Oracle® Rdb Data Provider for .NET

Release Notes
V7.3-10
January 2009

Contents

Send Us Your Comments

Oracle Rdb Data Provider for .NET Release Notes, V7.3-10

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

- Did you find any errors?
- Is the information clearly presented?

- Do you need more information? If so, where?
- Are the examples correct? Do you need more examples?
- What features did you like most about this manual?

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

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

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

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

Preface

This document is your primary source of 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
- Organization
- Related Documentation
- Conventions
- Documentation Accessibility

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.

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 7.3-10"**
  Describes new and changed features in Oracle Rdb Data Provider for .NET release 7.3-10.
- **Chapter 3, "Problems Corrected"**
  Describes problems corrected in Oracle Rdb Data Provider for .NET release 7.3-10.
- **Chapter 4, "Known Problems, Restrictions and Workarounds"**
  Describes known problems, restrictions, and workarounds for Oracle Rdb Data Provider for .NET release 7.3-10.
- **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 7.2 SQL Reference Manual*
- *Oracle Rdb Guide to SQL Programming*
- *Oracle SQL/Services Server Configuration Guide*
- *Guide to Using the Oracle Rdb7 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/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';
```

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	V2.0
Windows NT, Windows XP, Windows Vista, Windows 2000, or Windows Server 2003	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	V7.1.1

Note:
The installation of ORDP 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.

On the SERVER:

Software	Minimum Version	
	Alpha	Integrity
HP OpenVMS	V7.1	V8.2-1
Oracle SQL/Services	V7.1.5	V7.2
TCP/IP Services for OpenVMS	V3.3	V5.5-11
Oracle Rdb	V7.1-24	V7.2

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	V7.3-2	V8.2-1
Oracle JDBC for Rdb	V7.2.5.0	V7.2.5.0
HP Java [tm] SDK/RTE	V1.4.1	V1.4.2-1
Oracle Rdb	V7.1-24	V7.2

See Also:

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

1.2 Installing Oracle Rdb Data Provider for .NET

NOTE: **If you have previously installed a beta version of the Oracle Rdb Data Provider for .NET you must de-install this beta version prior to installation of this version.**

Two variants of the installation files and procedures are available:

1. Normal installation
2. Strongly Named installation

The strongly named installation will place the ORDP assembly in your system Global Assembly Cache so that the single assembly instance may be used by multiple applications.

Please refer to your Microsoft .NET documentation for information on strongly named assemblies and the Global Assembly Cache.

The normal installation will place the ORDP assembly in the installation directory you choose during the installation.

1.2.1 Normal Installation

For normal installation the Oracle Rdb Data Provider for .NET comes packaged in the Microsoft installation file:

```
OracleRdbDataProvider73100.msi
```

The installation may be carried out by invoking the MSI file provided directly or by running the setup executable file that is also provided in the same installation kit:

```
setup_ORDP.exe
```

These are standard Microsoft installation file and setup executables that will install Oracle Rdb Data Provider for .NET when it is invoked. Just follow the steps displayed during the installation process.

1.2.2 Strongly Named Installation

For a strongly named installation the Oracle Rdb Data Provider for .NET comes packaged in the Microsoft installation file:

```
OracleRdbDataProvider73100SN.msi
```

The installation may be carried out by invoking the MSI file provided directly or by running the setup executable file that is also provided in the same installation kit:

```
setup_ORDP_SN.exe
```

These are standard Microsoft installation file and setup executables that will install Oracle Rdb Data Provider for .NET when it is invoked. Just follow the steps displayed during the installation process.

During a strongly named installation the ORDP assembly file Oracle.DataAccess.Rdb.dll will be installed into your system Global Assembly Cache.

1.2.2.1 Assembly Version Change for V7.3-10

The assembly version and public key has changed for the ORDP V7.3-10 Strongly Named assembly.

References made to prior version ORDP strongly names assemblies in configuration files 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 from the System Assembly Cache.

To uses the `RdbFactory` class 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 your .NET Framework install directory, for example,

on Windows 2000 the directory would be:

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

on Window XP the directory would be:

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

The following must be added to the <DbProviderFactories> section of your system's 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=7.3.1.0,
Culture=neutral,PublicKeyToken=24caf6849861f483" />
.
.
.
```

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

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

.
.
.
  </configSections>
</configuration>
```

1.3 File Locations

The installation files will be placed in the directory chosen during the installation procedure. The default directory is:

```
<systemdisk>\Program Files\Oracle\ORDP\
```

All the required DLL files and documentation will be copied to the installation directory:

File	Description
ORDP_dev_guide.(html/pdf)	ORDP Developer Guide
ORDP_rel_notes.(html/pdf)	ORDP Release Notes

File	Description
Oracle.DataAccess.Rdb.dll	the ORDP assembly
rdbnet.dll	required library for Rdb access
sql_cosi.dll	required library for internal Rdbnet operations
sqsapi32.dll	required library for SQL/Services access

In addition, if a strongly named installation was carried out, the ORDP assembly file Oracle.DataAccess.Rdb.dll will also be copied to your system Global Assembly Cache.

1.4 Post Installation Procedures

After installation, prior to using Oracle Rdb Data Provider for .NET, you may have to update your PATH environment variable to include the directory into which the kit was installed.

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.

Chapter 2
Enhancements Provided in Oracle Rdb Data Provider for .NET Release 7.3-10

This chapter describes new and changed features in Oracle Rdb Data Provider for .NET release 7.3-10.

2.1 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 connections pools.

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

2.2 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:

```
RdbCommand cmd = cn.CreateCommand();
cmd.RdbCommandType =
   RdbCommandTypes.ExternalProcedure;
cmd.CommandText = "MY_EXTERNAL_PROCEDURE";
RdbCommandBuilder.DeriveParameters(cmd);
RdbParameterCollection col1 = cmd.Parameters;
```

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

2.3 GetSchema Collections Support

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

Chapter 3
Problems Corrected

This chapter describes problems corrected in Oracle Rdb Data Provider for .NET release 7.3-10.

3.1 RdbFactory Documentation Omission

RdbFactory was introduced in the ORDP release 7.3-01 but information about enabling the use of the RdbFactory code within .NET was omitted from the release notes for ORDP release 7.3-01.

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 your .NET Framework install directory, for example on Windows 2000 the directory would be:

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

If you wish to use the RdbFactory facility in conjunction with ORDP V7.3-01 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 V7.3-01, to use ORDP V7.3-10 see Assembly Version Change for V7.3-10

3.2 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.

Fixed

build 20080408.

3.3 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***.

Fixed

build 20080411.

3.4 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 method may cause ORDP to raise an *System.InvalidCastException:*

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

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

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

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

Fixed

build 20080411.

3.5 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

Fixed

build 20080421.

3.6 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:

```
RdbCommand = 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.
```

Fixed

build 20080429.

3.7 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 transaction may be found to be committed prematurely due to this AUTOCOMMIT action.

Fixed

build 20080429.

3.8 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.

Fixed

build 20080527.

3.9 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 in the firing of the event notification during the close of the `RdbConnection` prevents the ***ConnectionState.Closed*** event from being sent to the event handlers.

Fixed

build 20080527.

3.10 Un-named Parameters Names may Clash with User Defined Parameter Names.

During the parsing of a SQL statement all parameter 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>

Fixed

build 20080527.

3.11 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:

```
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.

Fixed

build 20080602.

3.12 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:

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

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

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

An alternative word-around is to set the `RdbCommand FetchSize` explicitly .

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

Fixed

build 20080527.

3.13 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 DataAdapater will carry out the underlying deletions correctly.

This problem only affects connections using SQL/Services connectivity.

Fixed

build 20080612.

3.14 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 DataAdapater 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.

Fixed

build 20080612.

3.15 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 which may truncate the value.

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

```
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.

Fixed

build 20080612.

3.16 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:

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

Fixed

build 20080626.

3.17 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.

Fixed

build 20080707.

3.18 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:

```
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.

Fixed

build 20080731.

3.19 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
```

Fixed

build 20080807.

3.20 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.

Fixed

build 20080829.

3.21 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.

Fixed

build 20081202.

3.22 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".

Fixed

build 20081202.

3.23 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

```
.
.
.
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
```

Fixed

build 20090102.

Chapter 4
Known Problems, Restrictions and Workarounds

This chapter describes known problems, restrictions, and workarounds for Oracle Rdb Data Provider for .NET release 7.3-10.

4.1 Distributed transaction support not available in this version.

The use of distributed transactions is not currently supported in ORDP. This feature will be available in a future release of this product.

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

Oracle Rdb Data Provider for .NET is not currently integrated into the Microsoft Visual Studio for .NET IDE. This feature will be available in a future release of this product.

4.3 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.

```
.
.
.
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's 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:

```
.
.
.
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())
.
.
.
```

4.4 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.

```
.
.
.
// 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));
}
.
.
.
```

Chapter 5
New Features and Corrections in Previous Releases

5.1 New Features for Release 7.3.0.1

This section describes new and changed features in Oracle Rdb Data Provider for .NET release 7.3-01

5.1.1 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.

```
.
.
.
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();
DbCommand cmd = c.CreateCommand();

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

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

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

5.1.2 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.

```
.
.
.
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.

5.2 Corrections in Release 7.3.0.1

This section describes software errors corrected in the Oracle Rdb Data Provider for .NET release 7.3-01.

5.2.1 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.

```
.
.
.
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 be passed through to the Rdb Server correctly.

Fixed

build 20071023.

5.2.2 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:

```
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.

Fixed

build 20071110.

5.2.3 Cultural Information Settings may Cause Invalid numeric values to be Sent on SQL/Services service-based 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.

Fixed

build 20071204.

5.2.4 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.

Fixed

build 20071204.

5.2.5 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 SQS/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.

Fixed

build 20080208.

5.2.6 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.

Fixed

build 20080219.

5.2.7 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
```

Fixed

build 20080219.

5.2.8 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.

Fixed

build 20080219.

5.2.9 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.

Fixed

build 20080219.

5.2.10 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.

Fixed

build 20080305.