

Oracle® Rdb Data Provider for .NET

Release Notes
Release 7.4.0.0.0
May 2020

Oracle Rdb Data Provider for .NET Release Notes, Release 7.4.0.0.0

Copyright © 2011, 2020 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	7
Chapter 1 Installing and Configuring	10
1.1 System Requirements	10
1.2 De-install previous Oracle Rdb Data Provider for .NET versions.....	11
1.3 Installing Oracle Rdb Data Provider for .NET	12
1.3.1 ORDP.NET 64 Bit support	12
1.3.2 ORDP.NET Assembly Version in Release 7.4.0.0.0 and Above	12
1.4 File Locations	13
1.5 Post Installation Procedures.....	14
1.6 Locating the ORDP.NET unmanaged DLLs.....	15
Chapter 2 Enhancements Provided in Oracle Rdb Data Provider for .NET Release 7.4.0.0.x	16
2.1 Windows 64 Bit support.....	16
Chapter 3 Problems Corrected	17
Chapter 4 Known Problems, Restrictions and Workarounds	18
4.1 Distributed Transaction Support not Available in This Version	18
4.2 Using FetchSize with Nested Queries and SQL/Services Type RdbConnection.....	18
4.3 Using FetchSize with Blobs and SQL/Services Type RdbConnection	19
4.4 SQL/Services Transaction Reusable Database Services are not Supported	20
4.5 Rdb Release 7.3.x.x.x Datatype Changes.....	20
Chapter 5 New Features and Corrections in Previous Releases	22
5.1 New Features for Release 7.3.6.0.0	22
5.2 Corrections in Release 7.3.6.0.0.....	22
5.2.1 Incorrect READ_WRITE transaction started	22
5.2.2 NOSUCHCUR error when using RdbDataAdapter.....	22
5.3 New Features for Release 7.3.5.0.0	23
5.3.1 SQL Returning Syntax Enhancement.	23
5.4 Corrections in Release 7.3.5.0.0.....	24
5.4.1 NLSLANG Problem with INSERT/UPDATE	24

5.4.2 Blob Problem with INSERT/UPDATE Statements.....	24
5.4.3 RdbDataReader Failure in Multi-Threaded Environment	24
5.4.4 NullReferenceException during ExecuteScaler.....	25
5.4.5 Error Message Truncation.....	25
5.5 <i>New Features for Release 7.3.4.0.0</i>	25
5.5.1 Additional Character Set support for NLSLANG.	25
5.6 <i>Corrections in Release 7.3.4.0.0</i>	26
5.6.1 Unbalanced Stack Exception	26
5.6.2 Updated Record Count Incorrect after CALL	26
5.6.3 Statement not Closed when Using Multiple ResultSets	26
5.6.4 Literals with As Clause Causes Exception	27
5.6.5 NullReferenceException on GetSchemaTable	27
5.7 <i>New Features for Release 7.3.3.0.0</i>	28
5.7.1 THIN Connections using SSL now allowed.....	28
5.7.2 BIG5 string literals now supported in SQL statements	28
5.7.3 Batched Processing now Available for JDBC Connectivity.....	29
5.7.4 PoolValidateConnection – New Connection Attribute.....	30
5.8 <i>Corrections in Release 7.3.3.0.0</i>	31
5.8.1 Read Write Transaction Started	31
5.8.2 Pooled Connections may use Excess Memory	31
5.8.3 Connection Pool Fails to Maintain Correct Number of Free Connections	31
5.9 <i>New Features for Release 7.3.2.2.x</i>	32
5.9.1 New Connection String Attribute - Keyinfo.....	32
5.10 <i>Corrections in Release 7.3.2.2.x</i>	32
5.10.1 Excessive IOs During Metadata Retrieval.....	33
5.10.2 Extra Key Information No Longer Retrieved by Default	33
5.10.3 Managed Debug Assistant Exception	33
5.10.4 Begin/End Block Syntax Error	34
5.10.5 SQL/Services Connectivity and INSERT/RETURNING.....	34
5.10.6 Exception Raised During Connection Finalizer	35
5.10.7 Problem with Update Statements and Thin Connectivity.....	35
5.10.8 RdbCommandBuilder Failure.....	36
5.10.9 BIG5 Character set problem	37
5.11 <i>New Features for Release 7.3.2.1.0</i>	37
5.11.1 Oracle Rdb Entity Framework Provider	37
5.12 <i>Corrections in Release 7.3.2.1.0</i>	38
5.12.1 PopulateReservedWords Problem on Pooled Connection.....	38
5.12.2 Domains collection Remarks and Default Value Columns not Converted.....	38
5.12.3 Retrieval of Scaled BIGINT Values Incorrect.....	38
5.12.4 Insert or Update of Scaled BIGINT Problem	39
5.12.5 RdbDataReader.GetSchemaTable() Problem	39
5.13 <i>New Features for Release 7.3.2.0.0</i>	40
5.13.1 Integration of Oracle Rdb Data Provider for .NET into Microsoft Visual Studio.....	40

5.14 Corrections in Release 7.3.2.0.0.....	40
5.14.1 CallBackOnCollectedDelegate Problem.....	40
5.15 New Features for Release 7.3.1.1.0.....	41
5.15.1 Additional DbType Support in RdbParameters.....	41
5.16 Corrections in Release 7.3.1.1.0.....	41
5.16.1 Unsupported DbType in RdbParameter Silently Ignored.....	42
5.16.2 Cast problem with Tiny Int.....	42
5.16.3 Possible Memory Leak when Updating Blob Columns.....	42
5.16.4 Statement Handles not Released.....	43
5.16.5 Retrieving Blob Data from Column with Associate Variables.....	43
5.16.6 Fraction of Seconds Trimmed from VMS DATE.....	44
5.16.7 Delimited Identifiers not Handled Correctly.....	44
5.17 New Features for Release 7.3.1.0.0.....	45
5.17.1 Pooled Connections.....	45
5.17.2 External Procedure Support.....	45
5.17.3 GetSchema Collections Support.....	46
5.18 Corrections in Release 7.3.1.0.0.....	48
5.18.1 RdbFactory Documentation Omission.....	48
5.18.2 RdbConnection.IsOpen() Method May Return Incorrect State Information.....	48
5.18.3 RdbConnection Connection State not Set Correctly When Connection Lost.....	49
5.18.4 RdbParameterCollection Methods Incorrectly Try to Cast the Value Object to RdbParameter Type.....	49
5.18.5 Incorrect Error Message Returned When Error Code is Unknown.....	49
5.18.6 Output Parameter Values on Stored Procedures Not Set.....	50
5.18.7 Explicit Transactions using SQL/Services Connectivity Fail to Disable AUTOCOMMIT of Statements.....	50
5.18.8 Incorrect Nested Query Exception Raised.....	51
5.18.9 StateChangeEvent Not Raised when RdbConnection is Closed.....	51
5.18.10 Un-named Parameters Names May Clash with User Defined Parameter Names.....	51
5.18.11 FetchSize is not Correctly Set for RdbCommands Containing Input or Output Parameters.....	52
5.18.12 Default FetchSize May be Set Incorrectly.....	52
5.18.13 ExecuteNonQuery() Returns Wrong Number of Records Affected.....	54
5.18.14 Concurrency Violation Exception Incorrectly Raised.....	54
5.18.15 Overflow/Underflow Exception not Raised on RdbParameter Assignments.....	54
5.18.16 Possible Memory Leak Due to RdbDataAdapter.Dispose() Problem.....	55
5.18.17 Memory Access Problem with SetStrVal.....	55
5.18.18 DataAdapter.Fill() May Lose Last Column in Select.....	56
5.18.19 DbType.Time Datatype not Handled Correctly by RdbParameter.....	57
5.18.20 Multiple Threads Connecting Concurrently May Cause an Access Violation.....	57
5.18.21 Setting RdbConnection.Autocommit May Cause Null Pointer Exception.....	58
5.18.22 Read-Write transactions Incorrectly Started Within ReadOnly Connections.....	58
5.18.23 Connection.State Incorrect After Execution of DataReader on Stored Procedure.....	58
5.19 New Features for Release 7.3.0.1.0.....	59

5.19.1 RdbFactory and RdbConnectionStringBuilder Classes	59
5.19.2 Limited TransactionScope Now Available	60
5.20 Corrections in Release 7.3.0.1.0	62
5.20.1 Input Parameter Marker Values not Correctly Set When Using the SQL/Services Client Connectivity	62
5.20.2 Using RdbCommand.CommandType Other Than TEXT May Cause SQL Syntax Exceptions	62
5.20.3 Cultural Information Settings may Cause Invalid Numeric Values to be Sent on SQL/Services RdbConnections	63
5.20.4 BigDecimal and Floating Numeric Problem When Using Thin Server Connectivity	63
5.20.5 DateTime Values Incorrect When Using SQL/Services Connectivity	63
5.20.6 RdbCommandBuilder Does Not Automatically Open Connection	64
5.20.7 RdbCommandBuilder May Lose Internal Statements	64
5.20.8 RdbCommandBuilder Fails to Recognize Unique Columns	65
5.20.9 RdbCommandBuilder Incorrectly used Blob Columns in Where Clause	65
5.20.10 Some LATIN1 Characters May Not be Retrieved Correctly From the Database	65

Send Us Your Comments

Oracle Rdb Data Provider for .NET Release Notes, Release 7.3.6.0.x.

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 release 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 release 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 Release Notes 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, "Installing and Configuring"](#)**
Describes how to install Rdb Data Provider for .NET and provides system requirements. Read this chapter *before* installing or using Rdb Data Provider for .NET.
- **[Chapter 2, "Enhancements Provided in Oracle Rdb Data Provider for .NET Release 7.3.6.0.x"](#)**
Describes new and changed features in Oracle Rdb Data Provider for .NET Release 7.3.6.0.x.
- **[Chapter 3, "Problems Corrected"](#)**
Describes problems corrected in Oracle Rdb Data Provider for .NET Release 7.3.6.0.x.
- **[Chapter 4, "Known Problems, Restrictions and Workarounds"](#)**
Describes known problems, restrictions, and workarounds for Oracle Rdb Data Provider for .NET Release 7.3.6.x.x.
- **[Chapter 5, "New Features and Corrections in Previous Releases"](#)**
Describes new and changed features and problems corrected in previous versions of Oracle Rdb Data Provider for .NET.

Related Documentation

For more information, see these 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 7.3 SQL Reference Manual*
- *Oracle Rdb Guide to SQL Programming*
- *Oracle SQL/Services Server Configuration Guide*
- *Guide to Using the Oracle SQL/Services (tm) Client API*
- *Oracle Rdb JDBC Driver User's Guide*
- *Oracle Rdb Data Provider for .NET Developer's Guide*

To download free release notes, installation documentation, white papers, or other collateral, please visit the 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 or simply ORDP.

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';
```

Examples of exception messages that may be raised by applications are also displayed using the above convention.

Passwords in Code Examples

For simplicity in demonstrating this product, code examples do not perform the password management techniques that a deployed system normally uses.

In a production environment, please follow your corporate password management guidelines and other security recommendations.

▲ [contents](#)

Chapter 1 Installing and Configuring

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

1.1 System Requirements

Oracle Rdb Data Provider for .NET requires the following products to be installed:

Software	Minimum Version
Microsoft .NET Framework	V4.6.1
Windows Server 2012 and above, Windows 8 or Windows 10	As released

If SQL/Services will be used for server connectivity then the following software is required.

On the CLIENT:

Software	Minimum Version
Rdb SQL/Services client	Release 7.3.0.1.0

Note:

The installation of ORDP.NET will install the minimum SQL/Services client software required for ORDP - SQL/Services connectivity from a Microsoft Windows environment, so a separate installation of SQL/Services client software is not required. During installation a template version of `SQSAPI32.INI` will be copied to the ORDP.NET installation directory. This file may be modified to suit your SQL/Services environment as specified in the SQL/Services documentation. To change your SQL/Services setting this INI file must be copied to your system Windows directory. See your SQL/Services documentation for more information on `SQSAPI32.INI`.

On the SERVER:

Software	Minimum Version	
	Alpha	Integrity
HP OpenVMS	V8.2	V8.2-1
Oracle SQL/Services	Release 7.3	Release 7.3
TCP/IP Services for OpenVMS	V5.5	V5.5-11
Oracle Rdb	Release 7.3	Release 7.3

If JDBC thin servers will be used for server connectivity then the following software is required.

On SERVER:

Software	Minimum Version	
	Alpha	Integrity
HP OpenVMS	V8.2	V8.2-1
Oracle JDBC for Rdb	Release 7.3.0.0.0	Release 7.3.0.0.0
HP Java™ SDK/RTE	Release 1.5.0-1	Release 1.5.0-1
Oracle Rdb	Release 7.3	Release 7.3

Note:

CREATE/DROP database is only supported when using Thin connectivity in conjunction with Oracle JDBC for Rdb server release 7.3.4.0.0 and later.

See Also:

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

1.2 De-install previous Oracle Rdb Data Provider for .NET versions

If you have a previously installed version of ORDP.NET you must de-install this prior to installing this version of ORDP.NET.

If you installed ORDP.NET as part of Release 7.3.2.0.0 ORDT you should use the MSI file provided with that installation kit to de-install ORDT. If you no longer have this MSI file then you should use Control Panel / Add or Remove Programs to remove ORDT.

If the prior version of ORDP.NET was lower than Release 7.3.2.0.0 and you have changed your machine.config to add references to ORDP.NET then you should remove these changes prior to installing the new version of ORDP.NET.

In particular, if the following sections exist in your current machine.config, they should be removed:

```
<system.data>
  <DbProviderFactories>
    <add name="Oracle Rdb Data Provider" . . .
```

and

```
<configuration>
  <configSections>
    <section name="oracle.dataaccess.rdbclient" . . .
```

1.3 Installing Oracle Rdb Data Provider for .NET

Starting with release 7.3.2.0.0 of ORDP.NET, ORDP.NET is installed as part of Oracle Rdb Developer Tools for Visual Studio (ORDT).

If you do not require the Visual Studio integration features of ORDT then when you install ORDT you may select a customized installation and choose only to install ORDP.NET.

During the installation of ORDT the strongly named ORDP.NET assembly will be added to your system Global Assembly Cache so that a single assembly instance may be used by multiple applications.

Note:

Starting with release 7.3.2.0.0 of ORDP.NET, ORDP.NET will no longer provide a standard assembly.

The ORDT installation will also update your system PATH variable to include the ORDP.NET installation directory and will make the appropriate changes to your `machine.config` file so that ORDP.NET will be registered correctly for use with .NET applications.

The final step of the installation merges the ORDT Visual Studio extensions into your Visual Studio environment.

Note:

The re-configuration of Visual Studio that occurs at the end of the installation and de-installation may take some time to complete, as it must merge all the VSPackages present within the Visual Studio environment.

Please do not interrupt the installation/de-installation during this process as it may leave your Visual Studio environment in an unstable state.

Please refer to the Installation section of the ORDT Release Notes for more information on ORDT and ORDP.NET installation. You should also refer to your Microsoft .NET documentation for information on strongly named assemblies and the Global Assembly Cache.

1.3.1 ORDP.NET 64 Bit support

Starting with ORDP.NET Release 7.4.0.0.0, the ORDT installation Zip file provides both a 32 Bit and a 64 Bit version of the ORDP.NET assemblies.

ORDP.NET requires unmanaged DLLs to be installed on your system to allow ORDP.NET to use SQL/Services to access Oracle Rdb databases.

Both 32 Bit and 64 Bit versions of these unmanaged DLLs are provided.

1.3.2 ORDP.NET Assembly Version in Release 7.4.0.0.0 and Above

Starting with ORDP.NET Release 7.4.0.0, the ORDT installation procedure will install only one version of the ORDP.NET assembly file Oracle.DataAccess.Rdb.dll onto your system.

Note that the ORDP.NET assemblies retains the same PublicKeyToken as previous versions of ORDP.NET.

The ORDP for .NET framework version 4.0 assembly will have the following specifications:

- Name =Oracle.DataAccess.Rdb,
- Version=7.4.0.0
- Culture=neutral
- PublicKeyToken= 24caf6849861f483

References made to prior version ORDP.NET strongly named assemblies in your configuration and system registry files may need to be changed to reflect the new assembly information to ensure that applications on your system will use the new version of ORDP.NET from the System Assembly Cache.

Note:

The ORDT installation procedure will automatically update the standard .NET machine.config file and Visual Studio registration files to reference the correct version of the ORDP.NET assembly.

However there may be optional features within Visual Studio or third party products that may require changes to the referenced ORDP.NET assembly version in order to function properly. Please refer to your Visual Studio or third party product documentation for further details.

1.4 File Locations

The ORDP-specific installation files will be copied to the ORDP sub-directory under the installation directory which, by default, will be:

<system program files>\Oracle\ORDP

Where <system program files> will depend on your windows system:

Typically:

- On 32 Bit Windows machines –
C:\Program Files\
• On 64 Bit Windows machines using the 32 Bit installation –
C:\Program Files (x86)\
- On 64 Bit Windows machines using the 64 Bit installation –
C:\Program Files\
• On 64 Bit Windows machines using the 64 Bit installation –
C:\Program Files\
C:\Program Files (x86)\

Note:

The 64 Bit installation will also install the 32 bit ORDP assemblies to C:\Program Files (x86)\Oracle\ORDP

Note:

If you choose to install ORDT to a directory other than the default, you may be required to provide more information to your applications that are using ORDT or ORDP.NET to help them determine where to find the ORDP.NET unmanaged DLL files.

See [Locating the ORDP.NET unmanaged DLLs](#) for more details.

The following assembly, library and documentation files will be copied to the appropriate installation sub-directories:

File	Description
ORDP_dev_guide_<version>.pdf	ORDP Developer Guide
ORDP_rel_notes_<version>.pdf ORDP_rel_notes_<version>.txt	ORDP Release Notes
Oracle.DataAccess.Rdb.dll	The ORDP assembly
rdbnet.dll	Required library for Rdb access. Two version will be installed, 32 and 64 bit
sqsapi32.dll	Required library for SQL/Services access
sqsapi64.dll	Required library for SQL/Services 64 bit access
sqsapi32.ini	Configuration template file for use with SQL/Services
sqsapi64.ini	Configuration template file for use with SQL/Services in 64 bit environment

In addition, the ORDP.NET assembly will be added to your Global Assembly Cache.

1.5 Post Installation Procedures

Starting with Release 7.3.2.0.0, the ORDT installation will update your system PATH variable to include the ORDP.NET installation directory, so this no longer has to be done as a post-installation step.

The ORDT installation will also update your `machine.config` file to register ORDP.NET correctly for use with Visual Studio and other .NET applications.

However, there are still some manual post-installation steps that have to be carried out.

If specific SQL/Services configuration settings are required, the template SQSAPI32.INI file provided with the ORDP.NET kit must be modified to reflect the required settings and moved to the windows sub-directory of your system directory. See your SQL/Services documentation for more information.

The actual post-installation steps you will have to carry out will depend on how the DLL files will be used and may require changes to your development environment to either include this new directory path or to move the provided DLL files to the appropriate third-party directory.

Please refer to the documentation provided with your development software to determine what steps may be involved in order to use the Oracle Rdb Data Provider for .NET classes and libraries.

1.6 Locating the ORDP.NET unmanaged DLLs

By default, ORDP will attempt to load the correct ORDP.NET DLL files from the default ORDP installation directory, typically:

- On 32 Bit Windows machines –
C:\Program Files\ORACLE\ORDP\NETv4.0\
- On 64 Bit Windows machines running a 32 application –
C:\Program Files (x86)\ORACLE\ORDP\NETv4.0\
- On 64 Bit Windows machines running a 64 application –
C:\Program Files\ORACLE\ORDP\NETv4.0\

If you have chosen to install ORDP to a directory other than the default installation directory, or if .NET cannot locate the ORDP assemblies in these directories, .NET will carry out its standard search protocol to locate the DLLs. (See your Microsoft .NET documentation for more information).

If the DLLs still cannot be found, your application will fail to execute correctly.

You may choose to redirect where .NET first tries to locate the DLLs by setting up the correct path to the DLLs in the “appSettings” section of your applications configuration file (<app>.config) using the following two keys:

- dllImport32
- dllImport64

For Example:

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>

  <appSettings>
    <add key="dllImport32" value="E:\MYDIR\Win32\bin\Debug\"/>
    <add key="dllImport64" value="E:\MYDIR\x64\bin\Debug\"/>
  </appSettings>
</configuration>
```

The “dllImport[32 or 64]” key redirects ORDP to look in the provided path for the appropriate ORDP unmanaged DLLs.

If .NET fails to locate the DLL files it will then try the standard DLL search paths.

See your Microsoft .NET documentation for more information on DLL search paths and **app.config** files.

Chapter 2 Enhancements Provided in Oracle Rdb Data Provider for .NET Release 7.4.0.0.x

This chapter describes new and changed features in Oracle Rdb Data Provider for .NET release 7.4.0.0.x.

2.1 Windows 64 Bit support

ORDP.NET release 7.4.0.0.0 now provides support for Windows 64 Bit applications.

▲ [contents](#)

Chapter 3 Problems Corrected

This chapter describes problems corrected in Oracle Rdb Data Provider for .NET release 7.4.0.0.x.

None.

▲ [contents](#)

Chapter 4 Known Problems, Restrictions and Workarounds

This chapter describes known problems, restrictions, and workarounds for Oracle Rdb Data Provider for .NET release 7.4.0.x.x.

4.1 Distributed Transaction Support not Available in This Version

The use of distributed transactions is not currently supported in ORDP.

4.2 Using FetchSize with Nested Queries and SQL/Services Type RdbConnection

When SQL/Services is used to provide RdbConnection connectivity, the use of FetchSize with a value other than one (1) may cause problems if nested queries are involved.

For example the following code showing a nested query may fail if the FetchSize for the connection or command is anything other than 1.

```
// C#
.
.
.
RdbCommand employees = conn.CreateCommand();
employees.CommandText =
@"select employee_id,first_name,last_name from
employees limit to 5 rows";
RdbDataReader employee = employees.ExecuteReader();

while (employee.Read())
{
    String id = employee.GetString(0);
    RdbCommand degrees = conn.CreateCommand();
    degrees.CommandText =
@"select degree,degree_field from degrees where employee_id='" + id + "'";
    RdbDataReader degree = degrees.ExecuteReader();
    while (degree.Read())
    {
        String type = degree.GetString(0);
        String field = degree.GetString(1);
        Log("Degree:" + type + " " + field);
    }
    degree.Close();
}
employee.Close();
.
.
```

The problem here is that when the FetchSize is greater than 1 an optimization called FETCH_MANY is used in the SQL/Services connection to allow multiple records to be sent from the server to the client in a single network IO.

SQL/Services does not allow any other fetch operation to be done on this same connection until all the records returned by the FETCH_MANY have been read by the application. This means that loops such as shown above cannot be done while FETCH_MANY is in operation as the code will try to do a fetch of the inner reader before getting all the records from the FETCH_MANY on the outer loop.

If fetch of another record set is attempted while FETCH_MANY is active the following exception is raised by SQS/Services:

```
batched sqlsrv_execute or sqlsrv_fetch_many context is active.
```

To work-around this limitation, FetchSize should be set to 1 for queries where nesting will be done.

For example:

```
// C#
.
.
.
RdbCommand employees = conn.CreateCommand();
employees.FetchSize = 1;
employees.CommandText =
@"select employee_id,first_name,last_name from
employees limit to 5 rows";
RdbDataReader employee = employees.ExecuteReader();
while (employee.Read())
.
.
.
```

4.3 Using FetchSize with Blobs and SQL/Services Type RdbConnection

For the same reason as described in the previous section, queries containing Blob columns should also be carried out using FetchSize = 1.

Oracle Rdb requires the use of a separate query to obtain the contents of a Blob (List of Byte Varying) columns. This retrieval query is setup and executed internally when the contents of a Blob column are retrieved.

As this may involve a nesting of queries, the FetchSize must be set to 1. Otherwise an exception as shown in the previous section will be raised.

The following is an example of coding the retrieval of Blob information.

```
// C#
.
.
.
// Get LATIN1 encoding as it's the closest to DEC_MCS
```

```

Encoding Latin1 = Encoding.GetEncoding("ISO-8859-1");

RdbCommand cmd = new RdbCommand(
@"SELECT employee_id, resume FROM resumes WHERE
    employee_id = '00164'", db);
cmd.CommandType = CommandType.Text;
cmd.Connection.Open();
cmd.FetchSize = 1;

RdbDataReader reader = cmd.ExecuteReader();
while (reader.Read())
{
    string id = reader.GetValue(0).ToString();
    Int length = (int)reader.GetBytes(1, 0, null, 0, Int32.MaxValue);
    byte[] resume = new byte[length];
    reader.GetBytes(1, 0, resume, 0, length);
    Console.WriteLine("resume for id = "+id);
    Console.WriteLine(Latin1.GetString(resume));
}
.
.
.

```

4.4 SQL/Services Transaction Reusable Database Services are not Supported

ORDP.NET does not support SQL/Services connections using executors that have transaction reusable database services enabled.

4.5 Rdb Release 7.3.x.x.x Datatype Changes

Rdb Release 7.3.x.x.x has made some changes to the datatypes used by some of the standard functions such as `Count()`, which will now be returned to applications as a 64 bit value.

This change may cause some problems with existing applications written using ORDP as the object type returned by some operations will change to reflect the new datatypes returned by Rdb.

In particular, care should be taken when using ORDP methods that return objects as their return values, as implicit casting may no longer be valid.

For example:

```

// C#
RdbCommand CmdObj = new RdbCommand("", conn);
CmdObj.CommandText = "select count(*) from employees";
int count = (int)CmdObj.ExecuteScalar();

```

This code will work fine with release 7.2.x.x.x Rdb and below but will throw the following exception when accessing release 7.3.x.x.x Rdb database using release 7.3.x.x.x SQS/Services:

```
System.InvalidCastException: Specified cast is not valid.
```

The reason for the exception has to do with implicit boxing and unboxing of objects carried out by C#.

When accessing Rdb databases using versions prior to Rdb 7.3 the execution of the above `ExecuteScalar` method would return an `int` (or `System.Int32`) boxed object. This object may be cast to an `int` as C# will implicitly unbox the object as `Int32` first prior to casting.

Using Rdb V7.3, the object returned by `ExecuteScalar` method in the example shown above will now be a `long` (or `System.Int64`). Implicit boxing requires that the object be unboxed to exactly the same type as it was boxed, however the cast to `int` in the expression will try to unbox the object as `Int32` which is not allowed and thus an exception is raised.

The application must use the appropriate datatype variable if using implicit unboxing, for example:

```
//C#  
long count = (long)CmdObj.ExecuteScalar();
```

Or should explicitly unbox the object correctly prior to casting:

```
//C#  
int count = (int)(long)CmdObj.ExecuteScalar();
```

▲ [contents](#)

Chapter 5

New Features and Corrections in Previous Releases

5.1 New Features for Release 7.3.6.0.0

This section describes new and changed features in the Oracle Rdb Data Provider for .NET release 7.3.6.0.0.

None.

5.2 Corrections in Release 7.3.6.0.0

This section describes software errors corrected in the Oracle Rdb Data Provider for .NET release 7.3.6.0.0.

5.2.1 Incorrect READ_WRITE transaction started

Release 7.3.6.0.0.

Also available in Release 7.3.5.0.2

According to the ORDP Developer Guide, unless otherwise changed by using specific transaction switches on a connection, or explicit transaction statements, ORDP should only start a read-write transaction if the SQL operation requires it, such as INSERT and UPDATE.

Release 7.3.3.0.0 addressed a problem seen during the initial setting of the transaction type for the connection that caused ORDP to start a read-write transaction even if the SQL statement does not require it; for example, a SELECT statement.

However the fix did not catch all of the situations where this problem may arise hence further changes have been made in Release 7.3.6.0.0 to try to address all of the remaining situations.

5.2.2 NOSUCHCUR error when using RdbDataAdapter

Release 7.3.6.0.0.

A problem introduced in Oracle JDBC for Rdb Release 7.3.5.0.0. may cause ORDP queries to fail during the closing of a Rdb-based DataAdapter.

When JDBC connectivity is being used in conjunction with an RdbDataAdapter, the close-down of the RdbDataAdapter may cause the following exception to be raised:

```
Oracle.DataAccess.RdbClient.RdbException : in <rdbjdbcsrv:close_cursor>%SQL-F-NOSUCHCUR, Cursor CC_00000002
```

This problem may be resolved by updating your JDBC servers to JDBC Release 7.3.5.1.0. or later.

A work-around for this problem is to use JDBC servers running with JDBC releases earlier than 7.3.5.0.0 or later than 7.3.5.0.9 or to use SQL/Services connectivity.

5.3 New Features for Release 7.3.5.0.0

This section describes new and changed features in the Oracle Rdb Data Provider for .NET release 7.3.5.0.0.

5.3.1 SQL Returning Syntax Enhancement.

Also in Release 7.3.4.0.9

ORDP will now allow a simplified Returning clause on SQL INSERT and UPDATE statements.

In prior versions to obtain a “Returning” value from the results of an INSERT or UPDATE statement a parameter marker and an associated **RdbCommand** parameter must be supplied:

```
// C#
.
.
.
cmd = new RdbCommand(
@"Update JOBS set job_title = 'Test2' where job_code='PRGM'
  returning job_title into :name",
  conn);

cmd.Parameters.Add("name", DbType.AnsiString,
ParameterDirection.Output);
string result = (string)cmd.ExecuteScalar();
.
.
.
```

To simplify coding, ORDP now allows you to omit the parameter marker and parameter. The appropriate syntax changes will automatically be made by ORDP prior to the SQL statement being executed to ensure the presentation of valid Oracle Rdb SQL syntax to the underlying database server.

```
// C#
.
.
.
cmd = new RdbCommand(
@"Update JOBS set job_title = 'Test2' where job_code='PRGM'
  returning job_title",
  conn);

string result = (string)cmd.ExecuteScalar();
.
.
.
```

This modification is only applicable to statements that have a single returning clause.

Note : As no **RdbCommand** parameter is provided , to obtain the value returned , an **ExecuteScaler** method call must be made. Other **Execute** methods may be used but the return value will not be available to the user code.

5.4 Corrections in Release 7.3.5.0.0

This section describes software errors corrected in the Oracle Rdb Data Provider for .NET release 7.3.5.0.0.

5.4.1 NLSLANG Problem with INSERT/UPDATE

Also fixed in Release 7.3.4.0.2.

A problem in the handling of input parameters in SQL prepared statements prevents the correct encoding of characters during the processing of INSERT and UPDATE operations, when NLSLANG is used on the connection string.

The NLSLANG feature allows ORDP to correctly encode non-MCS characters stored in database DEC_MCS columns. For example it is possible that existing older databases may have non-MCS characters such as Kanji or Hebrew characters stored in text columns that by default are considered to be encoded in DEC_MCS characters. NLSLANG set to the appropriate character set allows ORDP to correctly interpret these characters when retrieving or inserting them into the database.

A problem exists during the processing of input parameters such that all characters being inserted into DEC_MCS columns will be interpreted as DEC_MCS irrespective of the NLSLANG setting. This only effects input of text characters, the retrieval of text from the database still carries out the correct character encoding operations.

5.4.2 Blob Problem with INSERT/UPDATE Statements

Also fixed in Release 7.3.4.0.4.

A problem in the insertion of blob data during the processing of INSERT and UPDATE statements may cause ORDP to raise a SQL Syntax error.

During the normal insert or update, ORDP attempts to correctly derive the designated table into which blob data may be stored. A parsing problem introduced in ORDP release 7.3.4.0.0 cause ORDP to incorrectly delimit the table name, turning the table name into a delimited identifier. If the table name in the original SQL statement text is already uppercased, the delimiting of the identifier does not cause an error, however if the original table name was in lower or bumpy-case, the resultant delimited identifier will fail to match with the actual table name in the database causing a SQL syntax error to be raised..

5.4.3 RdbDataReader Failure in Multi-Threaded Environment

Also fixed in Release 7.3.4.0.7.

When ORDP is used within a multi-threaded application, Oracle advises that only one thread should access a RdbConnection at any one time. If more than one thread uses the same connection it is possible that they may interfere with each other and may cause exceptions to be raised on the connection.

ORDP is able to correctly manage multithreads if only one thread is assigned to any single concurrent connection, however a problem in the way connection information is passed between

underlying ORDP objects can cause ORDP to fail to handle multithreads properly under this environment.

The problem is seen when using the RdbDataReader to access the underlying data. When multiple threads access different DataReaders concurrently it is possible that the termination of one DataReader may interfere with others causing various protocol errors in the underlying database connection. These problems may be seen more often in the Entity Framework environment and when using SQL/Services connectivity. Errors raised include error messages similar to:

- `InvalidOperationException` New Connection request while old Connection executing
- `InvalidOperationException` Rdb Connection not open
- A batched `sqlsrv_execute` or `sqlsrv_fetch_many` context is active.
- Operation is not valid due to the current state of the object.

It is also possible that AccessViolations may occur.

This problem has now been fixed.

5.4.4 NullReferenceException during ExecuteScaler

Also fixed in Release 7.3.4.0.8.

The execution of `ExecuteScaler()` method of the `RdbCommand` class may raise the following exception:

```
System.NullReferenceException: Object reference not set to an instance of an object.  
at Oracle.DataAccess.RdbClient.RdbCommand.ExecuteScaler()
```

This problem has now been fixed.

5.4.5 Error Message Truncation

Also fixed in Release 7.3.4.0.9.

Error messages may be truncated during the handling of exceptions for SQS-based connections. The truncation may remove leading portions of the message text include the Rdb Error message code values. In some cases this problem may lead to unexpected AccessViolations during exception handling.

This has now been fixed.

5.5 New Features for Release 7.3.4.0.0

This section describes new and changed features in the Oracle Rdb Data Provider for .NET release 7.3.4.0.0.

5.5.1 Additional Character Set support for NLSLANG.

The configuration switch “@NlsLang” now supports more character sets.

Character sets such as ISOLatinHebrew, ISOLatin9, DEC_KANJI and others may now be used with the “@NlsLang” configuration switch to alter ORDP interpretation of literals and column data that would normally be interpreted as DEC_MCS (ISOLATIN1) characters.

See the **RdbConnection Properties** section of the **Oracle Rdb Data Provider for .NET Developer’s Guide** for more information.

5.6 Corrections in Release 7.3.4.0.0

This section describes software errors corrected in the Oracle Rdb Data Provider for .NET release 7.3.4.0.0.

5.6.1 Unbalanced Stack Exception

An exception similar to the following may be raised while debugging:

```
A call to PInvoke function 'Oracle.DataAccess.Rdb!  
Oracle.DataAccess.RdbClient.Common.SQS::StartTrans2' has  
unbalanced the Stack
```

The following entry points are known to have this problem:

- StartTrans
- StartTrans2
- SetDefTrans2
- SetBinVal
- SetBinValL

This is a further manifestation of the problem “Managed Debug Assistant Exception” referenced in release notes for release 7.3.2.2.0.

This problem has now been fixed.

5.6.2 Updated Record Count Incorrect after CALL

A successful call to the RdbCommand.ExecuteNonQuery() method when using command text containing a SQL CALL statement returns the value 1 when no records have been affected.

This type of statement now returns the value 0 indicating no records were updated.

5.6.3 Statement not Closed when Using Multiple ResultSets

The use of multiple SQL statements is supported when using RdbDataReader, however a problem in the transition between results sets may prevent the underlying SQL statements from closing correctly, resulting in metadata locking if any metadata operations are attempted subsequent to the execution of the reader using the same connection.

An example of using multiple result sets may be seen below:

```

// C#
.
.
.
using (RdbCommand cmd =
    new RdbCommand("select * from jobs; select * from employees;", conn)
{
    Using RdbDataReader reader = cmd.ExecuteReader()

    {
        .
        .
        .
    }
}

```

5.6.4 Literals with As Clause Causes Exception

The use of the AS clause when using literals values may cause the Rdb Data Provider to fail with an “Index was out of range” exception when parsing the statement.

An example of the type of statement that may raise the exception may be seen below:

```
cmd.CommandText = "SELECT 'a' AS XYZ from jobs"
```

This problem only occurs if there are no table column present in the selection criteria.

5.6.5 NullReferenceException on GetSchemaTable

ORDP will throw a `NullReferenceException` if the `RdbDataReader.GetSchemaTable` method is called when the data reader is accessing the results of a stored procedure.

The following example assumes the following stored procedure exists:

```
PROCEDURE DoNothing(); BEGIN END
```

The following code will throw a `NullReferenceException`:

```

// C#
.
.
.
RdbCommand cmd = new RdbCommand("DoNothing", conn);
cmd.CommandType = CommandType.StoredProcedure;
using (RdbDataReader reader = cmd.ExecuteReader())
{
    DataTable dt = reader.GetSchemaTable();
}
.
.
.

```

5.7 New Features for Release 7.3.3.0.0

This section describes new and changed features in the Oracle Rdb Data Provider for .NET release 7.3.3.0.0.

5.7.1 THIN Connections using SSL now allowed

ORDP will now allow SSL connections to be made when using THIN connectivity to JDBC servers.

The simplest way to tell ORDP to use SSL for THIN connectivity is to add the connection string qualifier “sslmode=Required” to the connection string:

```
string cnxString =  
@"Server=localhost:1707;Database=user1:[murray]mf_personnel;  
User Id=murray;Pwd=secret;SSLmode=Required;"
```

See the **Oracle Rdb Data Provider Developer’s Guide** for more information on SSL usage.

This feature is only available when using THIN connectivity.

5.7.2 BIG5 string literals now supported in SQL statements

ORDP will now allow BIG5 characters to be used in string literals used within SQL statements.

By default ORDP expects that characters within SQL text string literals to be LATIN1 characters and will convert them as such when delivering the SQL text to Rdb.

For example, the following SQL text literal value used in the select statement will be interpreted by ORDP as having 2 LATIN1 characters instead of 2 BIG5 characters.

```
select '測試' FROM rdb$database";
```

The consequence of this is that the results returned to the application will no longer be the original BIG5 characters.

ORDP now allows the correct translation of BIG5 characters within literal strings, but to enable this alternate character interpretation within the ORDP driver, the connection string qualifier “nlsLang=BIG5” must be added to the connection string:

```
string cnxString =  
@"Server=MYNODE.ME.COM:GENERIC;Database=user1:[murray]mf_personnel;User  
Id=murray;Pwd=secret;Type=SQS;nlsLang=BIG5";
```

This feature is available in both SQS and THIN connectivity.

See the Oracle Rdb Data Provider for .NET Developer’s Guide for more information.

5.7.3 Batched Processing now Available for JDBC Connectivity

The ability to carry out database update operations in batch is now available when using JDBC connectivity.

The RdbDataAdapter now supports ADO.NET Batch update.

The RdbDataAdapter now allows INSERT, UPDATE, and DELETE operations from a DataSet or DataTable to be sent to the server as a group, instead of sending one operation at a time.

Previously, when updating a database with changes from a DataSet, the Update method of RdbDataAdapter performed updates to the database one row at a time.

RdbDataAdapter now exposes the UpdateBatchSize property. Setting the UpdateBatchSize to a positive integer value causes updates to the database to be sent as batches of the specified size.

Setting the UpdateBatchSize to 0 will cause the DataAdapter to use the largest batch size that the server can handle. Setting it to 1 disables batch updates, as rows are sent one at a time.

For example:

```
//C#
.
.
.

// assume we have an Rdb table BAT_TABLE(f1 char(10), f2 char(10))

RdbDataAdapter rda = new RdbDataAdapter();
rda.SelectCommand = new RdbCommand(
    "Select * from BAT_TABLE", conn);

// Set the INSERT command and parameter.
rda.InsertCommand = new RdbCommand(
    "INSERT INTO BAT_TABLE (F1,F2) VALUES (@ID,@Name);",
    conn);
rda.InsertCommand.Parameters.Add("@ID", DbType.AnsiString, 20, "F1");
rda.InsertCommand.Parameters.Add("@Name", DbType.AnsiString, 20, "F2");

rda.InsertCommand.UpdatedRowSource = UpdateRowSource.None;

// build some records for testing
DataTable dt = new DataTable("Test");
rda.Fill(dt);
dt.PrimaryKey = new DataColumn[] { dt.Columns["F1"] };

DataTable dt2 = dt.Clone();
int maxR = 21;
for (i = 0; i < maxR; i++)
{
    DataRow row1 = dt2.NewRow();
    row1["F1"] = "Row_"+i;
    row1["F2"] = "descr_"+i;
    dt2.Rows.Add(row1);
}

// Set the batch size.
```

```
adapter.UpdateBatchSize = 5;

// Execute the update. This should send records to the
// server 5 at a time
adapter.Update(dt2);

.
.
.
```

This feature is currently only available for use with JDBC connectivity.

See your Microsoft ADO.NET documentation for more information about UpdateBatchSize and batch processing.

5.7.4 PoolValidateConnection – New Connection Attribute

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.

Prior to this release, if a pooled connection lost its underlying connectivity and was then subsequently reallocated to a new connection request, no exception would be raised during the connection open. However the first database operation on that connection would fail with a network, socket or database error.

This behavior can now be changed by using the new PoolValidateConnection connection option:

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

If this option is used, ORDP will check that the connection is viable before releasing it to the new connection request. During the viability check, 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.

As this operation does cost some network resources and increases the amount of time taken to do the connection, ORDP does not carry it out by default.

This feature is available in both SQS and THIN connectivity and is only applicable to POOLEDSQS and POOLEDTHIN connection types. This feature is available when using either implicit or explicit pool management.

See the Oracle Rdb Data Provider for .NET Developer's Guide for more information.

5.8 Corrections in Release 7.3.3.0.0

This section describes software errors corrected in the Oracle Rdb Data Provider for .NET release 7.3.3.0.0.

5.8.1 Read Write Transaction Started

According to the ORDP Developer Guide, unless otherwise changed by using specific transaction switches on a connection, or explicit transaction statements, ORDP should only start a read-write transaction if the SQL operation requires it, such as INSERT and UPDATE.

A problem during the initial setting of the transaction type for the connection causes ORDP to start a read-write transaction even if the SQL statement does not require it; for example, a SELECT statement.

This problem has been fixed.

5.8.2 Pooled Connections may use Excess Memory

A problem in the internal operation of connection pooling when using connection types POOLEDSQL or POOLEDTHIN may cause a slow increase in memory usage every time a pooled connection is reused.

This problem does not affect the functionality of the pooled connection, but may eventually lead to an out-of-memory error in user applications using ORDP with connection type POOLEDSQL or POOLEDTHIN.

This problem has been fixed.

5.8.3 Connection Pool Fails to Maintain Correct Number of Free Connections

(BUG 17921100)

If a connection pool has a non-zero value specified for its `PoolMinFree` attribute, it should strive to maintain that number of free connections while the pool is open. In addition, a non-zero value specified for its `PoolMaxFree` attribute should determine the maximum number of free connections that will be kept around while the pool is still active.

A problem in the determination of the actual count of free connections already in the pool will cause the pool manager to fail to maintain this minimum number of free connections.

This same problem in determining the free connection count within the pool also prevents the pool manager from correctly determining when the pool capacity is exceeded.

These problem occur for both POOLEDSQS and POOLEDTHIN connections.

▲ [contents](#)

5.9 New Features for Release 7.3.2.2.x

This section describes new and changed features in the Oracle Rdb Data Provider for .NET release 7.3.2.2.x.

5.9.1 New Connection String Attribute - Keyinfo.

Release 7.3.2.2.2.

The changes made to Release 7.3.2.2.0 of ORDP, highlighted in release note “**Extra Key Information No Longer Retrieved by Default**” shown below, help improve performance over the network by reducing the queries sent to Rdb to obtain metadata information about columns used in SQL statements.

This extra information, pertaining to the primary and unique key status, may be used by internal Visual Studio DDEX operations to help build queries and display column information.

As the cited release note states, you may tell ORDP to continue to collect this extra key information by prefixing the SQL statement text with the keyword “{keyinfo}”.

Alternatively, starting with release 7.3.2.2.2, ORDP will now also allow you to specify that key information should be retrieved by using the connection string attribute, “keyinfo”. For example:

```
string cnxString =  
@"Server=MYNODE.ME.COM:GENERIC;Database=user1:[murray]mf_personnel;User  
Id=murray;Pwd=secret;Type=SQS;Fetchsize=100;keyinfo=true;"
```

If you have a large number of queries that require key information, or you are unable to alter queries to add the “{keyinfo}” prefix to the SQL text, you may use this new connection string attribute instead.

If this attribute is present and is set to TRUE, all queries compiled within the connection will request the extra key information from Rdb during SQL statement preparation.

5.10 Corrections in Release 7.3.2.2.x

This section describes software errors corrected in the Oracle Rdb Data Provider for .NET release 7.3.2.2.x.

5.10.1 Excessive IOs During Metadata Retrieval

Several SQL queries used by ORDP to retrieve metadata associated with compiled queries may cause excessive IOs on database systems where the column or table definitions are volatile and have had many revisions during the life of the database.

This problem has been fixed.

5.10.2 Extra Key Information No Longer Retrieved by Default

The query used by ORDP to retrieve additional information about how query columns are being used within the database, for example checking if the query column is a primary key, is no longer being executed by default.

Performance concerns raised after the first release of ORDP 7.3.2.1.0 have led to this change. By default, ORDP no longer retrieves extra information from the database pertaining to the Primary Key and Unique Key status of query columns.

This information may be useful for some implementations of Entity Framework applications and other DDEX features but is not used in general by ORDP.

Key information may still be retrieved by ORDP, if applications require it, by adding the new escape qualifier, "{keyinfo}" to the start of select query text, for example:

```
RdbCommand cmd = new RdbCommand(
    "{keyinfo}select * from employees", conn);
```

5.10.3 Managed Debug Assistant Exception

Changes in the way Visual Studio and .NET carry out stack cleanup now requires that unmanaged C++ code be prototyped explicitly with the **CDecl** calling convention.

The unmanaged code for SQL/Services connectivity only explicitly declared the calling convention on some of the prototyped routines.

In more recent version of Visual Studio 2010 this may cause the following exception to be raised while debugging:

```
A call to PInvoke funtion 'Oracle.DataAccess.Rdb!
Oracle.DataAccess.RdbClient.Common.SQS::Init' has unbalanced the
Stack
```

This problem only occurs while debugging applications using ORDP within Visual Studio 2010.

This problem has now been fixed.

5.10.4 Begin/End Block Syntax Error

SQL Text containing a `BEGIN/END` compound statement that has a `RETURNING` clause in it may be problematic during the parsing and recognition of parameter markers.

If the statement contains a `RETURNING` clause with associated parameter marker followed immediately by a semicolon “;” and then further statements or an `END` clause without any intervening whitespace, ORDP incorrectly parses the statement and will cause a `SYNTAX` error to be raised when the statement is compiled by the underlying Rdb database system.

The problem only occurs when named parameters using the “@” prefix are used.

If JDBC connectivity is used then the `SYNTAX` error may be overwritten by a subsequent `NULL` reference pointer exception.

The following is an example of the type of query that may present with this problem:

```
"begin insert into id_tab1(c2) values (@nam) returning c1 into @id;end"
```

A workaround is to place whitespace between the parameter marker and semicolon:

```
"begin insert into id_tab1(c2) values (@nam) returning c1 into @id ;end"
```

Another workaround is to use “?” parameter markers instead:

```
"begin insert into id_tab1(c2) values (?) returning c1 into ?;end"
```

This problem has now been fixed.

5.10.5 SQL/Services Connectivity and INSERT/RETURNING

The use of the `RETURNING` clause within `INSERT` statements is currently limited by SQL/Services to only return the `DBKEY` value of the inserted row. Operations such as returning `IDENTITY` columns or computed-by columns by using the `RETURNING` clause in an `INSERT` statement is not currently supported by SQL/Services.

This limitation is inherited by ORDP when you use SQL/Services (SQS) connectivity. Although the underlying database will correctly compile the SQL statement containing the `RETURNING` clause, no data values are actually returned to the application except in the case where the returning column is the `DBKEY`.

For example if `C1` is an `IDENTITY` column the following SQL Text:

```
"insert into id_tab1(c2) values (?) returning c1 into ?"
```

will compile correctly but the parameter associated with the `C1` columns will always be null.

The following workaround can be used; place the insert statement within a BEGIN/END block:

```
"begin insert into id_tab1(c2) values (?) returning c1 into ? ; end"
```

In addition, if the workaround is used, you must ensure that the statement is issued under a READ_WRITE transaction. The automatic determination of transaction type by ORDP, where READ_WRITE transactions are automatically started if the operation is an update type, does not operate with compound statements.

Another workaround is to use JDBC connectivity as the RETURNING clause is fully supported by JDBC.

SQL/Services engineering is aware of the problem and an enhancement request has been raised.

Meanwhile, ORDP has now been changed to recognized INSERT statements containing the RETURNING clause and automatically apply the workaround of adding the BEGIN/END wrapper to the statement prior to executing if the connectivity is SQS.

ORDP now also keeps track of the fact that this wrapped statement is an INSERT and now correctly applies the appropriate transaction semantics.

This problem has now been fixed.

5.10.6 Exception Raised During Connection Finalizer

ORDP incorrectly raises an exception during garbage collection if a connection is found to be no longer viable. This exception will cause the .NET application that was using the connection to terminate.

The behaviour for exception handling during finalizers changed in .NET V2.0 to force the CLR to terminate the application if an exception is raised.

In order to reduce the impact on existing applications, ORDP has been modified to no longer throw an exception if an underlying connection is found to be broken during connection cleanup.

An error message will be logged, but the exception will no longer be escalated to the application.

Exceptions will still be raised and escalated to the application if the connection is found not to be viable during execution of application or ORDP driver code other than the ORDP connection finalizer code.

This problem has now been fixed.

5.10.7 Problem with Update Statements and Thin Connectivity

Recent changes to ORDP to allow the correct handling of "RETURNING" in both INSERT and UPDATE statements introduced a problem in how ORDP carries out the execution of statements that use parameters when passing the request to a thin server.

For example, an update statement using parameter markers such as the following :

```
command.CommandText = "UPDATE testtab set c1 = :column1";
```

will cause an exception to be raised on the client-side application when executed:

```
Invalid request <unknown: . . .  
@rdb.Client.ROLL_TRANS
```

In addition, if server tracing is enabled, the thin server will log the following exception :

```
oracle.rdb.jdbc.common.RdbException: Invalid request <idle>
```

A workaround is to not use parameter markers and add the update value as a literal directly into the SQL text:

```
command.CommandText = "UPDATE testtab set c1 = 'myNewValue'";
```

This problem only affects THIN connectivity and has now been fixed.

5.10.8 RdbCommandBuilder Failure.

Fixed in release 7.3.2.2.2.

The changes made to the first Release 7.3.2.2.0 of ORDP, highlighted in release note “**Extra Key Information No Longer Retrieved by Default**” above, will prevent `RdbCommandBuilder` operations from completing correctly.

During the operation of the `RdbCommandBuilder`, ORDP checks to see if the SQL text references a table that has either a primary or unique key associated with it. If not, the building of appropriate update statements cannot proceed.

The changes mentioned in the release note cited above, prevents ORDP from registering the appropriate key information which leads to the following failure during execution of the `RdbCommandBuilder`:

```
RdbCommandBuilder cannot operate on tables with no unique or key columns.
```

This has now been changed. The `RdbCommandBuilder` will now correctly retrieve the appropriate key column information prior to attempting to build associated commands,

A work around is to use the “{keyinfo}” prefix in the SQL text for the select command you setup prior to calling the `RdbCommandBuilder` class, for example:

```
// C#
```

```

.
.
.
conn.Open();

string selectCmd = "{keyinfo}SELECT * from job_info WHERE id = @P1";

RdbCommand cmd = new RdbCommand(selectCmd);
cmd.Connection = conn;

.
.
.

da = new RdbDataAdapter(cmd);

DataTable dt = new DataTable();
da.Fill(dt);

RdbCommandBuilder bldr = new RdbCommandBuilder(da);
.
.
.

```

5.10.9 BIG5 Character set problem

Fixed in release 7.3.2.2.3.

ORDP failed to correctly deliver the content of table columns encoded using the Chinese BIG5 character set.

Within .NET string data is encoded using UNICODE and part of the operation that ORDP carries out when retrieving data from the Rdb database is to correctly encode the contents of text columns into UNICODE data for use within .NET applications.

A problem in how ORDP handled data conversion when using SQL/Services connectivity (type=SQS) caused incorrect data encoding and subsequent corruption of the Chinese BIG5 characters.

This problem only occurs when using SQL/Services connectivity.

This has now been fixed.

5.11 New Features for Release 7.3.2.1.0

This section describes new and changed features in the Oracle Rdb Data Provider for .NET release 7.3.2.1.0.

5.11.1 Oracle Rdb Entity Framework Provider

Additional classes and templates have been added to ORDP to allow ORDP to carry out the appropriate operations required of an Entity Framework Provider within .NET framework 3.5 and .NET 4.0.

Please refer to your .NET framework documentation for information on Entity Framework.

Also, please refer to your ORDT documentation for information on the Oracle Rdb Entity Framework provider and examples on how to use the provider.

▲ [contents](#)

5.12 Corrections in Release 7.3.2.1.0

This section describes software errors corrected in the Oracle Rdb Data Provider for .NET release 7.3.2.1.0.

5.12.1 PopulateReservedWords Problem on Pooled Connection

The following exception may be raised when using POOLEDTHIN or POOLEDSQS type connections:

```
"Index was outside the bounds of the array."  
At Oracle.DataAccess.RdbClient.Common.ConnectionSchema.  
    PopulateReservedWords(String collectionName,  
    String normalizedDBVersion)\r\n
```

This problem has been fixed.

5.12.2 Domains collection Remarks and Default Value Columns not Converted

The Remarks column of the Rdb specific Collection **Domains** was not correctly converted to String on retrieval of the column description segmented string.

This problem has been fixed.

5.12.3 Retrieval of Scaled BIGINT Values Incorrect

A problem in the conversion of scaled BIGINT values returned by the SQL/Services interface to ORDP may cause incorrect values to be returned to DECIMAL variables.

In some circumstances the following error is raised during data retrieval:

```
Error: Input string was not in a correct format.
```

This problem is only seen when using SQL/Services connectivity.
A possible workaround is to use JDBC thin connectivity.

This problem has now been fixed.

5.12.4 Insert or Update of Scaled BIGINT Problem

During an insert or update operation on a scaled BigInt column, any digits after the decimal point are incorrectly stripped from the value, resulting in an incorrect value being saved into the database.

The following code fragment shows an example of an operation that will show this problem:

```
// C#
.
.
.
// create table in db with
// "create table bigint_trunc1 (c1 bigint(4));"

RdbCommand cmd = new RdbCommand(
    "Insert into bigint_trunc1(c1) values (@d_value)", conn);
RdbParameter prmDecimal = cmd.Parameters.Add(
    new RdbParameter("@d_value", System.Data.DbType.Decimal));
prmDecimal.Value = 1.234;
cmd.ExecuteNonQuery();
.
.
.
```

Inspection of the value within the database will show that the value stored will be "1.0000" and not "1.2340".

This problem is only seen when using SQL/Services connectivity. A possible workaround is to use JDBC thin connectivity.

This problem has now been fixed.

5.12.5 RdbDataReader.GetSchemaTable() Problem

Several properties returned by the GetSchemaTable() method of the RdbDataReader class may show wrong information.

The following properties may return incorrect return values:

- "isExpression"
- "isReadOnly"
- "isAutoIncrement"

Note:

If you are using SQL/Services connectivity this fix will be immediately available after the installation of release 7.3.2.1.0 of ORDT.

If you are using JDBC thin connectivity, the fix also requires appropriate changes in the JDBC servers. In order for this fix to work while using JDBC thin connectivity, the connected servers must be running release 7.3.0.2.0 or greater of Oracle JDBC for Rdb.

This problem has now been fixed.

5.13 New Features for Release 7.3.2.0.0

This section describes new and changed features in the Oracle Rdb Data Provider for .NET release 7.3.2.0.0.

5.13.1 Integration of Oracle Rdb Data Provider for .NET into Microsoft Visual Studio

Oracle Rdb Data Provider for .NET is now integrated into the Microsoft Visual Studio for .NET IDE.

Please refer to the Oracle Rdb Developer Tools for Visual Studio Developer's Guide and the information on the Oracle Rdb web site for more details.

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

5.14 Corrections in Release 7.3.2.0.0

This section describes software errors corrected in the Oracle Rdb Data Provider for .NET release 7.3.2.0.0.

5.14.1 CallbackOnCollectedDelegate Problem

Bug 9304303.

ORDP.NET uses callbacks into .NET-managed code to handle things like raising exceptions.

The ORDP.NET code tries to maintain correct delegate references to these callback routines in a manner that prevents the .NET garbage collector from harvesting them, thus allowing them to be available for the duration of the callback to managed code.

Unfortunately a problem in how the delegates were set up when multiple concurrent connections are made from the same application, left the delegate methods vulnerable for garbage collection.

The problem may raise an exception similar to the following:

```
CallbackOnCollectedDelegate was detected
Oracle.DataAccess.Rdb!Oracle.DataAccess.RdbClient.Common_SQS+
exceptionCallback :Invoke
```

Depending on the amount of memory used and exactly when the garbage collector runs, it is possible that the delegates may be removed during the call into the underlying Rdb system, and if one of these callback delegates happens to be required, for example if an exception is raised, then this problem may occur.

In general the following conditions have to be met before this type of exception may be raised:

- The .NET application must open multiple concurrent connections.
- Memory used within the application and underlying ORDP.NET methods must be high enough to cause the .NET garbage collector to be invoked.
- An Exception is raised during an operation within a concurrent connection stream that was open prior to the last connection stream created.

This problem has been fixed.

5.15 New Features for Release 7.3.1.1.0

This section describes new and changed features in the Oracle Rdb Data Provider for .NET release 7.3.1.1.0.

5.15.1 Additional DbType Support in RdbParameters

The following DbType datatypes were supported for RdbParameter datatypes in Oracle Rdb Data Provider for .NET release 7.3.1.0.0 and prior:

- `DbType.String`
- `DbType.SByte`
- `DbType.Int16`
- `DbType.Byte`
- `DbType.Int32`
- `DbType.UInt16`
- `DbType.Int64`
- `DbType.UInt32`
- `DbType.UInt64`
- `DbType.Time`
- `DbType.Date`
- `DbType.DateTime`
- `DbType.Single`
- `DbType.Double`
- `DbType.Decimal`
- `DbType.Binary`

The following additional DbTypes are now supported starting with release 7.3.1.1.0:

- `DbType.AnsiString`
- `DbType.AnsiStringFixedLength`
- `DbType.StringFixedLength`

5.16 Corrections in Release 7.3.1.1.0

This section describes software errors corrected in the Oracle Rdb Data Provider for .NET release 7.3.1.1.0.

5.16.1 Unsupported DbType in RdbParameter Silently Ignored

Setting the DbType value of an RdbParameter to a datatype not supported by Oracle Rdb Data Provider for .NET was silently ignored.

During the processing of the RdbParameter by Prepared Statements, data values for RdbParameters with unsupported DbType would not be recognized as being set and will raise an error similar to the following when the statement is executed:

```
Oracle.DataAccess.RdbClient.RdbException: Missing IN or OUT parameter at
index: 1 at
Oracle.DataAccess.RdbClient.Common.PreparedStatement.Execute() in
. . .
```

At the time the RdbParameter is created, ORDP will now raise the following exception if the DbType specified for the RdbParameter is unsupported:

```
Oracle.DataAccess.RdbClient.RdbException: Unsupported DbType
```

This problem has been fixed.

5.16.2 Cast problem with Tiny Int

Retrieving a tiny int column value from a table may raise the following exception:

```
Specified cast is not valid. @rdb.Client.GetByteVal
```

A work around is to cast the column to another appropriate SQL datatype in your SQL statement. For example:

```
Select cast (tinycolumn as integer) from table1;
```

This has now been fixed.

5.16.3 Possible Memory Leak when Updating Blob Columns

During the processing of update statements containing references to Blobs (list of byte varying) columns, ORDP incorrectly compiles the statement twice.

Only the second statement handle is released when the statement is closed which will lead to Rdb statement handles still being active within the current connection.

These active handles use memory and resources within ORDP and the Rdb system which may eventually lead to problems due to insufficient resources in long running connections.

In addition, these compiled queries may prevent metadata operations from occurring on the affected tables within the current connection.

This has now been fixed.

5.16.4 Statement Handles not Released

A problem in the ORDP driver code prevents some statement handles from being released correctly.

This problem is only found when the associated SQL statement is other than a `SELECT` statement, for example, `INSERT` or `UPDATE` statements

During the rundown of the `RdbCommand` object, the `Dispose()` method failed to close the associated statements and thus did not release the Rdb handles for those statements.

This accumulation of statement handles over time within connections that are open for lengthy periods may cause memory problems.

This problem has now been fixed and the `RdbCommand.Dispose()` method now correctly releases the underlying statement handles.

Note:

Oracle recommends that to help reduce unnecessary overhead, especially when dealing with statements that access Blob data, that any `RdbCommand` that has SQL text containing operations other than a `SELECT` should be explicitly disposed as soon as they are no longer used, by calling the `RdbCommand.Dispose()` method.

If the `RdbCommand` is not explicitly disposed of, the .NET garbage collector will eventually carry out an implicit dispose of the object. But until the object is disposed, underlying ORDP, SQL and Rdb resources will be held and not released and may have an accumulative resource depletion effect within the underlying database system and any associated Thin Server or SQL/Services service.

5.16.5 Retrieving Blob Data from Column with Associate Variables

A problem in the ORDP driver code prevents the retrieval of Blob data when associate variables are used in conjunction with the list of byte varying column name in the SQL text.

For example, the following SQL text will raise an exception during query compilation:

```
// C#
RdbCommand CmdObj = new RdbCommand("", conn);
CmdObj.CommandText = "select resume RES from resumes";
byte[] resume = (byte[])cmd.ExecuteScalar();
```

This code will raise the following exception:

```
%SQL-F-FLDNOTCRS, Column RES was not found in the tables in current scope
```

The associate variable RES hides the actual source column name from ORDP.

ORDP incorrectly uses the associate variable name when building the underlying SQL statements used to retrieve the Blob data from the database.

A work-around for this problem is to not use associate variables on list of byte varying columns.

This has now been fixed.

5.16.6 Fraction of Seconds Trimmed from VMS DATE

During retrieval of data from the Rdb database, the ORDP code handling the conversion of DATE VMS data to the appropriate .NET datatype inadvertently trimmed off the fraction of seconds value.

This problem only occurs with SQS connections and where the Rdb datatype of the source column or variable is DATE VMS.

This has now been fixed.

5.16.7 Delimited Identifiers not Handled Correctly

Using delimited identifiers for column or table names in SQL text may cause problems with the parsing of the statement by the ORDP drivers.

An error in how double quotation characters are parsed within the SQL text prevents ORDP from correctly handling delimited identifiers, for example the following code snippet:

```
// C#
.
.
.
DataSet ds = new DataSet();
string selectString = @"select ""JOB_CODE"" from JOBS";
RdbDataAdapter adapter = new RdbDataAdapter(
    selectString, conn);
adapter.Fill(ds);
.
.
.
```

may raise the following exception:

```
Oracle.DataAccess.RdbClient.RdbException occurred
Message="Unhandled exception@Statement.Execute :
```

```
Index was out of range. Must be non-negative and less than the size of the collection"
```

This has now been fixed.

[▲ contents](#)

5.17 New Features for Release 7.3.1.0.0

This section describes new and changed features in the Oracle Rdb Data Provider for .NET release 7.3.1.0.0.

5.17.1 Pooled Connections

Two new types of connections are now available:

- POOLEDSQS
- POOLEDTHIN

These new connection types are Pooled Connection variants of the SQS and THIN connection types that allow simple connection pooling. Their behavior is basically the same as the underlying basic connection type except that during connection open the connection will be taken from a pool of established connections and on close will be returned to the pool of connections again.

The connection pools established are application instance local, that is, the pools of connections are only available to the current application instance and its associated threads. Separately running application instances will have their own pools of connections that are not sharable across instances.

This type of pooling was primarily created to help reduce the connection overhead in those applications that may be constantly connecting and disconnecting from the underlying data source, and may be using separate threads to carry out discrete operations on the same data source.

RdbConnection has new methods that allow the setup and maintenance of these connection pools.

Please refer to the *Oracle Rdb Data Provider for .NET Developer's Guide* for information on the use of connection pools.

5.17.2 External Procedure Support

Rdb External Procedures can now be used within ORDP in a similar manner to *StoredProcedures*.

RdbCommand has been enhanced to allow the specification of a new command type called ***RdbCommandTypes.ExternalProcedure***, allowing Rdb external procedures to be invoked directly using the procedure name as the sole contents of the *RdbCommand* CommandText. For example:

```
// C#
```

```

.
.
.
RdbCommand cmd = cn.CreateCommand();
cmd.RdbCommandType =
    RdbCommandTypes.ExternalProcedure;
cmd.CommandText = "MY_EXTERNAL_PROCEDURE";
RdbCommandBuilder.DeriveParameters(cmd);
RdbParameterCollection coll = cmd.Parameters;
.
.

```

Please refer to the *Oracle Rdb Data Provider for .NET Developer's Guide* for information on the use of External Procedures.

5.17.3 GetSchema Collections Support

The `RdbConnection.GetSchema()` method may now be used to retrieve information about connected databases.

The following Collections are now supported:

- MetaDataCollections
- DataSourceInformation
- DataTypes
- Restrictions
- ReservedWords
- Tables
- Columns
- Views
- Synonyms
- Sequences
- Functions
- Procedures
- ProcedureParameters
- Indexes
- IndexColumns
- PrimaryKeys
- PrimaryKeyColumns
- ForeignKeys
- ForeignKeyColumns
- UniqueKeys
- UniqueKeyColumns
- Domains
- Outlines
- Constraints

Please refer to the *Oracle Rdb Data Provider for .NET Developer's Guide* for information on Supported Collections.

See Also:

"Understanding the Common Schema Collections" in the MSDN Library

▲ [contents](#)

5.18 Corrections in Release 7.3.1.0.0

This section describes software errors corrected in the Oracle Rdb Data Provider for .NET release 7.3.1.0.0.

5.18.1 RdbFactory Documentation Omission

RdbFactory was introduced in the ORDP release 7.3.0.1.0 but information about enabling the use of the RdbFactory code within .NET was omitted from the release notes for ORDP release 7.3.0.1.0.

Before you can use RdbFactory it must be registered as a Data Provider Factory in your .NET machine.config file. 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 the %systemroot%\Microsoft.NET\Framework folder on your computer, for example on Windows 7 the file specification for the configuration file would typically be:

```
C:\WINDOWS\Microsoft.NET\Framework\{Version Number}\CONFIG\machine.config
```

If you wish to use the RdbFactory facility in conjunction with ORDP release 7.3.0.1.0 then the following must be added to your .NET machine.config file:

```
.  
. .  
  <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=7.3.0.0,  
    Culture=neutral,PublicKeyToken=3a9ba85e401d6be5" />  
  .  
  .
```

Note that this is applicable only if you wish to use ORDP release 7.3.0.1.0. To use ORDP release 7.3.1.0.0, see [Assembly Version Change for Release 7.3.1.0.0](#)

5.18.2 RdbConnection.IsOpen() Method May Return Incorrect State Information

The IsOpen() method of RdbConnection was incorrectly determining the state of the connection by testing if the current state of the connection was *ConnectState.Open*. However as there could be other states that also imply that the connection is open, this method may return the value *false* even when the connection is currently open and active.

This has now been fixed.

5.18.3 RdbConnection Connection State not Set Correctly When Connection Lost

If during an active connection the connection to the underlying data source server becomes unavailable, the state of the `RdbConnection` object was not being set appropriately and would show `ConnectionState.Open` even when the connection was known to be lost.

On lost connections, the `RdbConnection` state is now correctly set to `ConnectionState.Broken`.

This has now been fixed.

5.18.4 RdbParameterCollection Methods Incorrectly Try to Cast the Value Object to RdbParameter Type

A problem in the handling of the `val` parameter passed to the following methods may cause ORDP to raise an `System.InvalidCastException`:

- `RdbParameterCollection.Add(Object val)`
- `RdbParameterCollection.Contains(Object val)`

The `val` parameter to these methods should have been handled as an `RdbParameter Value` object rather than an `RdbParameter` object.

A work-around for this problem for the `Add()` method is to use the named parameter method instead:

```
RdbParameterCollection.Add(String paramName, Object value)
```

This has now been fixed.

5.18.5 Incorrect Error Message Returned When Error Code is Unknown

When an `RdbException` is raised with an unknown internal error code the following message should be set in the `RdbException` and returned by `RdbException.GetMessage()` method:

```
Unknown error code : NNNNNN
```

Instead the following message was returned:

```
Non supported SQL92 token at position NNNNNN
```

This has now been fixed.

5.18.6 Output Parameter Values on Stored Procedures Not Set

After execution of an `RdbCommand` of type `CommandType.StoredProcedure` the *Value* objects of output parameters were not being set, leaving the *Value* objects as null objects.

For example:

```
// C#
.
.
.
RdbCommand cmd = new RdbCommand("MY_STORED_PROC", conn);
cmd.CommandType = CommandType.StoredProcedure;

RdbParameter p1 = new RdbParameter();
p1.ParameterName = "out1";
p1.DbType = DbType.Int32;
p1.Direction = ParameterDirection.Output;
cmd.Parameters.Add(p1);

int result = cmd.ExecuteNonQuery();
System.out.println("result = " + p1.Value.ToString());
.
.
.
```

The query as show above would raise the following exception when trying to execute the last line:

```
System.NullReferenceException:
Object reference not set to an instance of an object.
```

This has now been fixed.

5.18.7 Explicit Transactions using SQL/Services Connectivity Fail to Disable AUTOCOMMIT of Statements

When explicit transactions are enabled by using `RdbConnection.BeginTransaction` or by using a mechanism such as enlistment, the default behavior of ORDP to AUTOCOMMIT updateable SQL statements should automatically be disabled.

A problem in the ORDP code specific to `RdbConnections` using SQL/Services prevented this AUTOCOMMIT from being disabled.

Thus transactions may be found to be committed prematurely due to this AUTOCOMMIT action.

This has now been fixed.

5.18.8 Incorrect Nested Query Exception Raised

When an `Exception` is raised during the execution of a `RdbConnection.ExecuteRead()`, or `RdbConnection.ExecuteScalar()` the underlying `RdbDataReader` object does not get correctly removed from the `RdbConnection`'s list of active readers which may cause the following `RdbException` to be incorrectly raised on subsequent calls to `RdbConnection.ExecuteRead()` or `RdbConnection.ExecuteScalar()` on the same `RdbConnection`:

```
An open RdbDataReader has FetchSize greater than 1,
query nesting is not allowed in this case
```

A workaround for this problem is to close and re-open the connection after the primary exception is raised, or set the current `RdbConnection` fetch size to 1.

This has now been fixed.

5.18.9 StateChangeEvent Not Raised when RdbConnection is Closed

Whenever the `RdbConnection` *ConnectionState* changes all handlers registered as `RdbStateChangeEventHandlers` should be notified of the change of connection state.

During the close of the `RdbConnection` the connection state is correctly set to *ConnectionState.Closed*. However, a problem event notification during the close of the `RdbConnection` prevents the *ConnectionState.Closed* event from being sent to the event handlers.

This has now been fixed.

5.18.10 Un-named Parameters Names May Clash with User Defined Parameter Names

During the parsing of a SQL statement, all parameters found in the statement are recorded to allow for the correct assignment of parameters declared in the Parameter List by the caller. If a simple parameter marker is used in the SQL text (i.e. "?"), this "un-named" parameter is given a name internally to allow for correct positional alignment of the parameter. The following naming convention for these "un-named" parameters is used:

```
PARAMETER<position index>
```

However, the name generated is too likely to clash with user provided names for parameters in the statement, and if this happens, incorrect assignments of parameters or incorrect parameter lookup inside the ORDP engine may occur.

For example, due to the use of the same naming format for user defined parameter names and internal parameter name execution of the following SQL text:

```
SELECT last_name INTO ? FROM employees WHERE
```

```
employee_id = @PARAMETER1 AND first_name = @PARAMETER2"
```

may cause the following exception to be raised:

```
RdbCommand cannot find parameter with name 'PARAMETER3'
```

In an attempt to minimize this problem the naming of "un-named" parameters has now been changed to:

```
ORDPPARAM<position index>
```

This has now been fixed.

5.18.11 FetchSize is not Correctly Set for RdbCommands Containing Input or Output Parameters

During the internal preparation of the SQL statement associated with an `RdbCommand` the `RdbConnection` `FetchSize` is copied to the newly prepared statement and it is this value that is used in subsequent executions of the statement to determine the number of records that should be passed between the client and the server process for each network IO.

A problem in how the `FetchSize` value is determined may prevent the correct `FetchSize` being set for the statement, if the `RdbCommand` contains any `RdbParameters`.

For example:

```
// C#
.
.
.
RdbConnection db = new RdbConnection (ConnectionString);
db.Open();
RdbCommand cmd = new RdbCommand(
    "select * from t1 where f1 = :VAL1", db);
db.FetchSize = 1;
.
.
.
```

will NOT set the `FetchSize` correctly and multiple records may still be returned during a single Network IO.

This problem has now been fixed.

5.18.12 Default FetchSize May be Set Incorrectly

If not explicitly set, ORDP is documented to use a default `FetchSize` of 100 for both the `RdbCommand` and `RdbDataReader` classes.

However a problem in how the default value is determined causes ORDP to set the default `FetchSize` to 1 in certain circumstances.

If the `RdbCommand` is instantiated and the `CommandText` property set prior to the associated `RdbConnection` being `Open`, ORDP will default the `RdbCommand FetchSize` to 1.

The following example shows the type of code that will show this problem:

```
// C#
.
.
.
using (conn = new RdbConnection(cs))
{
    RdbCommand cmd = new RdbCommand(
        "SELECT ...", conn);
    cmd.Connection.Open();
.
.
.
```

A workaround for this problem is to `Open` the `RdbConnection` prior to establishing the `RdbCommand` or setting the `RdbCommand CommandText` property:

```
// C#
.
.
.
using (conn = new RdbConnection(cs))
{
    cmd.Connection.Open();
    RdbCommand cmd = new RdbCommand(
        "SELECT ...", conn);
.
.
.
```

An alternative workaround is to set the `RdbCommand FetchSize` explicitly:

```
// C#
.
.
.
using (conn = new RdbConnection(cs))
{
    RdbCommand cmd = new RdbCommand(
        "SELECT ...", conn);
    cmd.FetchSize = 100;
    cmd.Connection.Open();
.
.
.
```

This has now been fixed.

5.18.13 ExecuteNonQuery() Returns Wrong Number of Records Affected

The `RdbCommand.ExecuteNonQuery()` method returns an integer value specifying the number of rows affected by the command.

A problem in how the `RdbDataAdapter` determined the number of records affected during delete operations causes the value returned to be always set to zero.

Although the return value is incorrect, the `DataAdapter` will carry out the underlying deletions correctly.

This problem only affects connections using SQL/Services connectivity.

This has now been fixed.

5.18.14 Concurrency Violation Exception Incorrectly Raised

When the `RdbDataAdapter.Update()` method is called, the contents of the internal cache of records is checked to see if any records need to be inserted, deleted or updated in the database.

During this update process the `DataAdapter` will issue the appropriate SQL statements to make the updates to the underlying database table. During the execution of these updates the `DataAdapter` checks to see the number of records altered, and if the number is not the number expected an exception similar to the following will be raised:

System.Data.DBConcurrencyException: Concurrency violation: the DeleteCommand affected <n1> of the expected <n2> records.

A problem in how the `RdbDataAdapter` determined the number of records affected by the update statement may cause this exception to be raised incorrectly.

The `DataAdapter` will carry out the underlying updates correctly and the database will have the updates correctly applied to it, but after the updates have been applied this exception may be raised.

This problem only affects connections using SQL/Services connectivity.

This has now been fixed.

5.18.15 Overflow/Underflow Exception not Raised on RdbParameter Assignments

The assignment of a value to an `RdbParameter` with an integer `DbType` does not raise an overflow or underflow exception if the value is outside the range allowed for the given `DbType`.

Instead the value is cast to the appropriate datatype that may truncate the value.

For example, the following statement will silently truncate the value specified:

```
// C#
.
.
.
RdbParameter p1 =
    new RdbParameter(":iEMPNO", DbType.Int16, 9999999999, "EMPNO");
.
.
.
```

This has now been changed. Overflow or underflow of integer values in RdbParameter value assignments will now raise an appropriate exception.

5.18.16 Possible Memory Leak Due to RdbDataAdapter.Dispose() Problem

The RdbDataAdapter.Dispose() method does not correctly dispose of all of its children objects which may mean that memory may be slowly used up in long-lasting RdbConnection instances.

This problem affects only the .NET objects used within the RdbDataAdapter and RdbConnection.

All underlying database and SQL/Services resources are correctly deallocated on the Dispose of the RdbDataAdapter or the closing of any DataReader objects used by the RdbDataAdapter.

Oracle advises that RdbDataAdapter objects should be explicitly disposed once they are no longer required instead of relying on the implicit disposal done by the .NET Garbage Collector.

For example:

```
// C#
.
.
.
DataSet ds = new DataSet();
string selectString = "select * from JOBS";
RdbDataAdapter adapter =
    new RdbDataAdapter(selectString, conn);
adapter.Fill(ds);
.
.
.
adapter.Dispose();
.
.
.
```

This has now been fixed.

5.18.17 Memory Access Problem with SetStrVal.

When passing string parameters to the underlying SQL statement, ORDP must convert from the internal .NET string format encoded in UNICODE to the character set specified for the destination

column or variable. During this conversion, a problem in how data is copied between internal buffers may cause the following exception to be raised:

```
Attempted to read or write protected memory
```

and the stack trace will contain reference to the following method:

```
Oracle.DataAccess.RdbClient.Common.SQS.SetStrVal
```

This problem only affects connections using SQL/Services connectivity.

This has now been fixed.

5.18.18 DataAdapter.Fill() May Lose Last Column in Select

The DataAdapter.Fill() method may fail to deliver the last select column to the DataSet when the select statement contains references to multiple tables.

This problem only affects statements that do not return simple, single-table dbkeys. Depending on the SQL statement the simple parser used by the drivers may fail to recognize that the statement cannot return a simple dbkey, in which case the driver will expect that the last column returned by the database server will be the row's dbkey. If this occurs, the last column of the selection will not be added to the resultant DataSet.

For example:

```
// C#
.
.
.
DataSet ds = new DataSet();
string selectString =
@"select first_name, last_name from (employees) inner join
salary_history on employees.employee_id = salary_history.employee_id
where employees.last_name = 'Toliver'";

RdbDataAdapter adapter =
new RdbDataAdapter(selectString, conn);
adapter.Fill(ds);
.
.
.
```

In the above example, the parenthesis around the **employees** table reference prevents the simple parser from determining that dbkeys will not be returned.

Removing the parenthesis will allow the DataSet to be created correctly.

This problem only affects connections using Thin Server connectivity.

This has now been fixed.

5.18.19 DbType.Time Datatype not Handled Correctly by RdbParameter

When a value is set in an RdbParameter, ORDP will try to convert the input value specified to a value and datatype that is compatible with the underlying database object.

If the RdbParameter datatype is `DbType.Time` the conversion is not handled correctly and ORDP will raise the following one of the following exceptions:

```
System.NullReferenceException: Object reference not set to an instance of an object
```

or

```
%RDB-E-ARITH_EXCEPT, truncation of a numeric value at runtime  
-COSI-F-IVTIME, invalid date or time
```

When reading a TIME column from the database, problems in the conversion may also cause the following exception to be raised:

```
System.FormatException: String was not recognized as a valid DateTime
```

This has now been fixed.

5.18.20 Multiple Threads Connecting Concurrently May Cause an Access Violation

When multiple threads within a single process create concurrent RdbConnections, it is possible that an Access Violation may be thrown in the RDBNET.DLL.

Multiple concurrent threads can be handled by ORDP. However, there is a chance that two concurrent threads within a single process attempting to establish connections at exactly the same time may interact and cause one thread to fail with an Access Violation.

This problem only affects connectivity using SQS type connections.

This has now been fixed.

5.18.21 Setting RdbConnection.Autocommit May Cause Null Pointer Exception

Trying to set the RdbConnection.Autocommit property when the connection is not open, will fail with a Null Pointer exception.

This has now been fixed.

5.18.22 Read-Write transactions Incorrectly Started Within ReadOnly Connections

Setting a Connection to ReadOnly indicates to the underlying Data Provider that only read operations should be executed on the database.

In addition, if a connection has been set ReadOnly, ORDP will try to use Read Only transactions on the underlying database where ever possible.

A problem in ORDP prevents it from determining the correct transaction to use within a ReadOnly connection when the following RdbConnection methods are called:

```
BeginTransaction()  
BeginTransaction(IsolationType)  
BeginTransaction(String)
```

This has now been fixed. If the RdbConnection has been set ReadOnly:

```
BeginTransaction() – will now start a READ ONLY transaction  
BeginTransaction(IsolationType) – will raise an exception as READ WRITE transactions  
are not supported within a ReadOnly connection  
BeginTransaction(String) – will raise an exception if the transaction string supplied does  
not start with "read only".
```

5.18.23 Connection.State Incorrect After Execution of DataReader on Stored Procedure

When a DataReader is executed on an RdbCommand with type CommandType.StoredProcedure, the RdbConnection.State is incorrectly left at ConnectionState.Fetching.

Subsequent operations on the connection may result in the following exception:

```
System.InvalidOperationException:  
Operation is not valid due to the current state of the object.
```

Example

```
// C#  
.
```

```

.
.
RdbCommand cmd = new RdbCommand(conn);
RdbTransaction txn = conn.BeginTransaction();

cmd.CommandType = CommandType.StoredProcedure;
cmd.CommandText = "testParam2";
RdbCommandBuilder.DeriveParameters(cmd);
IDataReader reader = cmd.ExecuteReader();
while (reader.Read())
{
    .
    .
    .
}

txn.Rollback() //← exception may be thrown here

```

This has now been fixed.

[▲contents](#)

5.19 New Features for Release 7.3.0.1.0

This section describes new and changed features in Oracle Rdb Data Provider for .NET release 7.3.0.1.0.

5.19.1 RdbFactory and RdbConnectionStringBuilder Classes

New classes have been added to ORDP to provide support for DbProviderFactories.

The new `RdbFactory` and `RdbConnectionStringBuilder` classes allow the .NET developer to utilize generic `DataProvider` classes to access and interact with Oracle Rdb.

Introduced in .NET V2.0, generic data provider classes and methods may be used to access ADO.NET compliant data sources. Using these generic classes improves the development of code that is independent of specific data providers.

```

// C#
.
.
.
string res;
string cs = "Server=MYNODE:MY_SRV;Database=MY_DBS:MF_PERSONNEL";
DbProviderFactory f =
    DbProviderFactories.GetFactory("Oracle.DataAccess.RdbClient");
DbConnection c = f.CreateConnection();
RdbConnectionStringBuilder sb = new RdbConnectionStringBuilder();
sb.ConnectionString = cs;
sb.TryGetValue("server", out res);
Console.WriteLine(" server = " + res);
sb.DataSource = "MY_DBS:personnel";
Console.WriteLine(" con str = " + sb.ConnectionString);

```

```

c.ConnectionString = sb.ConnectionString;
c.Open();
SqlCommand 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();
.
.
.

```

See the Microsoft .NET V2.0 documentation for more details on writing data provider independent code.

5.19.2 Limited TransactionScope Now Available

TransactionScope may now be used with RdbEnlistments. However as ORDP does not currently support the use of distributed transactions, the support for TransactionScope is currently limited to transactions within single connections.

```

// C#
.
.
.
RdbCommand cmd = new RdbCommand(
    "insert into customers values (888,1092,'GEORGE')", conn);
try
{
    using (TransactionScope scope = new TransactionScope())
    {
        //Create an enlistment object
        RdbEnlistment myEnlistment = new RdbEnlistment(conn);
        Guid myGuid = Guid.NewGuid();
        //Enlist on the current transaction with the enlistment object
        Transaction.Current.EnlistDurable(myGuid, myEnlistment,
            EnlistmentOptions.None);

        cmd.ExecuteNonQuery();
        scope.Complete();
    }
}
catch (System.Transactions.TransactionException ex)
{
    Console.WriteLine(ex);
}
.
.
.

```

See the Microsoft .NET V2.0 documentation for more details on TransactionScope.

▲ [contents](#)

5.20 Corrections in Release 7.3.0.1.0.

This section describes software errors corrected in the Oracle Rdb Data Provider for .NET release 7.3.0.1.0.

5.20.1 Input Parameter Marker Values not Correctly Set When Using the SQL/Services Client Connectivity

A problem in how input parameters were passed across to SQL/Services prevented the correct values to be assigned prior to execution of prepared statements.

This problem only occurred when using SQL select statements that contained one or more input parameter markers in conjunction with a connection using a "Type=SQS" connectivity type.

The following is an example of code that may fail.

```
// C#
.
.
.
RdbCommand cmd = conn.CreateCommand(
    "select * from employees where employee_id = :id");
RdbParameter p = cmd.Parameters.Add(":id", DbType.String,5);
p.Value = "00164";
RdbDataReader rdr = cmd.ExecuteReader();
while (rdr.Read())
.
.
.
```

The Read() method would fail to find existing records as the value assigned to the input parameter was not passed through to the Rdb Server correctly.

This has now been fixed.

5.20.2 Using RdbCommand.CommandType Other Than TEXT May Cause SQL Syntax Exceptions

During the processing of RdbCommands, the CommandType of the command was lost preventing ORDP from interpreting the command correctly during execution.

For example, establishing a Stored Procedure type command:

```
// C#
.
.
.
RdbCommand cmd = conn.CreateCommand();
cmd.CommandType = CommandType.StoredProcedure;
```

```
cmd.CommandText = "TEST_PROC_1";  
.  
.  
.
```

may fail with a syntax error, as the CommandType will be redefined internally to text rather than StoredProcedure.

This has now been fixed.

5.20.3 Cultural Information Settings may Cause Invalid Numeric Values to be Sent on SQL/Services RdbConnections

When using SQL/Services connectivity for RdbConnections in conjunction with client Cultural Locales that have numeric representations that differ from standard English usage, a problem in the handling of the conversion of numeric values to their string representation may cause Rdb to receive invalid numeric data.

Clients using Locales that use alternate decimal point representation in the text form of numeric literals, for example the use of the comma character ‘,’ for the decimal point in German Locales, may find problems when floating-point values or Decimal values containing fractional values are used.

This has now been fixed.

5.20.4 BigDecimal and Floating Numeric Problem When Using Thin Server Connectivity

When using Thin Server connectivity for RdbConnections, a problem in how data is transferred between the client and the Thin Server may cause invalid values to be sent to the database server.

During the processing of BigDecimal datatypes that have one or more fractional digits or float or double datatypes, a problem with endian representation caused the values to be misinterpreted on the server side.

This problem could lead to various exceptions being raised on the server such as Arithmetic Overflow or Invalid floating point representations.

This has now been fixed.

5.20.5 DateTime Values Incorrect When Using SQL/Services Connectivity

When using Oracle SQL/Services connectivity for RdbConnections, a problem in how dates are transferred may cause invalid values to be sent to the database server.

During the processing of DateTime datatypes, a problem with the conversion to and from SQL/Services Generalized Date and internal .NET DateTime datatypes caused the values to be misinterpreted on the server side.

This problem could lead to incorrect date/time values being stored in and retrieved from the database.

This has now been fixed.

5.20.6 RdbCommandBuilder Does Not Automatically Open Connection

During the building of the Update, Insert and Delete command from the Select command, the RdbCommandBuilder must connect to the database to establish information about the fields within the specified select statement.

If the associated RdbConnection has not been opened already, the RdbCommandBuilder is meant to open the connection and then close it again once it has retrieved the appropriate select statement metadata.

Unfortunately, if the RdbConnection was not already open, the RdbCommandBuilder failed to open it, causing subsequent operations to fail with:

```
RdbCommand connection is not open
```

A work-around for the problem is to explicitly open the RdbConnection prior to using the RdbCommandBuilder methods.

This has now been fixed.

5.20.7 RdbCommandBuilder May Lose Internal Statements

During the establishment of an RdbCommand, simple parsing of the SQL command text is carried out to establish the list of statements contained in the original CommandText and their associated parameter markers.

Unfortunately this information may be lost during the copying of RdbCommand properties from the original RdbDataAdapter to a newly established RdbDataReader used by the RdbDataAdapter.

This in turn may prevent the RdbCommandBuilder class from correctly establishing the original SQL statements and building associated Insert, Delete and Update commands.

This may show up as the following exception that may be raised during the creation of the auxiliary SQL update statements:

```
RdbCommand CommandText does not contain any SQL statements
```

This has now been fixed.

5.20.8 RdbCommandBuilder Fails to Recognize Unique Columns

When the RdbCommandBuilder class tries to build associated Delete and Update commands from the Select command text, it requires that there exists at least one field in the select list that is either Unique or a Primary Key.

A problem in determining if a column was Unique prevented the RdbCommandBuilder from correctly using Unique columns as search keys.

Even if there are Unique columns in the table, if there are no Primary Keys available, the automatic creation of Delete and Update Commands from a Select command would fail with the following message:

```
RdbCommandBuilder cannot operate on queries with no unique or key columns
```

This has now been fixed.

However, in the case of a JDBC Thin Server connection, this problem will still persist. A future version of the Oracle JDBC for Rdb drivers and server will fix this problem.

5.20.9 RdbCommandBuilder Incorrectly used Blob Columns in Where Clause

During the creation of the "where" clause for Delete or Update commands, the RdbCommandBuilder incorrectly included Blob columns (Oracle Rdb datatype "list of byte varying ") in the where conditions.

This has now been fixed.

5.20.10 Some LATIN1 Characters May Not be Retrieved Correctly From the Database

A problem with the conversion of DEC_MCS characters from Oracle Rdb to internal UNICODE representation for use with .NET caused characters with hexadecimal values greater than 0x7F to be incorrectly translated.

This meant that LATIN1 characters containing special diacritic marks were replaced with a character representing an unprintable UNICODE character when retrieved or displayed within the .NET environment.

This problem is independent of the LOCALE or local language settings.

This has now been fixed.