Oracle® Database

Extensions for .NET Developer's Guide

11*g* Release 2 (11.2) for Microsoft Windows

E17724-02

August 2013

ORACLE®

Oracle Database Extensions for .NET Developer's Guide, 11g Release 2 (11.2) for Microsoft Windows

E17724-02

Primary Author: Janis Greenberg

Contributors: Kiminari Akiyama, Neeraj Gupta, Shailendra Jain, Chithra Ramamurthy, Gnanaprakash Rathinam, Christian Shay, Subramanian Venkatraman

Contents

3   Development and Deployment with Visual Studio

4   Development and Deployment of a .NET Stored Function Demo

A   Data Type Conversion

B   Troubleshooting Common Errors

Glossary

Index

Preface

This document describes the features of Oracle Database for Windows software installed on Windows 2003, Windows 2000, and Windows XP Professional operating systems.

This guide describes Oracle Database Extensions for .NET, which provides a Common Language Runtime (CLR) host for Oracle Database and data access through Oracle Data Provider for .NET (ODP.NET) classes.

This preface contains these topics:

- Audience
- Documentation Accessibility
- Related Documents
- Passwords in Code Examples
- Conventions

Audience

Oracle Database Extensions for .NET Developer's Guide is intended for programmers who are developing applications to access an Oracle Database using Oracle Database Extensions 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 another .NET language.

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

Documentation Accessibility

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

Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit
`http://www.oracle.com/pls/topic/lookup?ctx=accid=info` or visit

if you are hearing impaired.

Related Documents

For more information, see these Oracle resources:

- *Oracle Developer Tools for Visual Studio Help*

- *Oracle Data Provider for .NET Developer's Guide*

- *Oracle Database PL/SQL Packages and Types Reference*

- *Oracle Database SQL Reference*

- *Oracle Database Installation Guide for Windows*

- *Oracle Database Release Notes for Windows*

- *Oracle Database Platform Guide for Windows*

- *Oracle Database New Features*

- *Oracle Net Services Reference Guide*

Many of the examples in this book use the sample schemas, which are installed by default when you select the Basic Installation option with an Oracle Database installation. Refer to *Oracle Database Sample Schemas* for information on how these schemas were created and how you can use them yourself.

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

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

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

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

Passwords in Code Examples

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

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.

Convention	Meaning
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

What's New in Oracle Database Extensions for .NET

This section describes new features in Oracle Database Extensions for .NET and provides references to additional information.

The following section describes the new features in Oracle Database Extensions for .NET.

-
-

New Features in Oracle Database Extensions for .NET Release 11.2.0.1.2

Oracle Database Extensions for .NET release 11.2.0.1.2 includes the following:

- .NET Framework 4 support

  Oracle Database Extensions for .NET Framework 4 supports .NET Framework 4.

New Features in Oracle Database Extensions for .NET Release 11.1.0.6.20

Oracle Database Extensions for .NET release 11.1.0.6.20 includes the following:

- Support for Unloading .NET Assemblies

  You may unload .NET assemblies after .NET stored procedure execution is complete. This feature enables you to redeploy and to test code without having to restart `extproc`.

  > **See Also:** "Unloading .NET Assemblies for Easy Redeployment" on page 2-8

- Support for Nullable Types

  Oracle Database Extensions for .NET now supports nullable types as procedure or function parameters.

  > **See Also:** "Backward Compatibility for Nullable ODP.NET Connected Types" on page 2-9

x

1

Introduction to Oracle Database Extensions for .NET

This chapter introduces Oracle Database Extensions for .NET, which makes it possible to build and run .NET stored procedures or functions with Oracle Database for Microsoft Windows.

This chapter contains these topics:

- Oracle Database Extensions for .NET Overview
- Oracle Database Extensions for .NET Architecture
- Oracle Data Provider for .NET
- Oracle Developer Tools for Visual Studio
- Oracle Deployment Wizard for .NET

Oracle Database Extensions for .NET Overview

Oracle Database Extensions for .NET provides the following:

- A Common Language Runtime (CLR) host for Oracle Database
- Data access through Oracle Data Provider for .NET classes
- Oracle Deployment Wizard for Visual Studio

The Oracle Database hosts the Microsoft Common Language Runtime (CLR) in an external process, outside of the Oracle database process. The integration of Oracle Database with the Microsoft Common Language Runtime (CLR) enables applications to run .NET stored procedures or functions on Oracle Database, on Microsoft Windows Vista, Windows 2003, Windows 2000, and Windows XP.

Application developers can write stored procedures and functions using any .NET compliant language, such as C# and VB.NET, and use these .NET stored procedures in the database, in the same manner as other PL/SQL or Java stored procedures. .NET stored procedures can be called from PL/SQL packages, procedures, functions, and triggers; from SQL statements, or from anywhere a PL/SQL procedure or function can be called.

Application developers build .NET procedures or functions into a .NET assembly, typically using Microsoft Visual Studio. Oracle Data Provider for .NET is used in .NET stored procedures and functions for data access. After building .NET procedures and functions into a .NET assembly, developers deploy them in Oracle Database, using the Oracle Deployment Wizard for .NET, a component of the Oracle Developer Tools for Visual Studio.

The .NET stored procedure or function appears to the caller as a PL/SQL stored procedure or function because a PL/SQL wrapper has been generated for it. The user invokes a .NET stored procedure or function through this PL/SQL wrapper. Oracle Deployment Wizard for .NET determines the probable mappings between Oracle data types and .NET data types, which the user can override. The mappings are handled seamlessly by the PL/SQL wrapper.

Oracle Database Extensions for .NET Architecture
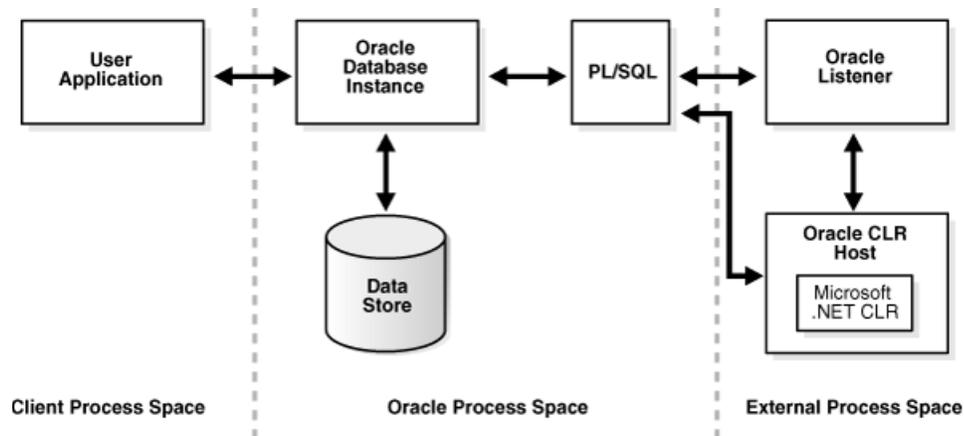
This architecture diagram shows the client application and then two process spaces, the Oracle process space and the external process space.

The Oracle process space includes the Oracle database instance and hosts the PL/SQL wrapper.

The external process space includes the Oracle CLR host, in which .NET stored procedures or functions are executed.

Oracle CLR Host

The Oracle CLR host is installed as part of Oracle Database Extensions for .NET installation and runs in the `extproc` process. The `extproc` process loads the Oracle CLR host which in turn loads an instance of the Microsoft Common Language Runtime (CLR), thus providing an interface for the wrapped PL/SQL procedure. These mechanics are not visible to the users. From a user's point of view, the application is invoking just another PL/SQL stored procedure or function.

> **Note:** The Microsoft .NET Framework must be installed on the same computer as the database.

External Processes

.NET stored procedures or functions are hosted in a process external to the Oracle Database. This external process is a heterogeneous service agent called `extproc`, external procedure agent, or external process. This guide uses the terms `extproc` process or `extproc` agent.

> **See Also:**
>
> *Oracle Database Heterogeneous Connectivity Administrator's Guide*

The `extproc` process supports the following architectures:

- Dedicated Agent Architecture
- Multithreaded Agent Architecture

Dedicated Agent Architecture

In dedicated (that is, single-threaded) agent architecture, an `extproc` process is started for each user session. The process terminates when the user session ends. This architecture can consume an unnecessarily large amount of system resources since, with every user session, a new `extproc` process must be started and shut down. Therefore dedicated agent architecture does not perform well in terms of system resources and runtime efficiency.

Multithreaded Agent Architecture

A multithreaded `extproc` process uses a pool of shared threads. The tasks requested by the user sessions are put on a queue and are picked up by the first available thread.

Multithreaded agent architecture allows more efficient use of system resources than dedicated architecture.

A separate multithreaded `extproc` process must be started for each system identifier (SID). Each TNS listener that is running on a system listens for incoming connection requests for a set of SIDs. If the SID in an incoming Oracle Net connect string is one that the listener is listening for, then that listener processes the connection. If a multithreaded process has been started for the SID, then the listener passes the request to that process.

> **See Also:**
>
> - *Oracle Database Application Developer's Guide - Fundamentals*
> - *Oracle Database Administrator's Guide*

Real Application Clusters (RAC) and External Processes The Oracle multithreaded `extproc` process is tightly coupled with the Oracle listener. Therefore, each node in a Real Application Clusters (RAC) environment has an Oracle multithreaded `extproc` process associated with the listener on that node.

Multiple Databases Instance and External Processes A single Oracle multithreaded `extproc` process is used with multiple database instances associated with a single Oracle home if a single listener is shared among multiple database instances.

Listener and Tnsnames Files

The following are typical examples of the `listener.ora` and `tnsnames.ora` files configured for Oracle Database Extensions for .NET. By default, Oracle Database Extensions for .NET uses `CLRExtProc` as the SID, but this can be changed using the Database Configuration Assistant (DBCA).

Listener.ora file

```
SID_LIST_LISTENER =
  (SID_LIST =
    (SID_DESC =
      (SID_NAME = PLSExtProc)
      (ORACLE_HOME = C:\oracle\database_1)
      (PROGRAM = extproc)
    )
```

```
            (SID_DESC =
              (SID_NAME = CLRExtProc)
              (ORACLE_HOME = C:\oracle\database_1)
              (PROGRAM = extproc)
              (ENVS="EXTPROC_DLLS=ONLY:C:\oracle\database_1\bin\oraclr11.dll")
            )
        )

    LISTENER =
      (DESCRIPTION_LIST =
        (DESCRIPTION =
          (ADDRESS = (PROTOCOL = IPC)(KEY = EXTPROC1))

        )
      )
```

Tnsnames.ora File

```
ORACLE =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP)(HOST = user.us.oracle.com)(PORT = 1521))
    )
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = oracle.us.oracle.com)
    )
  )

ORACLR_CONNECTION_DATA =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = IPC)(KEY = EXTPROC1))
    )
    (CONNECT_DATA =
      (SID = CLRExtProc)
      (PRESENTATION = RO)
    )
  )

MSOLNIT-PC =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP)(HOST = msolnit-pc)(PORT = 1521))
    )
    (CONNECT_DATA =
      (SERVICE_NAME = orcl)
    )
  )

EXTPROC_CONNECTION_DATA =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = IPC)(KEY = EXTPROC1))
    )
    (CONNECT_DATA =
      (SID = PLSExtProc)
      (PRESENTATION = RO)
    )
  )
```

> **See Also:** "Configuring Extproc Agent Using Windows Service" on page 2-3

Oracle Data Provider for .NET

Oracle Data Provider for .NET provides data access to the Oracle Database from any client application. Oracle Data Provider for .NET is available for free download on Oracle Technology Network (OTN).

> **See Also:** *Oracle Data Provider for .NET Developer's Guide* for detailed descriptions of ODP.NET classes

Oracle Developer Tools for Visual Studio

Oracle Developer Tools for Visual Studio is a set of application tools tightly integrated with the Visual Studio development environment. Oracle Developer Tools enables developers to execute a wide range of application development tasks, such as creating tables, editing stored procedures, and viewing data in the Oracle Database. Oracle Developer Tools for Visual Studio is available for free download on Oracle Technology Network (OTN).

> **See Also:** *Oracle Developer Tools for Visual Studio Help*

Oracle Deployment Wizard for .NET
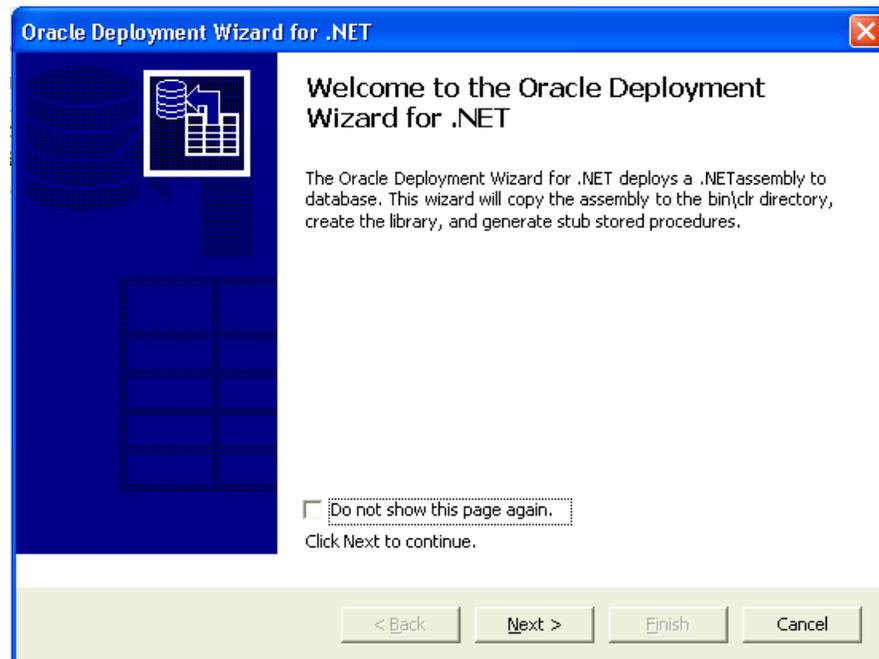
The Oracle Deployment Wizard for .NET is a graphical tool integrated with Microsoft Visual Studio which makes it easy to deploy any .NET procedure or function into an Oracle Database. It is installed as part of Oracle Developer Tools for Visual Studio.

> **See Also:** *Oracle Developer Tools for Visual Studio Dynamic Help*, available by installing Oracle Developer Tools for Visual Studio, for more information

2

Installation and Configuration

This chapter describes the installation of Oracle Database Extensions for .NET, system requirements, and file locations.

This chapter contains these topics:

- System Requirements
- Requirements for .NET Stored Procedures and Functions
- File Locations after Installation
- Installation and First Use
- Configuring Extproc Agent Using Windows Service
- Migrating .NET Stored Procedures from Oracle Database 10.2 to Oracle Database 11.2
- Mandatory Migration of .NET 1.x Stored Procedures to .NET 2.0
- Oracle Database Extensions for .NET Registry Options
- Unhandled Exceptions

System Requirements

Each release of Oracle Database Extensions for .NET has very specific version requirements. The following system requirements only apply to 32-bit Oracle Database Extensions for .NET version 11.2.0.1.0. If you are using a different version, please see the documentation specific to your version:

- Oracle Database 11g version 11.2.0.1.0 on 32-bit Windows.

  > **Note:** Oracle Database Extensions for .NET is only supported on the Windows Platform.

- Microsoft .NET Framework

  - ODE.NET for .NET Framework 2.0 is only supported with Microsoft .NET Framework 2.0, 3.0, 3.5, 4, and .NET Framework 4 Client Profile
  - ODE.NET for .NET Framework 4 is only supported with Microsoft .NET Framework 4 and .NET Framework 4 Client Profile

> **Note:** Microsoft Framework 1.x is no longer supported as of Oracle Database Extensions for .NET version 11.1.0.7.20. If you have stored procedures that require .NET Framework 1.x, you will need to take some special steps to make them work with this release. For more information see "Mandatory Migration of .NET 1.x Stored Procedures to .NET 2.0" on page 2-5.

- Oracle Data Provider for .NET version 11.2.0.1.0 or higher (if data access in stored procedures is required).

- Oracle Developer Tools for Visual Studio 10.2 or higher is required for .NET stored procedure deployment.

> **Note:** Oracle Developer Tools for Visual Studio is not released with Oracle Database. It can be obtained from the Oracle .NET Developer Center at OTN.

Requirements for .NET Stored Procedures and Functions

A .NET stored procedure or function must meet the following requirements:

- Be declared a public static method.

- Not be a constructor or a destructor.

- Use parameter types that are compatible with the Oracle native database types.

File Locations after Installation

`OraClr11.dll` is installed in the *ORACLE_BASE\ORACLE_HOME*\bin directory.

`Oracle.Database.Extensions.dll` is installed to the following locations:

- .NET Framework 2.0:

  ORACLE_BASE\ORACLE_HOME\ODE.NET\bin\2.*x*

- .NET Framework 4:

  ORACLE_BASE\ORACLE_HOME\ODE.NET\bin\4

The readme file, `readme.html`, is installed in the *ORACLE_BASE\ORACLE_HOME*\ODE.NET\DOC directory.

.NET assemblies deployed by developers are copied into the *ORACLE_BASE\ORACLE_HOME*\bin\CLR directory (or its subdirectory) by the Oracle Deployment Wizard for .NET.

Installation and First Use

Oracle Database Extensions for .NET is installed as part of a custom Oracle Database installation using the Oracle Universal Installer. From the Oracle Universal Installer Welcome screen, you must chose **Advanced Installation**, then from the Select Installation Type screen, choose **Custom**. When the Available Product Components list

is displayed, go to Enterprise Edition Options, and select **Oracle Database Extensions for .NET**.

Oracle Database Extensions for .NET is configured using the Database Configuration Assistant.

Configuring Extproc Agent Using Windows Service

As part of Oracle Database Extensions for .NET installation, a Windows service is installed. The service is called `OraClrAgnt` and can be accessed through the Service Control Panel, as `OracleORACLE_HOMEClrAgent`, where *ORACLE_HOME* represents your Oracle home.

This service is used for the startup, configuration, and shutdown of the `extproc` agent.

The parameters which can be configured using this service are listed in Table 2–1, " OraClrAgnt Service Parameters".

These parameter values can be specified as part of the Start Parameters in the properties window of the Control Panel Service. In this case, the parameter values are not saved and the values must be supplied again if the service is restarted later.

To persist the parameter values, you can change the Windows registry entry for this service and provide the parameter values as command line parameters to `OraClrAgnt.exe`. To do this, set the Windows registry key, `ImagePath`, located at

`HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\`
`OracleOracleHomeClrAgent`

The value should be something similar to the following:

`ORACLE_BASE\ORACLE_HOME\bin\OraClrAgnt.exe agent_sid=CLRExtProc`
`max_dispatchers=2 tcp_dispatchers=0 max_task_threads=6 max_`
`sessions=25 ENVS="EXTPROC_DLLS=ONLY:ORACLE_BASE\ORACLE_`
`HOME\bin\oraclr11.dll"`

If the service cannot be started or stopped, the error messages are logged in the Application Log of the Event Viewer, with the service name as the event source name.

OraClrAgnt Service Parameters

Table 2–1 lists the parameters which can be configured using this service.

Table 2–1   OraClrAgnt Service Parameters

Parameters	Descriptions
`agent_sid`	This represents the `SID` of the `extproc` process. The default value is `CLRExtProc`. This is a mandatory parameter. If this parameter value is changed, appropriate changes need to be made in `tnsnames.ora` and `listener.ora` files.
`ENVS`	Variable that specifies the `EXTPROC_DLLS` environment variable, which restricts the DLLs that `extproc` can load.
	This is similar to setting environment variables to external procedures using `listener.ora`.
	Refer to "Table 13–5 External Procedures Settings in listener.ora" in *Oracle Net Services Administrator's Guide* for more information.

Table 2–1   (Cont.) OraClrAgnt Service Parameters

Parameters	Descriptions
listener_address	Address on which the listener is listening. This is an optional parameter. If it is not specified, then this is set to the default value.
max_dispatchers	Number of maximum dispatchers in the extproc process. This is an optional parameter. If it is not specified, then this number is set to a default value.
max_sessions	Number of maximum sessions in the extproc process. This is an optional parameter. If it is not specified, then this number is set to a default value.
max_task_threads	Number of maximum task threads in the extproc process. This is an optional parameter. If it is not specified, then this number is set to a default value.
shutdown_address	Address on which the agent should listen for shutdown messages from agtctl. This is an optional parameter. If it is not specified, then this is set to the default value.
tcp_dispatchers	Number of TCP dispatchers in the extproc process. This is an optional parameter. If it is not specified, then this number is set to a default value.

Note: By default, this service is created and run in the Local System Account; however, it can be changed to work with a logged-on user account through the service control panel.

See Also: *Oracle Database Application Developer's Guide - Fundamentals,* Table A-2, Configuration Parameters for agtctl

Tuning OraClrAgnt for Performance

You should tune the OraClrAgnt to match the expected load on your system.

Excessive extproc.exe processes being spawned is a sign that you have set the configuration values too low.

Start with the following values and increase as you test your system for performance:

OraClrAgnt Parameter	Initial Value
max_sessions	25
max_task_threads	6
max_dispatchers	2

Migrating .NET Stored Procedures from Oracle Database 10.2 to Oracle Database 11.2

You can migrate .NET stored procedures from Oracle Database release 10.2 to release 11.2 as follows:

1. Select the libraries that are used by .NET stored procedures from the Oracle Database 10*g* release 2 (10.2) database. For example,

```
SELECT library_name, file_spec FROM ALL_LIBRARIES WHERE
OWNER='SYS' and FILE_SPEC LIKE '$ORACLE_HOME\bin\clr\%';
```

`library_name` is usually in the format `dll_name_DLL`. For example, the `library_name` for `Project1.dll` would be `PROJECT1_DLL`.

2. Create a SQL file manually (for example, `DotNetSP_Grant.sql`) with the following SQL statements:

```
CREATE LIBRARY "SYS"."library_name" AS 'file_spec'
GRANT EXECUTE ON "SYS"."library_name" TO "schema_name"
GRANT EXECUTE ON "SYS"."DBMS_CLR" TO "schema_name"
GRANT EXECUTE ON "SYS"."DBMS_CLRTYPE" TO "schema_name"
GRANT EXECUTE ON "SYS"."DBMS_CLRPARAMTABLE" TO "schema_name"
```

3. Run Oracle Data Pump Export utility for the Oracle Database 10*g* release 2 (10.2).

```
Expdp system schemas="schema_name" directory=ORACLECLRDIR dumpfile=DotNetSP.dmp
include=PROCEDURE,FUNCTION
```

4. Copy .NET stored procedure assemblies from Oracle Database 10*g* release 2 (10.2) *ORACLE_BASE\ORACLE_HOME*\bin\clr folder and its subfolders to the same directory structure in Oracle Database 11*g* release 2 (11.2).

5. Run `DotNetSP_Grant.sql` as `SYSDBA` against the Oracle Database 11*g* release 2 (11.2) database.

6. Run Oracle Data Pump Import utility for the Oracle Database 11*g* release 2 (11.2) database.

```
impdp system schemas="schema_name" directory=ORACLECLRDIR dumpfile=DotNetSP.dmp
```

Mandatory Migration of .NET 1.x Stored Procedures to .NET 2.0

Beginning with Oracle Database Extensions for .NET version 11.1.0.7.20, .NET 1.x stored procedures are no longer supported. Specifically, Oracle Database Extensions for .NET 1.x and Oracle Data Provider for .NET 1.x are no longer included in this release. If you have existing .NET 1.x stored procedures from an earlier release, you will need to take special migration steps to ensure that they work in this release.

> **WARNING:**   In some cases, this migration will require code changes. You should not install this release in a production environment if you have .NET 1.x stored procedures until you have verified in a test environment that your stored procedures have been successfully migrated. If you have already installed this release and are encountering errors in your .NET 1.x stored procedures, you should downgrade to an earlier version of Oracle Database Extensions for .NET until you are able to make any required code changes to your stored procedures.

Determining if Code Changes are Needed

You will need to analyze your .NET 1.x stored procedures to determine if code changes are required to migrate to this release. Specifically you should investigate:

- Code incompatibilities between ODP.NET for .NET 1.x and ODP.NET for .NET 2.0.

See "Addressing Code Incompatibilities Between ODP.NET for .NET 1.x and ODP.NET for .NET 2.0" on page 2-6.

- Code incompatibilities between Oracle Database Extensions for .NET 1.x and Oracle Database Extensions for .NET 2.0.

  See "Addressing Code Incompatibilities Between Oracle Database Extensions for .NET Versions 1.x and 2.0" on page 2-7

- ADO.NET 1.x and ADO.NET 2.0 migration issues

  See Microsoft documentation for more details:

  http://msdn.microsoft.com/en-us/netframework/aa570326.aspx

Migration Approaches

There are two possible approaches to allow your .NET 1.x stored procedures to work with this release:

- Recompile and redeploy your .NET 1.x stored procedures using ODP.NET for .NET 2.0. Oracle strongly recommends this approach and it is required if there are incompatibilities that require code changes. See "Recompile and Redeploy .NET 1.x Stored Procedures Using ODP.NET for .NET 2.0" on page 2-7.

- Configure your .NET 1.x stored procedures to run using ODP.NET for .NET 2.0. This does not require recompilation but introduces the possibility of run-time errors if there are unaddressed incompatibilities. See "Configure .NET 1.x Stored Procedures Using ODP.NET for .NET 2.0" on page 2-7.

Addressing Code Incompatibilities Between ODP.NET for .NET 1.x and ODP.NET for .NET 2.0

You may need to address the following code incompatibilities related to ODP.NET in your .NET 1.x stored procedures:

- In ODP.NET for .NET 2.0, `OracleParameter.Value` returns `OracleDecimal` instead of .NET native types when `OracleParameter.OracleDbType` is set to a number type, such as `Int32`, `Double`. This behavior change is summarized in the following table:

OracleParameter.OracleDbType	OracleParameter.Value Returned in .NET 1.x	OracleParameter.Value Returned in .NET 2.x
OracleDbType.Byte	System.Byte	OracleDecimal
OracleDbType.Double	System.Double	OracleDecimal
OracleDbType.BinaryDouble	System.Double	OracleDecimal
OracleDbType.Int16	System.Int16	OracleDecimal
OracleDbType.Int32	System.Int32	OracleDecimal
OracleDbType.Int64	System.Int64	OracleDecimal
OracleDbType.Single	System.Single	OracleDecimal
OracleDbType.BinaryFloat	System.Single	OracleDecimal

If any of the preceding `OracleDbType` enumeration values are used by your .NET 1.x stored procedure for an `out` or `in/out` `OracleParameter`, then it may need to be modified.

- In ODP.NET for .NET 2.0, `OracleParameter.Value` returns provider-type specific `null` value (`OracleClob.Null`) instead of `DBNull.Value` when `OracleParameter.OracleDbType` is set for connected types. For example, if `OracleParameter.OracleDbType` is set to `OracleDbType.Clob`, then `OracleParameter.Value` represents a `null` value by returning `OracleClob.Null` instead of `DBNull.Value`, which is the case in ODP.NET for .NET 1.x.

Addressing Code Incompatibilities Between Oracle Database Extensions for .NET Versions 1.x and 2.0

If any of the connected types are passed as parameters to your .NET 1.x stored procedure, and if the procedure checks for `null` values, then you may need to modify the procedure. Oracle Database Extensions for .NET 1.x uses a .NET `null` to represent a `null` value when the parameter happens to be a connected type. Oracle Database Extensions for .NET 2.0 uses provider-type specific `null` value, such as `OracleBFile.Null`, in these cases.

You can configure Oracle Database Extensions for .NET 2.0 to use .NET `null` values for connected type `null` values in place of provider-specific type `null` values. To do this, create and set the following registry value to 0:

```
HKEY_LOCAL_MACHINE\SOFTWARE\ORACLE\KEY_ORACLE_
HOME\ODE\ProviderNull
```

You can find the documentation about this registry value in the section "Backward Compatibility for Nullable ODP.NET Connected Types" in *Oracle Data Provider for .NET Developer's Guide*.

Recompile and Redeploy .NET 1.x Stored Procedures Using ODP.NET for .NET 2.0

If you find code incompatibilities, you will need to recompile your .NET 1.x stored procedures using ODP.NET for .NET 2.0.

Even if you do not find code incompatibilities, Oracle recommends that you recompile and redeploy your .NET 1.x stored procedures. Recompiling ensures that you do not get run-time errors, if there are any unaddressed compatibility issues. Redeploy the stored procedures after successful recompilation.

Configure .NET 1.x Stored Procedures Using ODP.NET for .NET 2.0

Alternatively, if there are no code incompatibilities, you can configure the .NET 1.x stored procedures to run with ODP.NET for .NET 2.0. This approach does not require you to recompile and redeploy the .NET stored procedures. However, you might get run-time errors, if there are any unaddressed incompatibilities between versions 1.x and 2.0 of Oracle Database Extensions for .NET, ODP.NET, and ADO.NET. Use the following steps to configure the .NET 1.x stored procedures to run with ODP.NET for .NET 2.0 without recompiling and redeploying them:

1. If the .NET 1.x stored procedures exist in an old Oracle home, then copy the .NET 1.x stored procedure DLLs to the new Oracle home, under *NewOracleHome*\bin\clr.

   If the stored procedures were originally deployed in a sub-folder of the previous Oracle home, such as *OldOracleHome*\bin\clr\Accounts, then create a similar directory structure under the new Oracle Home, such as *NewOracleHome*\bin\clr\Accounts, and copy the .NET 1.x stored procedure DLLs into it.

2. Modify or create the extproc.exe.config file in the *NewOracleHome*\bin folder to redirect ODP.NET (Oracle.DataAccess.dll) 1.x references to the installed version of ODP.NET for .NET 2.0. For example, to redirect ODP.NET 1.111.6.20 references to ODP.NET 2.111.7.20, the extproc.exe.config file should include the following configuration section:

```
<configuration>
  <runtime>
    <legacyUnhandledExceptionPolicy enabled="1"/>
    <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
      <dependentAssembly>
        <assemblyIdentity name="Oracle.DataAccess"
                          publicKeyToken="89b483f429c47342"
                          culture="neutral" />
        <bindingRedirect oldVersion="1.111.6.20"
                         newVersion="2.111.7.20"/>
      </dependentAssembly>
    </assemblyBinding>
  </runtime>
</configuration>
```

Oracle Database Extensions for .NET Registry Options

You can add functionality to Oracle Database Extensions for .NET using Windows registry entries that are located at

HKEY_LOCAL_MACHINE\SOFTWARE\ORACLE\KEY_*ORACLE_HOME*\ODE

Table 2–2 lists registry keys that add functionality to Oracle Database Extensions for .NET and the sections where the keys are discussed.

Table 2–2   Registry Options

Registry Key	Section
.NETFramework	"Selecting a .NET Run Time Version" on page 2-9
ProviderNull	"Backward Compatibility for Nullable ODP.NET Connected Types" on page 2-9
RecreateAppDomain	"Unloading .NET Assemblies for Easy Redeployment" on page 2-8
TraceFileName	"TraceFileName" on page 2-10
TraceOption	"TraceOption" on page 2-10
TraceLevel	"TraceLevel" on page 2-10

Unloading .NET Assemblies for Easy Redeployment

From release 11.1.0.6.20, you can unload .NET assemblies when .NET stored procedure execution completes. This makes it easier to repeatedly test your code during development. If this registry key is not enabled, the exproc.exe process must be stopped and started with each redeployment.

This feature should not be used during performance testing or for production, as it has a negative effect on performance.

To define assembly loading behavior, set the registry value RecreateAppDomain of type REG_SZ under this registry key:

HKEY_LOCAL_MACHINE\SOFTWARE\ORACLE\KEY_*ORACLE_HOME*\ODE

The valid values for `RecreateAppDomain` are:

`0` = .NET Assembly remains loaded when the .NET stored procedure execution completes.

`1` = .NET Assembly is unloaded when the .NET stored procedure execution completes.

Backward Compatibility for Nullable ODP.NET Connected Types

ODP.NET for .NET 2.0 supports a static `Null` property in ODP.NET Connected Types, in addition to the existing support for disconnected types such as `OracleDecimal`. It also supports a public property, `IsNull`, for each of these types to check whether or not objects of these types have been assigned a value.

> **See Also:** *Oracle Data Provider for .NET Developer's Guide* for more information on nullable types

This enables `Null` objects of ODP.NET Connected Types to be propagated to and from a .NET stored procedure. The list of these connected types follows:

- `OracleBlob`
- `OracleClob`
- `OracleBFile`
- `OracleXmlType`

Previous versions of .NET stored procedures expected ODP.NET connected type parameters to be passed as `NULL` rather than a `Type.Null` object. In order to support backward compatibility, the registry string `ProviderNull` can be used to retain the old behavior.

To determine how Oracle Database Extensions for .NET handles passing a `NULL` value to an ODP.NET connected type parameter in a .NET stored procedure, set the registry string `ProviderNull` under this registry key:

HKEY_LOCAL_MACHINE\SOFTWARE\ORACLE\KEY_*ORACLE_HOME*\ODE

The valid values for `ProviderNull` are:

`0` = ODP.NET connected-type parameters are passed as `NULL` rather than `Type.Null` object.

`1` = Oracle Database Extensions for .NET passes a `Type.Null` object to the .NET stored procedure in the case of a null value.

Selecting a .NET Run Time Version

If multiple .NET run time versions are installed on the database computer, then Oracle Database Extensions for .NET defaults to the latest .NET run time available. However, you can configure Oracle Database Extensions for .NET to load a particular .NET run time by setting a registry value.

To specify .NET run time version, set the registry value, `.NETFramework` under this registry key:

HKEY_LOCAL_MACHINE\SOFTWARE\ORACLE\KEY_*ORACLE_HOME*\ODE

Set the registry value to the appropriate .NET run time version, for example, v2.0.50727.

Note: .NET framework 1.x is not supported in this release. For more information see: Mandatory Migration of .NET 1.x Stored Procedures to .NET 2.0.

Debug Tracing

Oracle Database Extensions for .NET provides debug tracing support, which allows logging of all the Oracle Database Extensions for .NET activities into a trace file. Different levels of tracing are available.

The following registry settings should be configured under

`HKEY_LOCAL_MACHINE\SOFTWARE\ORACLE\KEY_ORACLE_HOME\ODE`

TraceOption

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

The valid values for `TraceOption` are:

`0` = Single trace file

`1` = Multiple trace files

Note: You can use Oracle Data Provider for .NET tracing mechanism to troubleshoot ODP.NET specific issues.

TraceFileName

`TraceFileName` specifies the file name that is to be used for logging trace information. If `TraceOption` is set to `0`, the name is used as is. However, if `TraceOption` is `1`, the Thread ID is appended to the file name provided.

The valid values for `TraceFileName` are: any valid path name and file name.

TraceLevel

`TraceLevel` specifies the level of tracing in Oracle Database Extensions for .NET.

The valid values for `TraceLevel` are:

`0` = None

`1` = Entry and exit information

> **See Also:** Debug Tracing section in *Oracle Data Provider for .NET Developer's Guide*

Unhandled Exceptions

.NET stored procedures are hosted inside the external procedure agent `extproc.exe`. .NET run time version 2.0 allows most unhandled exceptions in threads to proceed, which might cause an unhandled exception to terminate `extproc.exe`. This behavior is different from .NET run time versions 1.0 and 1.1, which provide a backstop for many unhandled exceptions.

extproc.exe Config File

Oracle Database Extensions for .NET installs `extproc.exe.config` file in the `ORACLE_BASE\ORACLE_HOME\Bin` directory to force .NET run time version 2.0 to

use the behavior of .NET run time version 1.1. If the Garbage Collector or a thread created inside the .NET stored procedures throws an exception, and if this exception is not handled by the .NET stored procedure, then the tracing mechanism reports the exception.

The following is an example `extproc.exe.config` file:

```
<configuration>
  <runtime>
    <legacyUnhandledExceptionPolicy enabled="1"/>
  </runtime>
</configuration>
```

You can change this file to revert back to .NET 2.0 behavior by modifying the `extproc.exe.config` file as follows:

```
<configuration>
  <runtime>
   <legacyUnhandledExceptionPolicy enabled="0"/>
  </runtime>
</configuration>
```

3

Development and Deployment with Visual Studio

This chapter describes the steps that are required to develop and deploy a .NET stored procedure or function into an Oracle Database.

This chapter contains these topics:

Step 1: Develop the Stored Procedure or Function and Build it into an Assembly

Write the stored procedure or function using Microsoft Visual Studio with an appropriate .NET language.

Use Oracle Data Provider for .NET (`Oracle.DataAccess.Client` and `Oracle.DataAccess.Types`) in a .NET stored procedure or function to provide data access.

Build the stored procedure or function into an assembly as a DLL, and not as an EXE. This is typically accomplished using a Class Library project.

Keep in mind the Oracle Data Provider for .NET limitations and restrictions, especially concerning connections and transactional semantics, when designing and developing a .NET procedure or function that uses data access.

> **See Also:** *Oracle Data Provider for .NET Developer's Guide*

Step 2: Run the Oracle Deployment Wizard for .NET

Run Oracle Deployment Wizard for .NET from Microsoft Visual Studio. Oracle Deployment Wizard for .NET is installed as part of the Oracle Developer Tools for Visual Studio. This wizard requires `SYSDBA` credentials, the name of the assembly to be deployed, and the database it is being deployed to.

Step 3: Choose the Procedure or Function to Deploy and Security Level

Choose the procedure or function to be deployed when the Oracle Deployment Wizard for .NET displays the list of methods for that assembly.

Also, choose the security level.

Oracle Database Extensions for .NET executes .NET stored procedures or functions at a specific security level. The security level dictates the Code Access Permissions granted to a .NET stored procedure or function. By default, .NET stored procedures or functions are executed with the SAFE security level.

The security levels are:

- Safe

  In Safe level, the .NET stored procedure or function is allowed to access only database resources. Access to any external resources such as local files, networks, and so on, is not allowed.

- External

  In External level, the .NET stored procedure or function is allowed to read or write to local files, and to access network resources such as sockets and internet nodes, and so on.

- Unsafe

  In Unsafe level, the .NET stored procedure or function is allowed unrestricted execution including execution of unmanaged code. It is a superset of all other security levels.

  > **See Also:** "Step 3: Choose the Function to Deploy" on page 4-8 for the process of entering security level

Step 4: Determine the Appropriate Parameter Type Mappings

Determine the correct mapping between .NET and Oracle data types for creating a PL/SQL wrapper for the .NET stored procedure or function. The Oracle Deployment Wizard for .NET provides default mappings, but they can be overridden.

In case of overloaded .NET stored procedures or functions, you need to provide distinct names for the PL/SQL wrappers.

Tables in Appendix A provides conversion information.

> **See Also:** "Data Type Conversion" on page A-1

Step 5: Deploy the Procedure or Function

Deploy the procedure or function in the database. The wizard performs the following steps:

1. Connects as `SYSDBA`.

2. Copies the user assembly to the `ORACLE_BASE\ORACLE_HOME`\bin\CLR directory or its subdirectory.

3. Creates an Oracle library object and grants execute privilege on this library object to the database user:

```
CREATE OR REPLACE LIBRARY CLRLIBRARY1_DLL AS '$ORACLE_HOME\
    bin\clr\CLRLibrary1_dll;
```

```
GRANT EXECUTE ON CLRLIBRARY1_DLL TO SCOTT;
```

4. Creates a PL/SQL wrapper in the user's database schema for each procedure or function, according to the parameter type mappings defined by the user.

Step 6: Test the Procedure or Function

Test the .NET stored procedure or function by calling the PL/SQL wrapper.

The PL/SQL wrapper can be located and executed easily using Oracle Developer Tools for Visual Studio, or from a tool like SQL*Plus.

Step 7: Debug the Procedure or Function

Whenever a .NET stored procedure or function is invoked, the Oracle database listener redirects the request to a multithreaded CLR external procedure agent, `extproc.exe`. Each .NET stored procedure or function is executed in the context of the `extproc.exe` process.

1. Ensure that the debug versions of the .NET assembly representing the .NET stored procedure or function and its `pdb` file and dependency DLLs, and their respective `pdb` files are copied to the *ORACLE_BASE\ORACLE_HOME*\bin\clr directory or one of its subdirectories, based on the path provided while creating the library.

2. Attach the debugger to the `extproc.exe` process. Note that the debugger should be capable of debugging .NET code. If Visual Studio is used for debugging, select the Native and Common Language Runtime options in the Attach to Process dialog box. The Native option can be deselected if any .NET stored procedure has already been run in the context of the same `extproc.exe` process.

3. When the debugger is attached, open the .NET stored procedure or function source code and set any breakpoints that are needed, at the required locations.

4. Debug the .NET stored procedure.

> **Note:** You can use Oracle Data Provider for .NET tracing mechanism to troubleshoot application issues. Please see Debug Tracing section in *Oracle Data Provider for .NET Developer's Guide*.

4

Development and Deployment of a .NET Stored Function Demo

This chapter demonstrates how to develop and deploy a .NET stored function.

This chapter contains these topics:

- Overview of .NET Stored Function Demonstration
- Step 1: Create the GetDeptNo Function and Build it into an Assembly
- Step 2: Start the Oracle Deployment Wizard for .NET
- Step 3: Choose the Function to Deploy
- Step 4: Determine the Appropriate Parameter Type Mappings
- Step 5: Deploy the Function to an Oracle Database
- Step 6: Test the Function

> **See Also:** *Oracle Developer Tools for Visual Studio Help* for further information for further information about these components

Overview of .NET Stored Function Demonstration

This demonstration uses Oracle Developer Tools for Visual Studio extensively although some processes can be performed with other Oracle tools. Also, the demonstration refers to the following components of Oracle Developer Tools for Visual Studio:

- Oracle Explorer
- Oracle Project
- Oracle Deployment Wizard for .NET

In this demonstration, you will develop and deploy a .NET stored function named `GetDeptNo`, with a PL/SQL wrapper, `GETDEPTNO`. The `GetDeptNo` function accepts an employee number (`EMPNO`), performs a query, and returns the department number (`DEPTNO`) of the employee.

Step 1: Create the GetDeptNo Function and Build it into an Assembly

This demonstration begins by opening Visual Studio, creating a function, and building it into an assembly.

1. Open Visual Studio and connect as `scott`/`password`. See *Oracle Developer Tools for Visual Studio Help* for information about connecting.

2. From the Visual Studio menu, select **File**, then **New Project**.

3. To create an Oracle Project template, select the project type **Visual C# Projects**, and select **Oracle Project**.

4. Name the project CLRLibrary1 and provide a location for it.

   A class named CLRLibrary1.Class1 appears. It contains a template for a stored procedure.

```
Start Page   Class1.cs

CLRLibrary1.Class1                          StoredProcedure1()

using System;
using System.Data;
using Oracle.DataAccess.Client;
using Oracle.DataAccess.Types;

namespace CLRLibrary1
{
    /// <summary>
    /// Summary description for Class1.
    /// </summary>
    public class Class1
    {
        public static void StoredProcedure1()
        {
            //
            // TODO: Add code here
            //
        }
    }
}
```
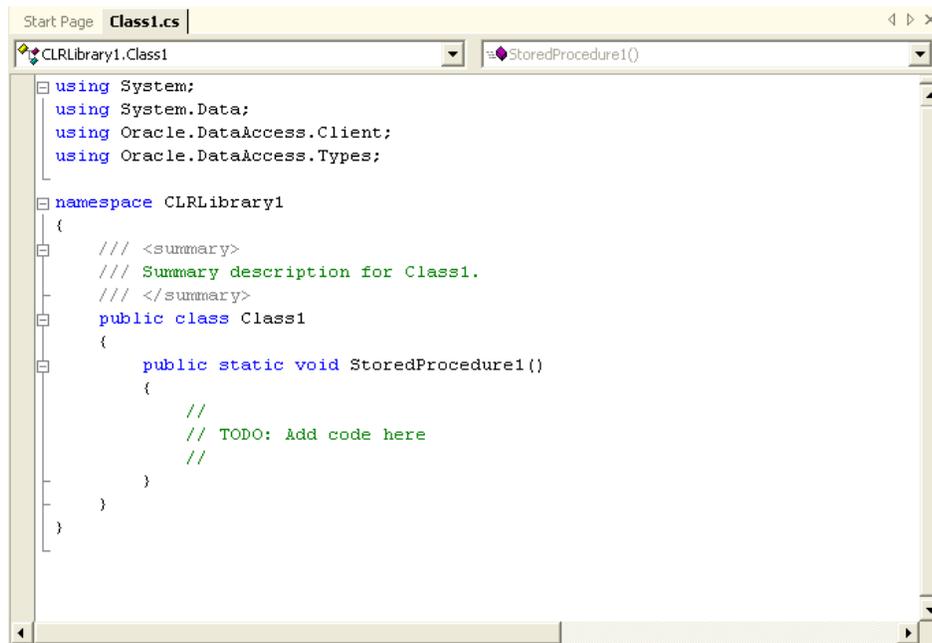
5. Copy the following code over the base class and save.

```csharp
using System;
// use the ODP.NET provider
using Oracle.DataAccess.Client;
using Oracle.DataAccess.Types;

namespace CLRLibrary1
{
  // Sample .NET stored function returning department number for
  // a given employee number
  public class Class1
  {
    public static int GetDeptNo(int empno)
    {
      int deptno = 0;

      // Check for context connection
      OracleConnection conn = new OracleConnection();
      if( OracleConnection.IsAvailable == true )
      {
        conn.ConnectionString = "context connection=true";
      }
      else
      {
        throw new InvalidOperationException("context connection" +
          "not available");
```

```
        }

        conn.Open();
        // Create and execute a command
        OracleCommand cmd = conn.CreateCommand();
        cmd.CommandText = "SELECT DEPTNO FROM EMP WHERE EMPNO = :1";

        cmd.Parameters.Add(":1",OracleDbType.Int32,empno,
          System.Data.ParameterDirection.Input);

        OracleDataReader rdr = cmd.ExecuteReader();

        if (rdr.Read())
          deptno = rdr.GetInt32(0);

        rdr.Close();
        cmd.Dispose();
        conn.Close();

        return deptno;

      } // GetDeptNo
    } // Class1
} // CLRLibrary1
```

6. From the **Build** menu, select **Build Solution** or **Build CLRLibrary1**.

   This builds the code into an assembly.

7. Save.

Step 2: Start the Oracle Deployment Wizard for .NET

Oracle Deployment Wizard for .NET can be started from the build menu.

1. From the **Build** menu, select **Deploy CLRLibrary1**.
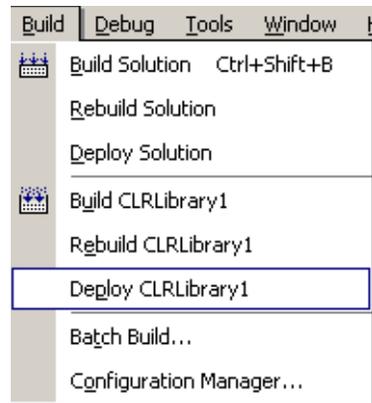
2. When the Welcome window appears, select **Do not show this page again**, if you want.

   Then, click **Next**.
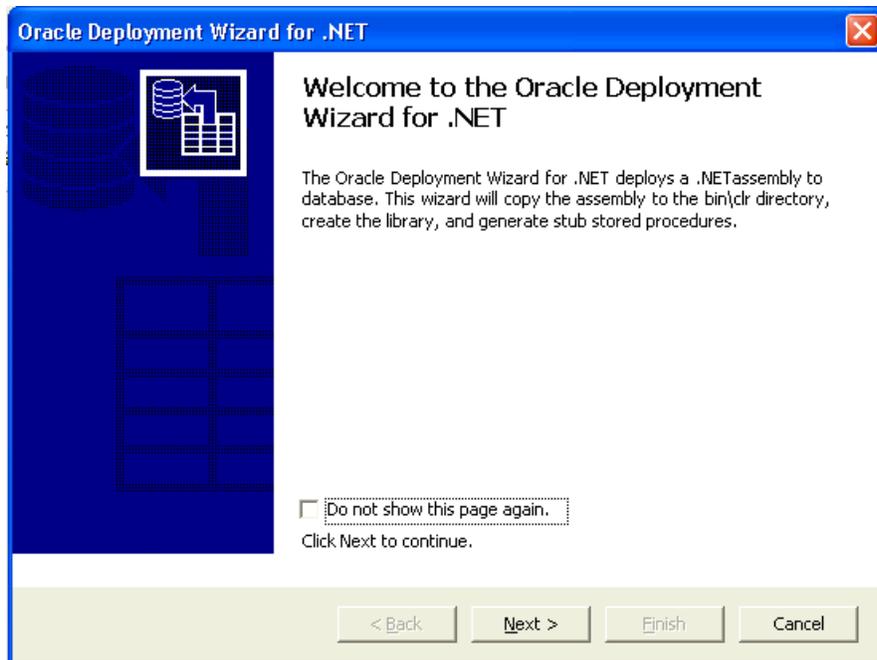
3. The Configure your OracleConnection window appears.

Choose your connection from the drop-down list, and click **Next** or click **New Connection**, if you are not connected.

You must choose or add a `SYSBA` connection.

4. If you have selected New Connection, the Add Connection window appears.

In the Connection Details tab, select the Data source name from the drop-down list.

You can select an option to Use Windows integrated authentication or an option Use a specific user name and password, and enter that information.

If you want, select **Save** password.

The option for Role shows SYSDBA, which is the only available option.

If you want to test the connection, click **Test connection**.

Click **OK**,

5. The Specify your deployment option window appears.

The first time you run the deployment wizard, select **Copy assembly and generate stored procedures**. If you later modify your function or stored procedure, you can run the deployment wizard again, and choose to perform just one of these operations.

Click **Next**.
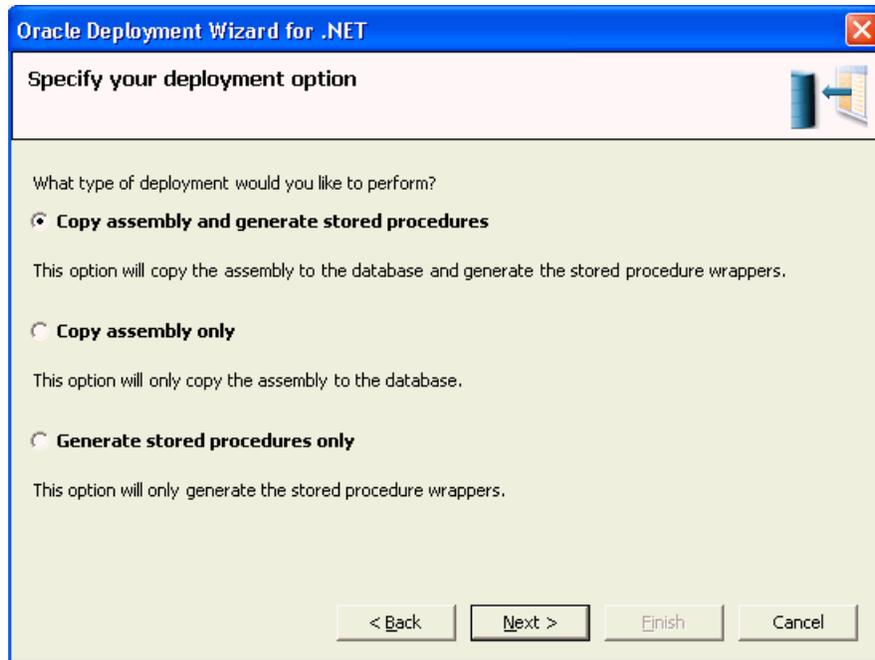
6. The Specify an assembly and library name window appears.

   To specify the assembly, select the project from the drop-down list, or select **File**, and click **Browse** to navigate to the one you want.

   To specify the name of the library database object to be used for the selected assembly, accept the default, select the name from the drop-down list, or enter a new name.

   For this demonstration, accept the default project and library name and click **Next**.
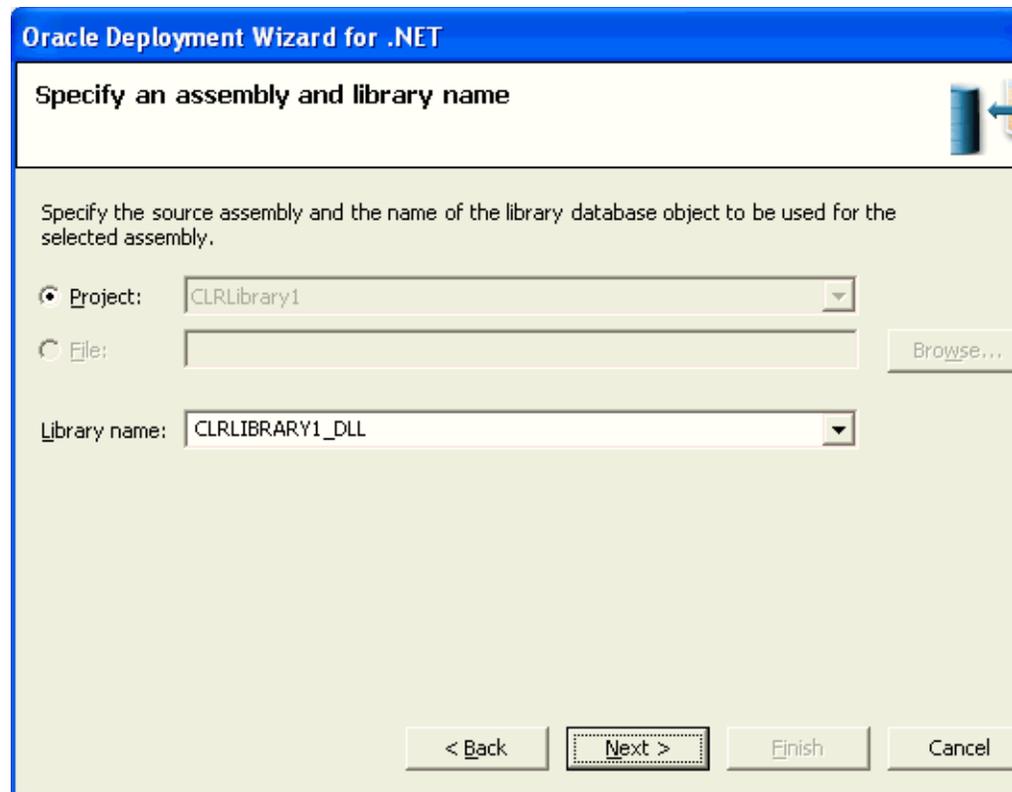
7. The Specify Copy Options window appears.

   To specify the dependent assemblies to copy to the database, select them from the list. The list displays all possible dependent assemblies. In this case, the assemblies displayed have already been copied to the database and, therefore, there is no need to copy them.

   To deploy the assembly to a directory other than the default `bin\clr` directory, modify the destination path. The destination must be a `bin\clr` directory or one of its existing subdirectories.

   For this demonstration, do not select any dependent assemblies, and do not modify the destination path.

   If you want to, select **Overwrite the file if it already exists in the database**.

   Then, click **Next**.
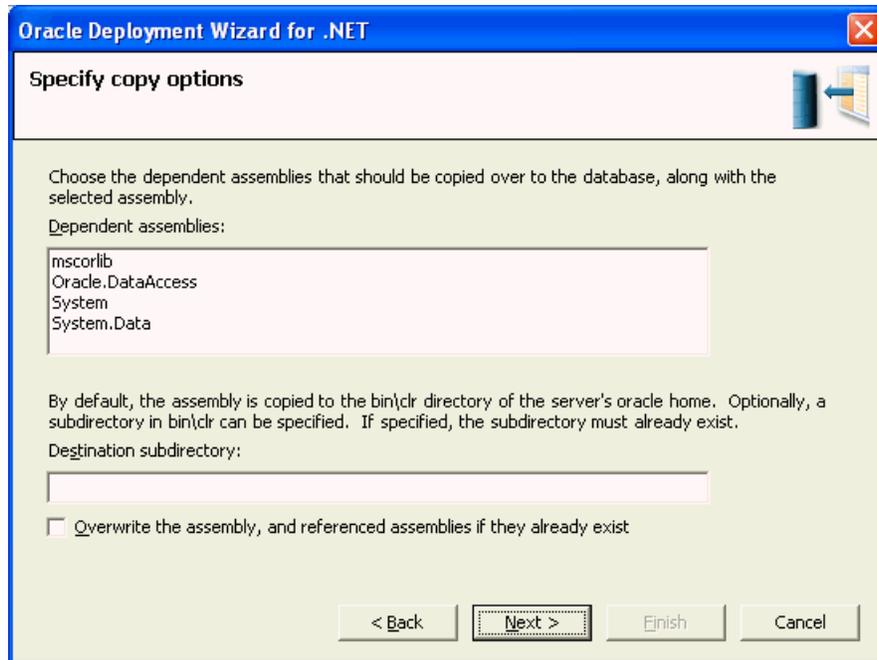
Step 3: Choose the Function to Deploy

The Specify methods and security details window appears.

You can select the entire project to deploy, or expand it to deploy specific functions. Because there is only one function in this project, selecting any one item, checks the entire project. If there were more functions or procedures, you could select individual items to deploy.

If you want to select a different schema to deploy, you can do so here. If the schema you want is not listed, you need to apply different filters. For information on this process, see *Oracle Developer Tools for Visual Studio Help*.

You can set the security level using the drop-down list. The possible levels are:

- **Safe** - (Default)

- **External**

- **Unsafe**

> **See Also:** "Step 3: Choose the Procedure or Function to Deploy and Security Level" on page 3-2

For this demonstration, do the following:

1. Choose `GetDeptNo()` from the list of procedures and functions contained within that assembly.

2. The schema initially says `SYS`. Change it to `Scott`, so that you can deploy it in the `scott` schema.

.

3. Accept the default security level. You can either click **Next** to continue, or you can click **Parameter Type Mapping...** to view the type mappings.
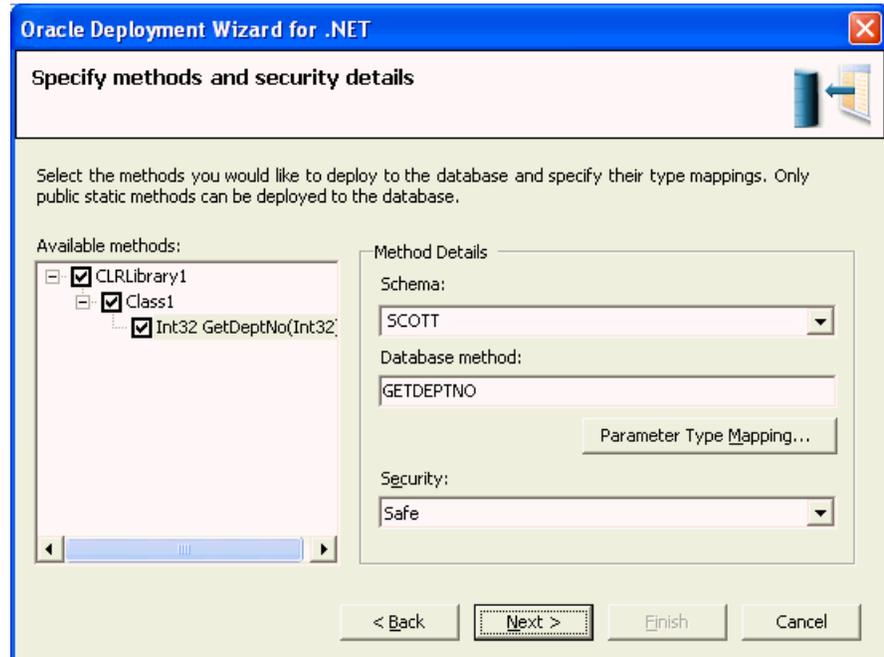
Step 4: Determine the Appropriate Parameter Type Mappings

If you have selected Parameter Type Mapping..., the Parameter Type Mapping window appears, which allows you to change the data type, using the drop-down list.

For this demonstration, accept the default mappings of the .NET data type `System.Int32` to the Oracle type `BINARY_INTEGER`.

Click **OK** to return to the Specify methods and security details window.

> **See Also:** "Data Type Conversion" on page A-1 for data type mapping tables
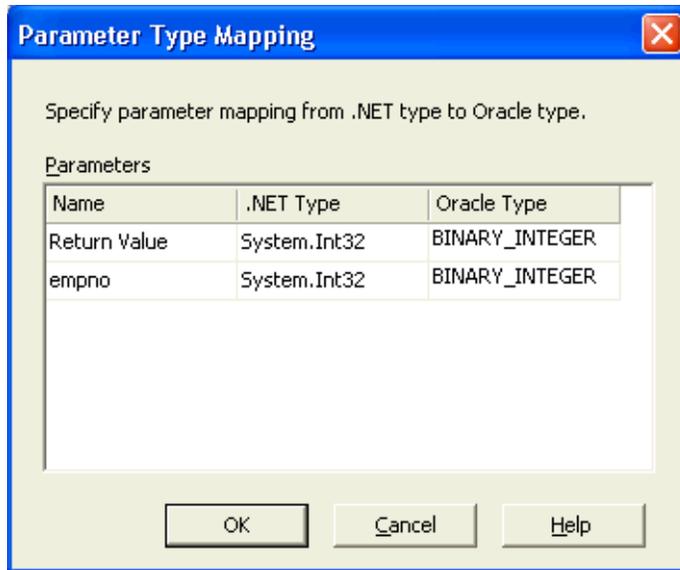
## Step 5: Deploy the Function to an Oracle Database

The Summary window of the Oracle Deployment Wizard for .NET appears, showing all the indicated specifications. This window permits you to modify any values by selecting Back.

To complete the demonstration, do the following:

1. Review the summary.

2. To verify SQL commands, select **Show Script**.
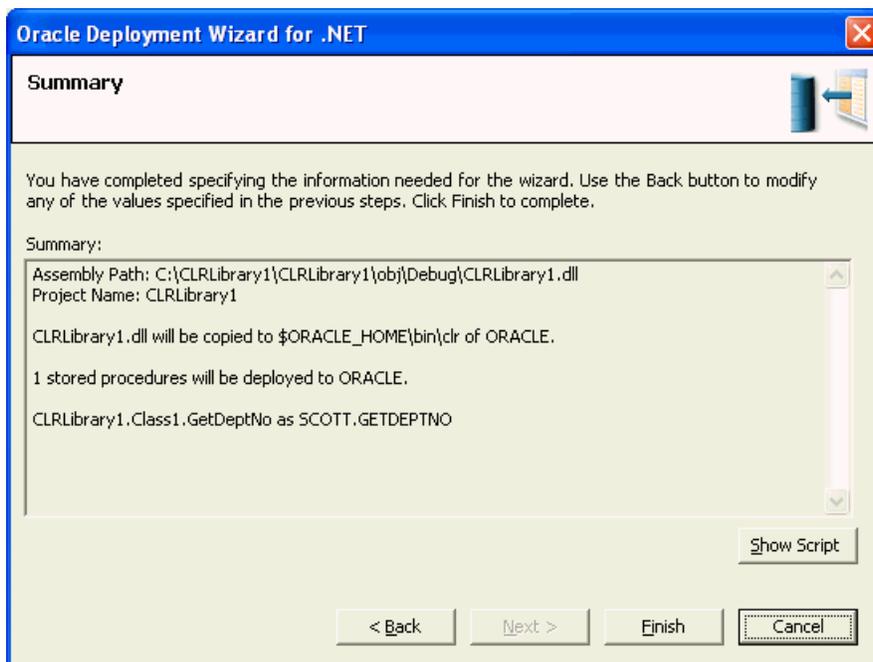
3. When the Show Sql window appears, review the code for the PL/SQL wrapper and click **OK** to return to the Summary window.
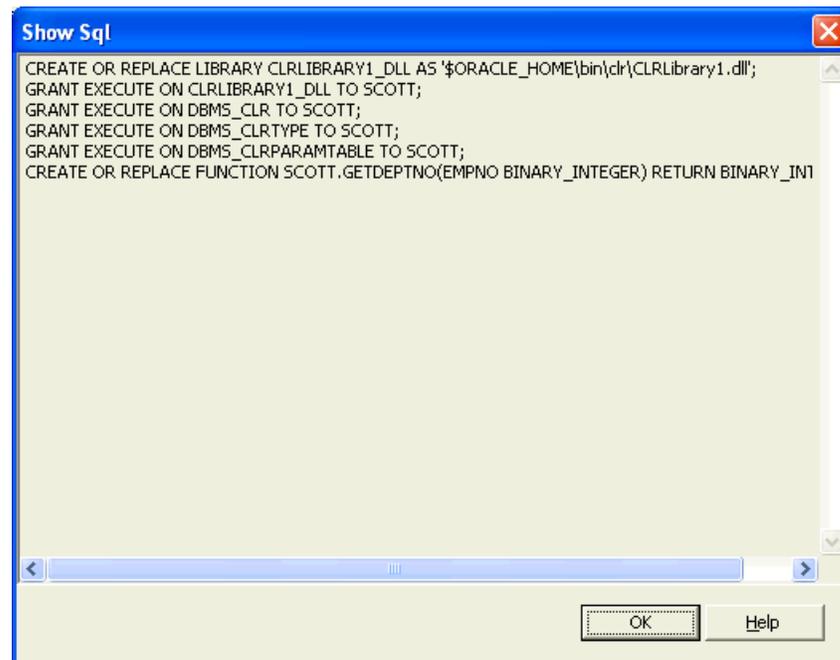
4. Click **Finish** to deploy the `GetDeptNo()` function.

Step 6: Test the Function

At this point, `GetDeptNo()` function has been deployed to the Oracle Database and you are ready to test it by invoking the PL/SQL wrapper function.

You must be connected as the default user, `scott`, in this demonstration, to call the function.

Test the function by invoking it from the following tools:

■ Invoking from Oracle Developer Tools for Visual Studio

■ Invoking from ODP.NET client

■ Invoking from SQL*Plus

Invoking from Oracle Developer Tools for Visual Studio

To locate and call the function from Oracle Developer Tools for Visual Studio:

1. From the **View** menu, select **Oracle Explorer**.

2. Expand the **Functions** node.

3. Locate **GETDEPTNO**.

4. Right-click **GETDEPTNO** and from the menu, select **Run**.
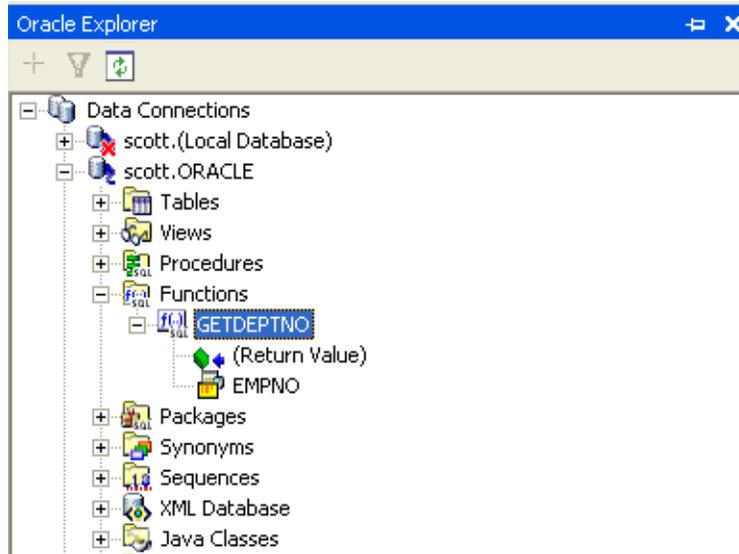
5. When the Run Function dialog box appears, enter employee number 7934 as the input value.
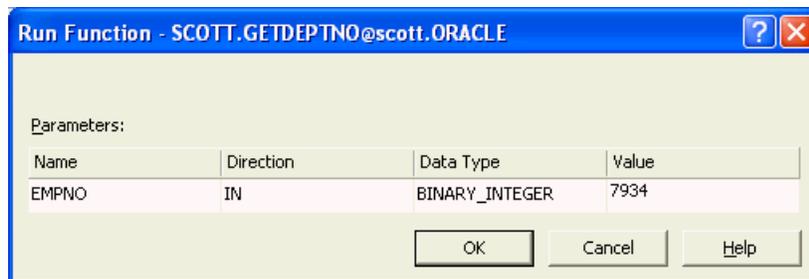
6. Click **OK**.

The output value 10 appears in the Document Window, indicating that employee number 7934 belongs to department 10.
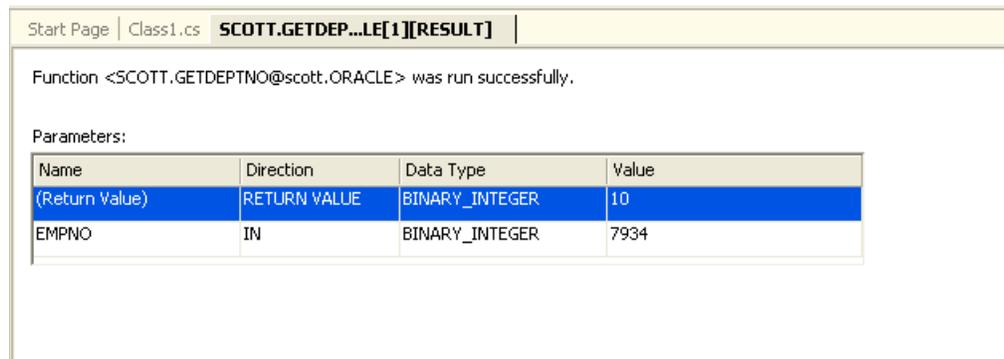
Invoking from ODP.NET client

The following code sample demonstrates how to invoke the PL/SQL wrapper for .NET function.

```
using System;
```

```
using System.Data;
using Oracle.DataAccess.Client;

namespace ODPNETClientApp
{
  public class Class1
  {
    public static void Main()
    {
      int empno  = 7934;
      int deptno = 0;

      try
      {
        // Open a connection to the database
        OracleConnection con = new OracleConnection(
          "User Id=scott; Password=tiger; Data Source=inst1");
        con.Open();

        // Create and execute the command
        OracleCommand cmd = con.CreateCommand();
        cmd.CommandType = CommandType.StoredProcedure;
        cmd.CommandText = "GETDEPTNO";

        // Set parameters
        OracleParameter retParam = cmd.Parameters.Add(":DEPTNO",
            OracleDbType.Int32, System.Data.ParameterDirection.ReturnValue);
        cmd.Parameters.Add(":EMPNO", OracleDbType.Int32, empno,
        System.Data.ParameterDirection.Input);

        cmd.ExecuteNonQuery();
        deptno = (int)retParam.Value;

        Console.WriteLine("\nEmployee# {0} working in department# {1}\n",
          empno, deptno);

        cmd.Dispose();
        con.Close();
      }
      catch (Exception e)
      {
        Console.WriteLine(e.Message);
      }
    }
  } // Class1
} // ODPNETClientApp namespace
```

Invoking from SQL*Plus

To invoke the GetDeptNo() function .NET function from SQL* Plus:

1. Start SQL*Plus and log in as user scott with the password tiger.

2. Enter the following commands:

```
SET SERVEROUTPUT ON;
DECLARE
deptno BINARY_INTEGER;
BEGIN
```

```
        deptno := GetDeptNo(7934);
        DBMS_OUTPUT.PUT_LINE(deptno);
    END;
```

Alternatively, you can execute following statement:

```
SELECT GetDeptNo(7934) FROM DUAL;
```

> **See Also:**
>
> - Oracle Developer Tools for Visual Studio Help
> - *Oracle Data Provider for .NET Developer's Guide*
> - *Oracle Database PL/SQL Packages and Types Reference*
> - *Oracle Database SQL Reference*

A

Data Type Conversion

This appendix contains the following tables used to determine the correct parameter type mappings.

- Table A–1, " Mapping of Oracle Native Data Type to .NET Framework Data Types"
- Table A–2, " Mapping of .NET Framework Data Types to Oracle Native Data Types"
- Table A–3, " Mapping of Oracle Native Data Type to ODP.NET Data Types"
- Table A–4, " Mapping of ODP.NET Data Types to Oracle Native Data Types"

Mapping of Oracle Native Data Type to .NET Framework Data Types

Table A–1 lists the supported mapping of Oracle native data types to the .NET Framework Types.

Table A–1   Mapping of Oracle Native Data Type to .NET Framework Data Types

Oracle Native Data Type	.NET Framework Data Types
BFILE *	Byte[]
BINARY_DOUBLE	Byte, Byte[], Decimal, Double, float, int, Int16, Int32, Int64, long, SByte, short, Single, String, uint, Uint16, Uint32, Uint64, ulong, ushort
BINARY_FLOAT	Byte, Byte[], Decimal, Double, float, int, Int16, Int32, Int64, long, SByte, short, Single, String, uint, Uint16, Uint32, Uint64, ulong, ushort
BINARY_INTEGER	Byte, Byte[], Char, Decimal, Double, float, int, Int16, Int32, Int64, long, SByte, short, Single, String, uint, Uint16, Uint32, Uint64, ulong, ushort
BLOB	Byte[]
BOOLEAN	*Not Supported*
CHAR	Byte, Char, Char[], DateTime, Decimal, Double, float, int, Int16, Int32, Int64, long, SByte, short, Single, String, uint, Uint16, Uint32, Uint64, ulong, ushort
CLOB	Byte, Char, Char[], DateTime, Decimal, Double, float, int, Int16, Int32, Int64, long, SByte, short, Single, String, uint, Uint16, Uint32, Uint64, ulong, ushort
DATE	Byte[], Char[], String, DateTime
FLOAT	Byte, Byte[], Decimal, Double, float, int, Int16, Int32, Int64, long, SByte, short, Single, String, uint, Uint16, Uint32, Uint64, ulong, ushort

Table A–1 (Cont.) Mapping of Oracle Native Data Type to .NET Framework Data Types

Oracle Native Data Type	.NET Framework Data Types
INTERVAL DAY TO SECOND	Byte[], Char, Char[], String, TimeSpan
INTERVAL YEAR TO MONTH	Byte, Byte[], Char, Char[], int, Int16, Int32, Int64, long, short, String
LONG	Byte, Char, Char[], DateTime, Decimal, Double, float, int, Int16, Int32, Int64, long, SByte, short, Single, String, uint, Uint16, Uint32, Uint64, ulong, ushort
LONG RAW	Byte[]
NCHAR	Byte, Char, Char[], DateTime, Decimal, Double, float, int, Int16, Int32, Int64, long, SByte, short, Single, String, uint, Uint16, Uint32, Uint64, ulong, ushort
NCLOB	Byte, Char, Char[], DateTime, Decimal, Double, float, int, Int16, Int32, Int64, long, SByte, short, Single, String, uint, Uint16, Uint32, Uint64, ulong, ushort
NUMBER	Byte, Byte[], Char, DateTime, Decimal, Double, float, int, Int16, Int32, Int64, long, SByte, short, Single, String, uint, Uint16, Uint32, Uint64, ulong, ushort
NVARCHAR2	Byte, Char, Char[], DateTime, Decimal, Double, float, int, Int16, Int32, Int64, long, SByte, short, Single, String, uint, Uint16, Uint32, Uint64, ulong, ushort
PLS_INTEGER	Byte, Byte[], Char, Decimal, Double, float, int, Int16, Int32, Int64, long, SByte, short, Single, String, uint, Uint16, Uint32, Uint64, ulong, ushort
RAW	Byte[]
REAL	Byte, Byte[], Decimal, Double, float, int, Int16, Int32, Int64, long, SByte, short, Single, String, uint, Uint16, Uint32, Uint64, ulong, ushort
ROWID	Char[], String
TIMESTAMP	Byte[], Char[], String, DateTime
TIMESTAMP WITH LOCAL TIME ZONE	Byte[], Char[], String, DateTime
TIMESTAMP WITH TIME ZONE	Byte[], Char[], String, DateTime
UROWID	Char[], String
VARCHAR2	Byte, Char, Char[], DateTime, Decimal, Double, float, int, Int16, Int32, Int64, long, SByte, short, Single, String, uint, Uint16, Uint32, Uint64, ulong, ushort
XMLType	Char[], String

Mapping of .NET Framework Data Types to Oracle Native Data Types

Table A–2 lists the supported mapping of .NET Framework Types to Oracle native data types.

Table A–2 Mapping of .NET Framework Data Types to Oracle Native Data Types

.NET Framework Data Types	Oracle Native Data Type
Byte	BINARY_DOUBLE, BINARY_FLOAT, BINARY_INTEGER, CHAR, CLOB, FLOAT, INTERVAL YEAR TO MONTH, LONG, NCHAR, NCLOB, NUMBER, NVARCHAR2, PLS_INTEGER, REAL, VARCHAR2

.NET Framework Data Types	Oracle Native Data Type
Byte[]	BINARY_DOUBLE, BINARY_FLOAT, BINARY_INTEGER, BLOB, DATE, FLOAT, INTERVAL YEAR TO MONTH, LONG RAW, NUMBER, PLS_INTEGER, RAW, REAL, TIMESTAMP, TIMESTAMP WITH LOCAL TIME ZONE, TIMESTAMP WITH TIME ZONE
Char	BINARY_INTEGER, CHAR, CLOB, INTERVAL DAY TO SECOND, INTERVAL YEAR TO MONTH, LONG, NCHAR, NCLOB, NUMBER, NVARCHAR2, PLS_INTEGER, VARCHAR2, XMLType
Char[]	CHAR, CLOB, DATE, INTERVAL DAY TO SECOND, INTERVAL YEAR TO MONTH, LONG, NCHAR, NCLOB, NVARCHAR2, ROWID, TIMESTAMP, TIMESTAMP WITH LOCAL TIME ZONE, TIMESTAMP WITH TIME ZONE, UROWID, VARCHAR2, XMLType
DateTime	CHAR, CLOB, DATE, LONG, NCHAR, NCLOB, NVARCHAR2, TIMESTAMP, TIMESTAMP WITH LOCAL TIME ZONE, TIMESTAMP WITH TIME ZONE, VARCHAR2
Decimal	BINARY_DOUBLE, BINARY_FLOAT, BINARY_INTEGER, CHAR,CLOB, FLOAT, LONG, NCHAR, NCLOB, NVARCHAR2, NUMBER, PLS_INTEGER, REAL, VARCHAR2
Double	BINARY_DOUBLE, BINARY_FLOAT, BINARY_INTEGER, CHAR, CLOB, FLOAT, LONG, NCHAR, NCLOB, NUMBER, NVARCHAR2, PLS_INTEGER, REAL, VARCHAR2
float	BINARY_DOUBLE, BINARY_FLOAT, BINARY_INTEGER, CHAR, CLOB, FLOAT, LONG, NCHAR, NCLOB, NUMBER, NVARCHAR2, PLS_INTEGER, REAL, VARCHAR2
int	BINARY_DOUBLE, BINARY_FLOAT, BINARY_INTEGER, CHAR, CLOB, FLOAT, INTERVAL YEAR TO MONTH, LONG, NCHAR, NCLOB, NUMBER, NVARCHAR2, PLS_INTEGER, REAL, VARCHAR2
Int16	BINARY_DOUBLE, BINARY_FLOAT, BINARY_INTEGER, CHAR, CLOB, FLOAT, INTERVAL YEAR TO MONTH, LONG, NCHAR, NCLOB, NUMBER, NVARCHAR2, PLS_INTEGER, REAL, VARCHAR2
Int32	BINARY_DOUBLE, BINARY_FLOAT, BINARY_INTEGER, CHAR, CLOB, FLOAT, INTERVAL YEAR TO MONTH, LONG, NCHAR, NCLOB, NUMBER, NVARCHAR2, PLS_INTEGER, REAL, VARCHAR2
Int64	BINARY_DOUBLE, BINARY_FLOAT, BINARY_INTEGER, CHAR, CLOB, FLOAT, INTERVAL YEAR TO MONTH, LONG, NCHAR, NCLOB, NUMBER, NVARCHAR2, PLS_INTEGER, REAL, VARCHAR2
long	BINARY_DOUBLE, BINARY_FLOAT, BINARY_INTEGER, CHAR, CLOB, FLOAT, INTERVAL YEAR TO MONTH, LONG, NCHAR, NCLOB, NUMBER, NVARCHAR2, PLS_INTEGER, REAL, VARCHAR2
SByte	BINARY_DOUBLE, BINARY_FLOAT, BINARY_INTEGER, CHAR, CLOB, FLOAT, LONG, NCHAR, NCLOB, NUMBER, NVARCHAR2, PLS_INTEGER, REAL, VARCHAR2
short	BINARY_DOUBLE, BINARY_FLOAT, BINARY_INTEGER, CHAR, CLOB, FLOAT, INTERVAL YEAR TO MONTH, LONG, NCHAR, NCLOB, NUMBER, NVARCHAR2, PLS_INTEGER, REAL, VARCHAR2
Single	BINARY_DOUBLE, BINARY_FLOAT, BINARY_INTEGER, CHAR, CLOB, FLOAT, LONG, NCHAR, NCLOB, NUMBER, NVARCHAR2, PLS_INTEGER, REAL, VARCHAR2

Table A–2   (Cont.) Mapping of .NET Framework Data Types to Oracle Native Data Types

.NET Framework Data Types	Oracle Native Data Type
String	BINARY_DOUBLE, BINARY_FLOAT, BINARY_INTEGER, CHAR, CLOB, DATE, FLOAT, INTERVAL DAY TO SECOND, INTERVAL YEAR TO MONTH, LONG, NCHAR, NCLOB, NUMBER, NVARCHAR2, PLS_INTEGER, REAL, ROWID, STRING, TIMESTAMP, TIMESTAMP WITH LOCAL TIME ZONE, TIMESTAMP WITH TIME ZONE, UROWID, VARCHAR2, XMLTYPE
TimeSpan	INTERVAL DAY TO SECOND
uint	BINARY_DOUBLE, BINARY_FLOAT, BINARY_INTEGER, CHAR, CLOB, FLOAT, LONG, NCHAR, NCLOB, NUMBER, NVARCHAR2, PLS_INTEGER, REAL, VARCHAR2
Uint16	BINARY_DOUBLE, BINARY_FLOAT, BINARY_INTEGER, CHAR, CLOB, FLOAT, LONG, NCHAR, NCLOB, NUMBER, NVARCHAR2, PLS_INTEGER, REAL, VARCHAR2
Uint32	BINARY_DOUBLE, BINARY_FLOAT, BINARY_INTEGER, CHAR, CLOB, FLOAT, LONG, NCHAR, NCLOB, NUMBER, NVARCHAR2, PLS_INTEGER, REAL, VARCHAR2
Uint64	BINARY_DOUBLE, BINARY_FLOAT, BINARY_INTEGER, CHAR, CLOB, FLOAT, LONG, NCHAR, NCLOB, NUMBER, NVARCHAR2, PLS_INTEGER, REAL, VARCHAR2
ulong	BINARY_DOUBLE, BINARY_FLOAT, BINARY_INTEGER, CHAR, CLOB, FLOAT, LONG, NCHAR, NCLOB, NUMBER, NVARCHAR2, PLS_INTEGER, REAL, VARCHAR2
ushort	BINARY_DOUBLE, BINARY_FLOAT, BINARY_INTEGER, CHAR, CLOB, FLOAT, LONG, NCHAR, NCLOB, NUMBER, NVARCHAR2, PLS_INTEGER, REAL, VARCHAR2

Mapping of Oracle Native Data Type to ODP.NET Data Types

Table A–3 lists the supported mapping of Oracle native data types to the ODP.NET Types.

Table A–3   Mapping of Oracle Native Data Type to ODP.NET Data Types

Oracle Native Data Type	ODP.NET Type
BFILE *	OracleBFile
BINARY_DOUBLE	OracleDecimal
	OracleString
BINARY_FLOAT	OracleDecimal
	OracleString
BINARY_INTEGER	OracleDecimal
	OracleString
BLOB	OracleBinary
	OracleBlob
BOOLEAN	*Not Supported*
CHAR	OracleString
CLOB	OracleClob
	OracleString

Table A–3 (Cont.) Mapping of Oracle Native Data Type to ODP.NET Data Types

Oracle Native Data Type	ODP.NET Type
DATE	OracleDate
	OracleString
	OracleTimeStamp
	OracleTimeStampTZ
	OracleTimeStampLTZ
FLOAT	OracleDecimal
	OracleString
INTERVAL DAY TO SECOND	OracleString
	OracleIntervalDS
INTERVAL YEAR TO MONTH	OracleIntervalYM
	OracleString
LONG	OracleString
LONG RAW	OracleBinary
NCHAR	OracleString
NCLOB	OracleClob
	OracleString
NUMBER	OracleDecimal
	OracleString
NVARCHAR2	OracleString
PLS_INTEGER	OracleDecimal
	OracleString
RAW	OracleBinary
REAL	OracleDecimal
	OracleString
ROWID	OracleString
TIMESTAMP	OracleDate
	OracleString
	OracleTimeStamp
	OracleTimeStampTZ
	OracleTimeStampLTZ
TIMESTAMP WITH LOCAL TIME ZONE	OracleDate
	OracleString
	OracleTimeStamp
	OracleTimeStampTZ
	OracleTimeStampLTZ
TIMESTAMP WITH TIME ZONE	OracleDate
	OracleString
	OracleTimeStamp
	OracleTimeStampTZ
	OracleTimeStampLTZ

Table A–3 (Cont.) Mapping of Oracle Native Data Type to ODP.NET Data Types

Oracle Native Data Type	ODP.NET Type
UROWID	OracleString
VARCHAR2	OracleString
XMLType	OracleClob
	OracleString
	OracleXmlType

Mapping of ODP.NET Data Types to Oracle Native Data Types
Table A–4 lists the supported mapping of the ODP.NET Types to the Oracle native data types.

Table A–4 Mapping of ODP.NET Data Types to Oracle Native Data Types

ODP.NET Type	Oracle Native Data Type
OracleBFile	BFILE *
OracleBinary	BLOB
	LONG RAW
	RAW
OracleBlob	BLOB
OracleClob	CLOB
	NCLOB
	XMLType
OracleDate	DATE
	TIMESTAMP
	TIMESTAMP WITH LOCAL TIME ZONE
	TIMESTAMP WITH TIME ZONE
OracleDecimal	BINARY_DOUBLE
	BINARY_INTEGER
	BINARY_FLOAT
	FLOAT
	NUMBER
	PLS_INTEGER
	REAL
OracleIntervalDS	INTERVAL DAY TO SECOND
OracleIntervalYM	INTERVAL YEAR TO MONTH

Table A–4   (Cont.) Mapping of ODP.NET Data Types to Oracle Native Data Types

ODP.NET Type	Oracle Native Data Type
OracleString	BINARY_DOUBLE
	BINARY_FLOAT
	BINARY_INTEGER
	CHAR
	CLOB
	DATE
	FLOAT
	INTERVAL DAY TO SECOND
	INTERVAL YEAR TO MONTH
	LONG
	NCHAR
	NCLOB
	NVARCHAR2
	NUMBER
	PLS_INTEGER
	REAL
	ROWID
	STRING
	TIMESTAMP
	TIMESTAMP WITH LOCAL TIME ZONE
	TIMESTAMP WITH TIME ZONE
	UROWID
	VARCHAR2
	XMLType
OracleTimeStamp	DATE
	TIMESTAMP
	TIMESTAMP WITH LOCAL TIME ZONE
	TIMESTAMP WITH TIME ZONE
OracleTimeStampTZ	DATE
	TIMESTAMP
	TIMESTAMP WITH LOCAL TIME ZONE
	TIMESTAMP WITH TIME ZONE
OracleTimeStampLTZ	DATE
	TIMESTAMP
	TIMESTAMP WITH LOCAL TIME ZONE
	TIMESTAMP WITH TIME ZONE
OracleXmlType	XMLType

* BFILE Mapping to .NET Framework

An Oracle native BFILE type parameter can be converted to a .NET Byte[].
However, converting a Byte[] to an Oracle native BFILE type is not supported. This
means BFILE to .NET Byte[] conversion can be done only if the BFILE parameter

type on the database side is an `IN` parameter and the corresponding parameter on the .NET stored procedure is an `IN` parameter of type `Byte[]`. For a `BFILE INOUT`, or `OUT` parameter or a `RETURN VALUE`, the corresponding .NET stored procedure parameters must be of type `Oracle.DataAccess.Types.OracleBFile`. Otherwise, an exception is thrown.

PL\SQL Associative Array

PL\SQL Associative array is not supported.

B

Troubleshooting Common Errors

This appendix discusses common errors.

Users may encounter various errors while running the PL/SQL wrapper. Causes and recommended actions for such errors are listed below.

ORA-03113: end-of-file on communication channel

Cause: The connection between Client and Server process was broken. It may also happen if the external agent `extproc` crashes for some reason.

Action: There was a communication error that requires further investigation. First, check for network problems and review the SQL*Net setup. Also, look in the `alert.log` file for any errors. Finally, test to see whether the server process is dead and whether a trace file was generated at failure time. There may be some system calls in the .NET function which might terminate the process. Remove such calls.

ORA-03114: not connected to ORACLE

Cause: The connection between Client and Server process was broken. This may also happen if the external agent `extproc` crashes for some reason.

Action: There was a communication error which requires further investigation. First, check for network problems and review the SQL*Net setup. Also, look in the `alert.log` file for any errors. Finally, test to see whether the server process is dead and whether a trace file was generated at failure time. There may be some system calls in the .NET function which might terminate the process. Remove such calls.

ORA-20100: System.BadImageFormatException. The format of the file is invalid.

Cause: The .NET Assembly is not in the proper format.

Action: Fix the .NET assembly format that contains the .NET stored procedures or functions. You need to recompile the .NET assembly.

ORA-20100: System.IO.FileNotFoundException. File or assembly name <assemblyname>.dll, or one of its dependencies, was not found.

Cause: The .NET Assembly or one of its dependent assemblies is not available in *ORACLE_BASE\ORACLE_HOME*\bin\clr or in one of the subdirectories as specified during the creation of the library object by the wizard.

Action: Copy the .NET assembly and all its dependent assemblies to *ORACLE_BASE\ORACLE_HOME*\bin\clr or to one of its subdirectories as appropriate.

ORA-20100: System.MissingMethodException

Cause: `MissingMethodException` is thrown for many possible reasons including:

- The stored procedure or function name does not match the actual stored procedure or function name defined in the .NET assembly.

- The number, sequence, and type of parameters passed do not match the actual parameters in the .NET stored procedure.

Action: Check the name of the called stored procedure or function for spelling mistakes or case mismatch (upper or lower). Check the number of parameters and check that the type and sequence of the parameters match those of the stored procedure or function defined in the .NET assembly.

ORA-20100: System.Reflection.TargetException. <typename> type not found

Cause: The namespace and/or the class name used in the PL/SQL wrapper is not defined in the .NET Assembly.

Action: Check the class name for spelling mistakes or case mismatch. Check the .NET assembly code for the type.

ORA-20100: System.Security.SecurityException

Cause: .NET stored procedure or function could not be executed with current security level.

Action: Use the appropriate security level. For example, if the .NET stored procedure or function requires file system access, then it should be created with EXTERNAL security level.

ORA-28575: unable to open RPC connection to external procedure agent

Cause: Initialization of a network connection to the `extproc` agent did not succeed. This problem can be caused by network problems, incorrect listener configuration, or incorrect transfer code.

Action: Check listener configuration in `LISTENER.ORA` and `TNSNAMES.ORA`, or check Oracle Names Server. Verify that the multithreaded `extproc` configuration entries are correct.

ORA-28578: protocol error during callback from an external procedure

Cause: An internal protocol error occurred. This could be due to some registration issue during creation of PL/SQL wrapper.

Action: Recreate the PL/SQL wrapper using the wizard.

PLS-00201: identifier 'DBMS_CLR' must be declared

Cause: Either Oracle Database Extensions for .NET is not installed and configured properly or the .NET stored procedure has not been deployed correctly using the Oracle Deployment Wizard for .NET.

Action: Use the Database Configuration Assistant to configure Oracle Database Extensions for .NET, if it has not been installed already. Deploy the .NET stored procedure using the Oracle Deployment Wizard for .NET.

Glossary

assembly

Assembly is the Microsoft term for the module that is created when a DLL or .EXE is complied by a .NET compiler.

Common Language Runtime

Microsoft Common Language Runtime (CLR) is the component of the .NET framework that allows many languages to create and develop applications using the same library.

Dynamic Link Library (DLL)

An executable file that a Windows application can load when needed.

external procedure

A function written in a third-generation language (3GL), such as C, and callable from within PL/SQL or SQL as if it were a PL/SQL function or procedure.

implicit database session

The database session of the caller.

Microsoft .NET Framework Class Library

The Microsoft .NET Framework Class Library provides the classes for the .NET framework model.

namespace

- .NET:

  A namespace is naming device for grouping related types. More than one namespace can be contained in an assembly.

- XML Documents:

  A namespace describes a set of related element names or attributes within an XML document.

Oracle Net Services

The Oracle client/server communication software that offers transparent operation to Oracle tools or databases over any type of network protocol and operating system.

PL/SQL

Oracle's procedural language extension to SQL.

result set

The output of a SQL query, consisting of one or more rows of data.

stored function

A stored function is a PL/SQL block that Oracle stores in the database and can be executed from an application.

stored procedure

A stored procedure is a PL/SQL block that Oracle stores in the database and can be executed from an application.

Index

V

VB.NET, 1-1
Visual Studio, 1-5
Visual Studio .NET, 1-1

W

Windows operating systems, 1-1, 2-1
Windows registry entries
   ImagePath, 2-3
   .NETFramework, 2-8
   ProviderNull, 2-8
   RecreateAppDomain, 2-8
wizard
   Oracle Deployment Wizard for .NET, 1-5
wrapper, 1-1, 1-2
   PL/SQL, 4-10