTransaction` on the `RdbConnection` object to create an `RdbTransaction` object. The `RdbTransaction` object can be created in one of the following two modes:

- Read Committed (default)
- Serializable

Any other mode results in an exception.

Operations like commit and rollback performed on the transaction have no effect on data in any existing `DataSet`.

Note:

Try/Catch exception handling should always be used when rolling back a transaction. A **Rollback** generates an **InvalidOperationException** if the connection is terminated or if the transaction has already been rolled back on the server.

Example

```
// C#
// Starts a transaction and inserts one record.
// If insert fails, rolls back
// the transaction. Otherwise, commits the transaction.
.
.
.
RdbConnection conn = new RdbConnection(ConStr);
conn.Open();
//Create an RdbCommand object using the connection object
RdbCommand cmd = new RdbCommand("", conn);
// Start a transaction
RdbTransaction txn = conn.BeginTransaction(IsolationLevel.ReadCommitted);
try
{
    cmd.CommandText = "insert into mytable values (99, 'foo')";
    cmd.CommandType = CommandType.Text;
    cmd.ExecuteNonQuery();
    txn.Commit();
    Console.WriteLine("One record is inserted into the database table.");
}
catch (Exception e)
```

```

{
    try
    {
        txn.Rollback();
    }
    catch(Exception e2)
    {
        Console.WriteLine("Problem with rollback: " + e2.ToString());
    }

    Console.WriteLine("No record was inserted into the database table.");
}
.
.
.

```

Requirements

Namespace: Oracle.DataAccess.RdbClient

Assembly: Rdb.DataAccess.Rdb.dll

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
- [RdbTransaction Members](#)
- [RdbTransaction Static Methods](#)
- [RdbTransaction Properties](#)
- [RdbTransaction Public Methods](#)

4.2.17.1 RdbTransaction Members

RdbTransaction members are listed in the following tables:

RdbTransaction Static Methods

RdbTransaction static methods are listed in [Table 4-55](#).

Table 4-55 RdbTransaction Static Methods

Methods	Description
Equals	Inherited from Object (Overloaded).

RdbTransaction Properties

RdbTransaction properties are listed in [Table 4-56](#).

Table 4-56 RdbTransaction Properties

Name	Description
IsolationLevel	Specifies the isolation level for the transaction.
Connection	Specifies the connection that is associated with the transaction.

RdbTransaction Public Methods

RdbTransaction public methods are listed in [Table 4-57](#).

Table 4-57 RdbTransaction Public Methods

Public Method	Description
Commit	Commits the database transaction.
Dispose	Frees the resources used by the <code>RdbTransaction</code> object.
<code>Equals</code>	Inherited from <code>Object</code> (Overloaded).
<code>GetHashCode</code>	Inherited from <code>Object</code> .
<code>GetType</code>	Inherited from <code>Object</code> .
Rollback	Rolls back a database transaction.
<code>ToString</code>	Inherited from <code>Object</code> .

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
- [RdbTransaction Members](#)
- [RdbTransaction Static Methods](#)
- [RdbTransaction Properties](#)
- [RdbTransaction Public Methods](#)

4.2.17.2 RdbTransaction Static Methods

`RdbTransaction` static methods are listed in [Table 4-55](#).

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
- [RdbTransaction Members](#)
- [RdbTransaction Class](#)

4.2.17.3 RdbTransaction Properties

`RdbTransaction` properties are listed in [Table 4-56](#).

IsolationLevel

This property specifies the isolation level for the transaction.

Declaration

```
// C#  
public IsolationLevel IsolationLevel {get;}
```

Property Value

`IsolationLevel`

Implements

`IDbTransaction`

Exceptions

`InvalidOperationException` - The transaction has already completed.

Remarks

Default = `IsolationLevel.ReadCommitted`

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
- [RdbTransaction Members](#)

-
- [RdbTransaction Class](#)
-

Connection

This property specifies the connection that is associated with the transaction.

Declaration

```
// C#  
public RdbConnection Connection {get;}
```

Property Value

Connection

Implements

IDbTransaction

Remarks

This property indicates the `RdbConnection` object that is associated with the transaction.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbTransaction Members](#)
 - [RdbTransaction Class](#)
-

4.2.17.4 RdbTransaction Public Methods

`RdbTransaction` public methods are listed in [Table 4-57](#).

Commit

This method commits the database transaction.

Declaration

```
// C#  
public void Commit();
```

Implements

IDbTransaction

Exceptions

`InvalidOperationException` - The transaction has already been completed successfully, has been rolled back, or the associated connection is closed.

Remarks

Upon a successful commit, the transaction enters a completed state.

Example

```
// C#  
// Starts a transaction and inserts one record. If insert fails, rolls  
// back the transaction. Otherwise, commits the transaction.  
.  
.  
.  
RdbConnection conn = new RdbConnection(ConStr);  
conn.Open();  
// Create an RdbCommand object using the connection object  
RdbCommand cmd = new RdbCommand("", conn);  
// Start a transaction
```

```

RdbTransaction txn = conn.BeginTransaction(IsolationLevel.ReadCommitted);
try
{
    cmd.CommandText = "insert into mytable values (99, 'foo')";
    cmd.CommandType = CommandType.Text;
    cmd.ExecuteNonQuery();
    txn.Commit();
    Console.WriteLine(
        "One record was inserted into the database table.");
}
catch(Exception e)
{
    try
    {
        txn.Rollback();
    }
    catch(Exception e2)
    {
        Console.WriteLine("Problem with rollback: " + e2.ToString());
    }

    Console.WriteLine(
        "No record was inserted into the database table.");
}
.
.
.

```

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbTransaction Members](#)
 - [RdbTransaction Class](#)
-

Dispose

This method frees the resources used by the `RdbTransaction` object.

Declaration

```
// C#
public void Dispose();
```

Implements

`IDisposable`

Remarks

This method releases both the managed and unmanaged resources held by the `RdbTransaction` object. If the transaction is not in a completed state, an attempt to rollback the transaction is made.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbTransaction Members](#)
 - [RdbTransaction Class](#)
-

Rollback

`Rollback` rolls back a database transaction.

Declaration

```
// C#  
public void Rollback();
```

Implements

IDbTransaction

Exceptions

`InvalidOperationException` - The transaction has already been completed successfully, has been rolled back, or the associated connection is closed.

Remarks

After a `Rollback()`, the `RdbTransaction` object can no longer be used because the `Rollback` ends the transaction.

Note:

Try/Catch exception handling should always be used when rolling back a transaction. A **Rollback** generates an **InvalidOperationException** if the connection is terminated or if the transaction has already been rolled back on the server.

Example

```
// C#  
// Starts a transaction and inserts one record. Then rolls back the  
// transaction.  
. . .  
RdbConnection conn = new RdbConnection(ConStr);  
conn.Open();  
RdbCommand cmd = conn.CreateCommand();  
// Start a transaction  
RdbTransaction txn = conn.BeginTransaction(IsolationLevel.ReadCommitted);  
cmd.CommandText = "insert into mytable values (99, 'foo')";  
cmd.CommandType = CommandType.Text;  
cmd.ExecuteNonQuery();  
try  
{  
    txn.Rollback();  
}  
catch(Exception e2)  
{  
    Console.WriteLine("Problem with rollback: " + e2.ToString());  
}  
  
Console.WriteLine("Nothing was inserted into the database table.");  
. . .
```

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbTransaction Members](#)
 - [RdbTransaction Class](#)
-

4.2.18 RdbConnectionStringBuilder Class

The `RdbConnectionStringBuilder` class allows ORDP.NET specific connections strings to be created easily.

Class Inheritance

Object

DbConnectionStringBuilder
RdbConnectionStringBuilder

Declaration

```
// C#  
public sealed class RdbConnectionStringBuilder :  
DbConnectionStringBuilder
```

Thread Safety

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

Remarks

The `RdbConnectionStringBuilder` class allows easy creation of syntactically correct connections strings that may be used with `RdbConnection` objects.

Example

```
// C#  
. . .  
string conStr =  
    @"Server=node1.oracle.com:GENSRVC;Database=mydb;  
    User Id=myname;Password=mypassword;";  
  
RdbConnectionStringBuilder sb = new RdbConnectionStringBuilder();  
sb.ConnectionString = conStr;  
// try to see what the server will be  
sb.TryGetValue("server", out res);  
Console.WriteLine(" server = " + res);  
// now change the database we should connect to  
sb.DataSource = "disk1:[my_dbs]personnel";  
Console.WriteLine(" con str = " + sb.ConnectionString);  
. . .
```

Requirements

Namespace: `Oracle.DataAccess.RdbClient`

Assembly: `Rdb.DataAccess.Rdb.dll`

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbConnectionStringBuilder Members](#)
 - [RdbConnectionStringBuilder Constructors](#)
 - [RdbConnectionStringBuilder Properties](#)
 - [RdbConnectionStringBuilder Methods](#)
-

4.2.18.1 RdbConnectionStringBuilder Members

`RdbConnectionStringBuilder` members are listed in the following tables:

RdbConnectionStringBuilder Constructors

RdbConnectionStringBuilder constructors are listed in [Table 4-58](#).

Table 4-58 RdbConnectionStringBuilder Constructors

Constructor	Description
RdbConnectionStringBuilder Constructor	Instantiates a new instance of RdbConnectionStringBuilder class (Overloaded) .

RdbConnectionStringBuilder Properties

RdbConnectionStringBuilder properties are listed in [Table 4-59](#).

Table 4-59 RdbConnectionStringBuilder Properties

Property	Description
ConnectionString	Inherited from DbConnectionStringBuilder .
ConnectionTimeout	Specifies the timeout (in seconds) for connection .
DataSource	Specifies the datasource or database file specification .
Enlist	Specifies if the connection should enlist in the current transaction.
Password	Specifies the password for the database connection.
Pooling	Specifies if connection pooling should take place.
ReadOnly	Specifies if the connection is set read-only.
Server	Specifies the server to attach to.
Style	Specifies the style of the connection.
TraceLevel	Specifies the trace level .
TraceFilename	Specifies the trace filename.
UserId	Specifies the username to use for the connection.
Version	Specifies the Rdb version to use.

RdbConnectionStringBuilder Methods

RdbConnectionStringBuilder methods are listed in [Table 4-60](#).

Table 4-60 RdbConnectionStringBuilder Methods

Method	Description
Equals	Inherited from Object (Overloaded).
GetHashCode	Inherited from Object .
GetType	Inherited from Object .
TryGetValue	Returns a value for the specified attribute .

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
- [RdbConnectionStringBuilder Members](#)
- [RdbConnectionStringBuilder Constructors](#)
- [RdbConnectionStringBuilder Properties](#)
- [RdbConnectionStringBuilder Methods](#)

4.2.18.2 RdbConnectionStringBuilder Constructors

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

Overload List:

- [RdbConnectionStringBuilder\(\)](#)
This constructor instantiates a new instance of `RdbConnectionStringBuilder` class.
- [RdbConnectionStringBuilder\(string\)](#)
This constructor instantiates a new instance of `RdbConnectionStringBuilder` class using the supplied connection string .

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbConnectionStringBuilder Members](#)
 - [RdbConnectionStringBuilder Class](#)
-

RdbConnectionStringBuilder()

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

Declaration

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

Remarks

Default constructor.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbConnectionStringBuilder Members](#)
 - [RdbConnectionStringBuilder Class](#)
-

RdbConnectionStringBuilder (string)

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

Declaration

```
// C#  
public RdbConnectionStringBuilder(string connectionString);
```

Parameters

- *connectionString*
A valid `RdbConnection` connection string, as defined for the [ConnectionString](#) property of the `RdbConnection` object.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbConnectionStringBuilder Members](#)
 - [RdbConnectionStringBuilder Class](#)
-

4.2.18.3 RdbConnectionStringBuilder Properties

`RdbConnectionStringBuilder` properties are listed in [Table 4-59](#).

ConnectionTimeout

This property specifies the timeout to place on the connection request.

Declaration

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

Property Value

An integer.

Remarks

This gets or sets the `ConnectionTimeout` attribute of the connection string. The supported connection string attributes are listed in [Table 4-17](#).

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbConnectionStringBuilder Members](#)
 - [RdbConnectionStringBuilder Class](#)
-

DataSource

This property specifies the datasource or database file specification.

Declaration

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

Property Value

A string.

Remarks

This gets or sets the `Data Source` attribute of the connection string. The supported connection string attributes are listed in [Table 4-17](#).

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbConnectionStringBuilder Members](#)
 - [RdbConnectionStringBuilder Class](#)
-

Enlist

This property specifies if the connection should automatically enlist in the current system transaction.

Declaration

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

Remarks

This gets or sets the `Enlist` attribute of the connection string. The supported connection string attributes are listed in [Table 4-17](#).

Property Value

A boolean.

See Also:

-
- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbConnectionStringBuilder Members](#)
 - [RdbConnectionStringBuilder Class](#)
-

Password

This property specifies the password to use on the connection request.

Declaration

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

Property Value

A string.

Remarks

This gets or sets the `Password` attribute of the connection string. The supported connection string attributes are listed in [Table 4-17](#).

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbConnectionStringBuilder Members](#)
 - [RdbConnectionStringBuilder Class](#)
-

Pooling

This property specifies if connection pooling is enabled.

Note:

This property is currently ignored by ORDP.NET.

Declaration

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

Property Value

An boolean.

Remarks

This gets or sets the `Pooling` attribute of the connection string. The supported connection string attributes are listed in [Table 4-17](#).

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbConnectionStringBuilder Members](#)
 - [RdbConnectionStringBuilder Class](#)
-

ReadOnly

This property specifies the connection READ-ONLY state.

Declaration

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

Property Value

A boolean.

Remarks

This gets or sets the `ReadOnly` attribute of the connection string. The supported connection string attributes are listed in [Table 4-17](#).

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbConnectionStringBuilder Members](#)
 - [RdbConnectionStringBuilder Class](#)
-

Server

This property specifies the server to use for the connection request.

Declaration

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

Property Value

A string.

Remarks

This gets or sets the `Server` attribute of the connection string. The supported connection string attributes are listed in [Table 4-17](#).

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbConnectionStringBuilder Members](#)
 - [RdbConnectionStringBuilder Class](#)
-

Style

This property specifies the style of connection to use.

Declaration

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

Property Value

A string.

Remarks

This gets or sets the `Style` attribute of the connection string. The supported connection string attributes are listed in [Table 4-17](#).

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbConnectionStringBuilder Members](#)
 - [RdbConnectionStringBuilder Class](#)
-

TraceFilename

This property specifies the filename to write trace message to.

Declaration

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

Property Value

A string.

Remarks

This gets or sets the `TraceFilename` attribute of the connection string. The supported connection string attributes are listed in [Table 4-17](#).

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbConnectionStringBuilder Members](#)
 - [RdbConnectionStringBuilder Class](#)
-

TraceLevel

This property specifies the trace level used for tracing.

Declaration

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

Property Value

An integer.

Remarks

This gets or sets the `Tracelevel` attribute of the connection string. The supported connection string attributes are listed in [Table 4-17](#).

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbConnectionStringBuilder Members](#)
 - [RdbConnectionStringBuilder Class](#)
-

UserId

This property specifies the username to use for the connection.

Declaration

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

Property Value

A string.

Remarks

This gets or sets the `Username` attribute of the connection string. The supported connection string attributes are listed in [Table 4-17](#).

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbConnectionStringBuilder Members](#)
 - [RdbConnectionStringBuilder Class](#)
-

Version

This property specifies the Rdb version to use for the connection.

Declaration

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

Property Value

An integer.

Remarks

This gets or sets the `Version` attribute of the connection string. The supported connection string attributes are listed in [Table 4-17](#).

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbConnectionStringBuilder Members](#)
 - [RdbConnectionStringBuilder Class](#)
-

4.2.18.4 RdbConnectionStringBuilder Methods

`RdbConnectionStringBuilder` methods are listed in [Table 4-60](#).

TryGetValue

This method sets the value of the attribute specified into the given variable if the keyword attribute exists. It returns `true` if the attribute exists, else it returns `false`.

Declaration

```
// C#  
public override bool TryGetValue(string keyword, out object value)
```

Return Value

Returns `true` if keyword found, else `false`.

Remarks

This method sets the value of the attribute specified by `keyword` into `value` if the keyword attribute exists. It returns `true` if the attribute exists, else it returns `false`.

Example

```
// C#  
.  
.  
.  
object res;  
RdbConnectionStringBuilder sb = new RdbConnectionStringBuilder();  
sb.ConnectionString = cs;  
sb.TryGetValue("server", out res);  
Console.WriteLine(" server = " + res);  
.
```

.
.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbConnectionStringBuilder Members](#)
 - [RdbConnectionStringBuilder Class](#)
-

4.2.19 RdbFactory Class

The `RdbFactory` class represents a set of methods for creating instances of the Rdb Data Provider's implementation of the data source classes.

Class Inheritance

Object
 DbProviderFactory
 RdbFactory

Declaration

```
// C#  
public sealed class RdbFactory : DbProviderFactory
```

Thread Safety

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

Remarks

The `RdbFactory` class provides standard methods for instantiation of common Rdb Data Provider objects allowing for more generic coding of data access methods.

Example

```
// C#  
.br/>.br/>.br/>DbProviderFactory f =  
  DbProviderFactories.GetFactory("Oracle.DataAccess.RdbClient");  
DbConnection c = f.CreateConnection();  
.br/>.br/>
```

Requirements

Namespace: `Oracle.DataAccess.RdbClient`
Assembly: `Rdb.DataAccess.Rdb.dll`

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbFactory Members](#)
 - [RdbFactory Methods](#)
-

4.2.19.1 RdbFactory Members

`RdbFactory` members are listed in the following tables:

RdbFactory Methods

RdbFactory methods are listed in [Table 4-61](#).

Table 4-61 RdbFactory Methods

Method	Description
Equals	Inherited from Object (Overloaded).
CanCreateDataSourceEnumerator	Inherited from DbProviderFactory.
GetHashCode	Inherited from Object.
GetType	Inherited from Object.
CreateCommand	Create a new RdbCommand object.
CreateCommandBuilder	Create a new RdbCommandBuilder object.
CreateConnection	Create a new RdbConnection object.
CreateConnectionStringBuilder	Create a new RdbConnectionStringBuilder object.
CreateDataAdapter	Create a new RdbDataAdapter object.
CreateParameter	Create a new RdbParameter object.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
- [RdbFactory Methods](#)

4.2.19.2 RdbFactory Methods

RdbFactory methods are listed in [Table 4-61](#).

CreateCommand

This method returns an RdbCommand object.

Declaration

```
// C#  
public override DbCommand CreateCommand();
```

Return Value

Returns an RdbCommand object.

CreateCommandBuilder

This method returns an RdbCommandBuilder object.

Declaration

```
// C#  
public override DbCommandBuilder CreateCommandBuilder();
```

Return Value

Returns an RdbCommandBuilder object.

CreateConnection

This method returns an RdbConnection object.

Declaration

```
// C#  
public override DbCoconnection CreateConnection();
```

Return Value

Returns an `RdbConnection` object.

CreateConnectionStringBuilder

This method returns an `RdbConnectionStringBuilder` object.

Declaration

```
// C#  
public override DbCoConnectionStringBuilder  
CreateConnectionStringBuilder();
```

Return Value

Returns an `RdbConnectionStringBuilder` object.

CreateDataAdapter

This method returns an `RdbDataAdapter` object.

Declaration

```
// C#  
public override DbDataAdapter CreateDataAdapter();
```

Return Value

Returns an `RdbDataAdapter` object.

CreateParameter

This method returns an `RdbParameter` object.

Declaration

```
// C#  
public override DbParameter CreateParameter();
```

Return Value

Returns an `RdbParameter` object.

See Also:

- [Oracle.DataAccess.RdbClient Namespace](#)
 - [RdbFactory Members](#)
 - [RdbFactory Class](#)
-

4.3 Oracle Rdb Data Provider Enumerations

This section describes the enumeration Oracle Rdb Data Provider for .NET exposes for ADO.NET programmers. It is :

- `RdbCommandTypes` Enumeration

4.3.1 RdbCommandTypes Enumeration

The `RdbCommandTypes` enumeration specifies the values that can be used in conjunction with [RdbCommandType](#) property.

When the [RdbCommandType](#) property is set to `RdbCommandTypes.StoredProcedure`, set the [CommandText](#) property to the name of the stored procedure. The command executes this stored procedure when you call one of the `Execute` methods of an `RdbCommand` object.

When the [RdbCommandType](#) property is set to `RdbCommandTypes.ExternalProcedure`, set the [CommandText](#) property to the name of the external procedure. The command executes this external procedure when you call one of the `Execute` methods of an `RdbCommand` object.

[Table 4-62](#) lists all the `RdbCommandTypes` enumeration values with a description of each enumerated value.

Table 4-62 RdbCommandTypes Enumeration Members

Member Name	Description
<code>ExternalProcedure</code>	Indicates the name of an external procedure.
<code>StoredProcedure</code>	Indicates the name of an stored procedure.
<code>Text</code>	Indicates an SQL text command. (Default).

Requirements

Namespace: `Oracle.DataAccess.RdbClient`

Assembly: `Oracle.DataAccess.Rdb.dll`

Microsoft .NET Framework Version: 1.0 or later

Chapter 5 Oracle Rdb Schema Collections

ORDP.NET provides standard metadata collections as well as various Oracle Rdb database-specific metadata collections that can be retrieved through the `RdbConnection.GetSchema` API.

See Also:

- [Support for Schema Discovery](#)
 - [RdbFactory Class](#)
-

This chapter contains the following topics:

- [Common Schema Collections](#)
- [ORDP.NET-Specific Schema Collection](#)

5.1 Common Schema Collections

The common schema collections are available for all .NET Framework managed providers. ORDP.NET supports the same common schema collections.

See Also:

"Understanding the Common Schema Collections" in the MSDN Library

- [MetaDataCollections](#)
- [DataSourceInformation](#)
- [DataTypes](#)
- [Restrictions](#)
- [ReservedWords](#)

5.1.1 MetaDataCollections

[Table 5-1 MetaDataCollections](#) is a list of metadata collections that is available from the data source, such as tables, columns, indexes, and stored procedures.

Table 5-1 MetaDataCollections

Column Name	Data Type	Description
CollectionName	string	The name of the collection passed to the <code>GetSchema</code> 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.

5.1.2 DataSourceInformation

[Table 5-2 DataSourceInformation](#) lists `DataSourceInformation` information which may include these columns and possibly others.

Table 5-2 DataSourceInformation

Column Name	Data Type	Description
CompositeIdentifier SeparatorPattern	String	Separator for multipart names. This value is always: @ \
DataSourceProductName	string	Database Product name. This value is always: Oracle Rdb
DataSourceProduct Version	string	Database Product version. Note that this is the version of the database instance currently being accessed by DbConnection.
DataSourceProduct VersionNormalized	string	A normalized DataSource version for easier comparison between different versions. For example: DataSource Version: 7.2.4.1 Normalized DataSource Version: 07.02.04.01.00.
GroupByBehavior	GroupBy Behavior	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	IdentifierCa se	An enumeration that specifies whether or not to treat non-quoted identifiers as case sensitive.
OrderByColumnsIn Select	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.
ParameterMarker Pattern	string	The format of a parameter marker.
ParameterNameMax Length	int	Maximum length of a parameter name.
ParameterNamePattern	string	The format for a valid parameter name.
QuotedIdentifier Pattern	string	The format of a quoted identifier.
QuotedIdentifierCase	IdentifierCa se	An enumeration that specifies whether or not to treat quote identifiers as case sensitive.
StringLiteralPattern	string	The format for a string literal.
SupportedJoinOperators	Supported Join Operators	An enumeration indicating the types of SQL join statements supported by the data source.

5.1.3 DataTypes

[Table 5-3 DataTypes](#) 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` data type.

Table 5-3 DataTypes

Column Name	Data Type	Description
TypeName	string	The provider-specific data type name. Example: <code>TIMESTAMP</code> .
ProviderDbType	int	The provider-specific type value. Example: 93.
ColumnSize	long	The length of a non-numeric column or parameter. Example: 29.
CreateFormat	string	A format string that indicates how to add this column to a DDL statement. Example: <code>TIMESTAMP ({0})</code> .
CreateParameters	string	The parameters specified to create a column of this data type. Example: precision of fractional seconds.
DataType	string	The .NET type for the data type. Example: <code>System.DateTime</code> .
IsAutoIncrementable	bool	A boolean value that indicates whether or not this data type can be auto-incremented. Example: <code>false</code> .
IsBestMatch	bool	A boolean value that indicates whether or not this data type is the best match to values in the <code>DataType</code> column. Example: <code>true</code> .
IsCaseSensitive	bool	A boolean value that indicates whether or not this data type is case-sensitive. Example: <code>false</code> .
IsFixedLength	bool	A boolean value that indicates whether or not this data type has a fixed length. Example: <code>true</code> .
IsFixedPrecisionScale	bool	A boolean value that indicates whether or not this data type has a fixed precision and scale. Example: <code>false</code> .
IsLong	bool	A boolean value that indicates whether or not this data type contains very long data. Example: <code>false</code> .
IsNullable	bool	A boolean value that indicates whether or not this data type is nullable. Example: <code>true</code> .
IsSearchable	bool	A boolean value that indicates whether or not the data type can be used in a <code>WHERE</code> clause with any operator, except the <code>LIKE</code> predicate. Example: <code>true</code> .
IsSearchableWithLike	bool	A boolean value that indicates whether or not this data type can be used with the <code>LIKE</code> predicate. Example: <code>false</code> .
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.

Column Name	Data Type	Description
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.
MinimumVersion	String	The earliest version of the database that can be used. Example: 02.00.00.00.00.
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: <code>TIMESTAMP ' .</code>
LiteralSuffix	string	The suffix of a specified literal. Example: <code>' .</code>

5.1.4 Restrictions

[Table 5-4 Restrictions](#) lists Restrictions, including the following columns.

Table 5-4 Restrictions

Column Name	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.

5.1.5 ReservedWords

The `ReservedWords` collection exposes information about the words that are reserved by the database currently connected to ORDP.NET.

[Table 5-5 ReservedWords](#) lists the ReservedWords Collection.

Table 5-5 ReservedWords

Column Name	Data Type	Description
ReservedWord	string	Provider-specific reserved word.

5.2 ORDP.NET-Specific Schema Collection

Oracle Rdb Data Provider for .NET supports both the common schema collections described previously and the following Rdb-specific schema collections:

- [Tables](#)
- [Columns](#)
- [Views](#)
- [Synonyms](#)
- [Sequences](#)
- [Functions](#)
- [Procedures](#)
- [ProcedureParameters](#)
- [Indexes](#)
- [IndexColumns](#)
- [PrimaryKeys](#)
- [PrimaryKeyColumns](#)
- [ForeignKeys](#)
- [ForeignKeyColumns](#)
- [UniqueKeys](#)
- [UniqueKeyColumns](#)
- [Domains](#)
- [Outlines](#)
- [Constraints](#)

5.2.1 Tables

[Table 5-6 Tables](#) lists the column name, data type, and description of the Tables Schema Collection.

Table 5-6 Tables

Column Name	DataType	Description
TABLE_NAME	String	Name of the table.
TABLE_TYPE	String	Type of table [SYSTEM] [USER] [INFORMATION] [LOCAL TEMPORARY] [GLOBAL TEMPORARY].

5.2.2 Columns

[Table 5-7 Columns](#) lists the column name, data type, and description of the Columns Schema Collection.

Table 5-7 Columns

Column Name	DataType	Description
TABLE_NAME	String	Name of the table or view.
COLUMN_NAME	String	Name of the column.
TYPE_NAME	String	Name of the data type of the column.
COLUMN_SIZE	int	For char or date datatypes this is the maximum number of characters, for numeric or decimal types this is precision.
DECIMAL_DIGITS	int	Digits to right of decimal point in a number.
REMARKS	String	Comments associated with this column.
COLUMN_DEF	String	Default value, if any, for this column in textual form.

CHAR_OCTET_LENGTH	int	For char datatypes, this is the length in octets of this column.
ORDINAL_POSITION	int	Index of the column within the table or view (starting at 1) .
IS_NULLABLE	String	Specifies whether or not a column allows NULLs : [YES] [NO] .

5.2.3 Views

[Table 5-8 Views](#) lists the column name, data type, and description of the Views Schema Collection.

Table 5-8 Views

Column Name	Data Type	Description
VIEW_NAME	String	Name of the view.
VIEW_TYPE	String	Type of view : [SYSTEM] [USER] .

5.2.4 Synonyms

[Table 5-9 Synonyms](#) lists the column name, data type and description of the Synonyms Schema Collection.

Table 5-9 Synonyms

Column Name	Data Type	Description
SYNONYM_NAME	String	Name of the synonym.
REFERENCED_NAME	String	Name of object referenced.
REFERENCED_TYPE	String	Type of object referenced : [TABLE] [DOMAIN] [SYNONYM] [OTHER] .

5.2.5 Sequences

[Table 5-10 Sequences](#) lists the column name, data type, and description of the Sequences Schema Collection.

Table 5-10 Sequences

Column Name	Data Type	Description
SEQUENCE_NAME	String	Sequence name.
START_VALUE	int	Starting value of the sequence.
INCR_VALUE	int	Value by which sequence is incremented.
MIN_VALUE	int	Minimum value of the sequence.
MAX_VALUE	int	Maximum value of the sequence.
OBJECT_TYPE	String	Object Type of this sequence: [SYSTEM] [USER] .

5.2.6 Functions

[Table 5-11 Functions](#) lists the column name, data type, and description of the Functions Schema Collection.

Table 5-11 Functions

Column Name	Data Type	Description
OBJECT_NAME	String	Name of the function.
OBJECT_TYPE	String	Object Type of function: [SYSTEM] [USER] .

5.2.7 Procedures

[Table 5-12 Procedures](#) lists the column name, data type, and description of the Procedures Schema Collection.

Table 5-12 Procedures

Column Name	DataType	Description
OBJECT_NAME	String	Name of the procedure.
OBJECT_TYPE	String	Object Type of procedure: [SYSTEM] [USER] .

5.2.8 ProcedureParameters

[Table 5-13 ProcedureParameters](#) lists the column name, data type and description of the ProcedureParameters Schema Collection.

Table 5-13 ProcedureParameters

Column Name	DataType	Description
PROCEDURE_NAME	String	Name of the procedure.
COLUMN_NAME	String	Name of the parameter.
TYPE_NAME	String	Name of the data type of the column.
LENGTH	int	For char or date datatypes this is the maximum number of characters, for numeric or decimal this is the length in bytes of that datatype.
SCALE	int	Digits to right of decimal point in a number.
REMARKS	String	Comments associated with this parameter.
ORDINAL	int	Index of the parameter (starting at 1) .
DIRECTION	String	Specifies the direction of the parameter : [IN] [IN/OUT] [OUT] [RETVAL] [UNKNOWN] .

5.2.9 Indexes

[Table 5-14 Indexes](#) lists the column name, data type, and description of the Indexes Schema Collection.

Table 5-14 Indexes

Column Name	DataType	Description
INDEX_NAME	String	Name of the index.
TABLE_NAME	String	Name of the table.
INDEX_TYPE	String	Type of index : [SORTED] [HASHED] [SORTED RANKED] .
UNIQUE	String	Indicates whether the index is unique: [YES][NO] .

5.2.10 IndexColumns

[Table 5-15 IndexColumns](#) lists the column name, data type, and description of the IndexColumns Schema Collection.

Table 5-15 IndexColumns

Column Name	DataType	Description
INDEX_NAME	String	Name of the index.
ORDINAL	int	Position of column in index.
COLUMN_NAME	String	Name of the column.
DIRECTION	String	Direction of sorting of this column within the index: ascending [ASC] or descending [DESC].

5.2.11 PrimaryKeys

[Table 5-16 PrimaryKeys](#) lists the column name, data type, and description of the PrimaryKeys Schema Collection.

Table 5-16 PrimaryKeys

Column Name	DataType	Description
TABLE_NAME	String	Name associated with the table with the constraint definition.
CONSTRAINT_NAME	String	Name of the constraint definition.
EVALUATE	String	When the constraint will be evaluated: [COMMIT TIME] [VERB TIME (Nondeferrable)] [VERB TIME (Deferrable)].

5.2.12 PrimaryKeyColumns

[Table 5-17 PrimaryKeyColumns](#) lists the column name, data type, and description of the PrimaryKeyColumns Schema Collection.

Table 5-17 PrimaryKeyColumns

Column Name	DataType	Description
TABLE_NAME	String	Name associated with the table with the constraint definition.
CONSTRAINT_NAME	String	Name of the constraint definition.
COLUMN_NAME	String	Name of column.
POSITION	int	Position of column in key.

5.2.13 ForeignKeys

[Table 5-18 ForeignKeys](#) lists the column name, data type, and description of the ForeignKeys Schema Collection.

Table 5-18 ForeignKeys

Column Name	DataType	Description
TABLE_NAME	String	Name associated with the table with the constraint definition.
CONSTRAINT_NAME	String	Name of the constraint definition.
EVALUATE	String	When the constraint will be evaluated: [COMMIT TIME] [VERB TIME (Nondeferrable)] [VERB TIME (Deferrable)].

5.2.14 ForeignKeyColumns

[Table 5-19 ForeignKeyColumns](#) lists the column name, data type, and description of the ForeignKeyColumns Schema Collection.

Table 5-19 ForeignKeyColumns

Column Name	DataType	Description
TABLE_NAME	String	Name associated with the table referencing the foreign key.
CONSTRAINT_NAME	String	Name of the constraint definition referencing the foreign key.

Column Name	DataType	Description
COLUMN_NAME	String	Name of referencing column.
POSITION	String	Position of column in key.
PRIMARY_KEY_TABLE_NAME	String	Name of the table referenced.
PRIMARY_KEY_CONSTRAINT_NAME	String	Name of the constraint definition referenced.
PRIMARY_KEY_COLUMN_NAME	String	Name of the primary key column referenced.

5.2.15 UniqueKeys

[Table 5-20 UniqueKeys](#) lists the column name, data type, and description of the UniqueKeys Schema Collection.

Table 5-20 UniqueKeys

Column Name	DataType	Description
TABLE_NAME	String	Name associated with the table with the constraint definition.
CONSTRAINT_NAME	String	Name of the constraint definition.
EVALUATE	String	When the constraint will be evaluated: [COMMIT TIME] [VERB TIME (Nondeferrable)] [VERB TIME (Deferrable)].

5.2.16 UniqueKeyColumns

[Table 5-21 UniqueKeyColumns](#) lists the column name, data type, and description of the UniqueKeyColumns Schema Collection.

Table 5-21 UniqueKeyColumns

Column Name	DataType	Description
TABLE_NAME	String	Name associated with the table with the constraint definition.
CONSTRAINT_NAME	String	Name of the constraint definition.
COLUMN_NAME	String	Name of column.
POSITION	int	Position of column in key.

5.2.17 Domains

[Table 5-22 Domains](#) lists the column name, data type, and description of the Domains Schema Collection.

Table 5-22 Domains

Column Name	DataType	Description
DOMAIN_NAME	String	Name of the domain.
TYPE_NAME	String	Name of the data type of the domain.
COLUMN_SIZE	int	For char or date datatypes this is the maximum number of characters, for numeric or decimal types this is precision.
DECIMAL_DIGITS	int	Digits to right of decimal point in a number.
REMARKS	String	Comments associated with this domain.
COLUMN_DEF	String	Default value if any for this column in textual form.

CHAR_OCTET_LENGTH	int	For char datatypes this is the length in octets of this column.
OBJECT_TYPE	String	Object Type of domain : [SYSTEM] [USER] .

5.2.18 Outlines

[Table 5-23 Outlines](#) lists the column name, data type, and description of the Outlines Schema Collection.

Table 5-23 Outlines

Column Name	Data Type	Description
OUTLINE_NAME	String	Name of the outline.
OBJECT_TYPE	String	Object Type of outline: [SYSTEM] [USER] .

5.2.19 Constraints

[Table 5-24 Constraints](#) lists the column name, data type, and description of the Constraints Schema Collection.

Table 5-24 Constraints

Column Name	Data Type	Description
CONSTRAINT_NAME	String	Name of the constraint.
TABLE_NAME	String	Table name associated with constraint
CONSTRAINT_TYPE	String	Type of constraint: [CONDITIONAL] [PRIMARY KEY] [UNIQUE] [REFERENTIAL] [NOT NULL] [UNKNOWN] .
EVALUATE		When the constraint will be evaluated: [COMMIT TIME] [VERB TIME (Nondeferrable)][VERB TIME (Deferrable)] .
OBJECT_TYPE	String	Object Type of constraint [SYSTEM] [USER] .

Glossary

assembly

Assembly is Microsoft's term for the module that is created when a DLL or .EXE is compiled by a .NET compiler.

Binary Large Object (BLOB)

A large object datatype whose content consists of binary data. Additionally, this data is considered raw as its structure is not recognized by the database.

Character Large Object (CLOB)

The LOB datatype whose value is composed of character data corresponding to the database character set.

data provider

As the term is used with Rdb Data Provider for .NET, a data provider is the connected component in the ADO.NET model and transfers data between a data source and the DataSet.

dirty writes

Dirty writes means writing uncommitted or dirty data.

DDL

DDL refers to data definition language, which includes statements defining or changing data structure.

DOM

Document Object Model (DOM) is an application program interface (API) for HTML and XML documents. It defines the logical structure of documents and the way that a document is accessed and manipulated.

flush

Flush or flushing refers to recording changes (that is, sending modified data) to the database.

instantiate

A term used in object-based languages such as C# to refer to the creation of an object of a specific class.

Large Object (LOB)

The class of SQL datatype that is further divided into internal LOBs and external LOBs. Internal LOBs include BLOBs, CLOBs, and NCLOBs while external LOBs include BFILEs.

Microsoft .NET Framework Class Library

The Microsoft .NET Framework Class Library provides the classes for the .NET framework model.

namespace

- .NET:

A namespace is naming device for grouping related types. More than one namespace can be contained in an assembly.

- XML Documents:

A namespace describes a set of related element names or attributes within an XML document.

National Character Large Object (NCLOB)

The LOB datatype whose value is composed of character data corresponding to the database national character set.

octet

An 8-bit unit, usually referred to as BYTE

RdbDataReader

An `RdbDataReader` is a read-only, forward-only result set.

primary key

The column or set of columns included in the definition of a table's PRIMARY KEY constraint.

reference semantics

Reference semantics indicates that assignment is to a reference (an address such as a pointer) rather than to a value. See **value semantics**.

result set

The output of a SQL query, consisting of one or more rows of data.

savepoint

A point in the workspace to which operations can be rolled back.

stored procedure

A stored procedure is a block of SQL code that Rdb stores in the database and can be executed from an application.

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**.