

# Delphi 2005 BDP Deployment

## How to deploy ASP.NET and WinForms BDP applications

by Bob Swart

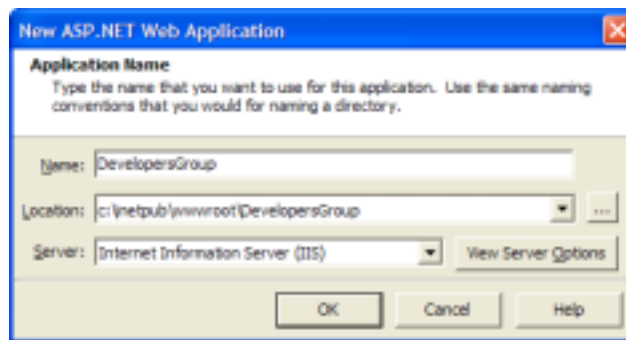
In this article, I will explain and demonstrate the steps needed for the successful deployment of a Delphi 2005 application using the Borland Data Provider (BDP) to work with databases.

### Deployment of Borland Data Provider (BDP) Applications

The techniques in this article apply to WinForms as well as ASP.NET Web Forms applications that use the Borland Data Provider. Since the deployment of an ASP.NET Web Forms application is a bit more complex than the deployment of a WinForms application, I'll use an ASP.NET Web Forms application as main example. One big difference is that a WinForms application doesn't use the Deployment Manager, but the set of BDP-related files to deploy is the same.

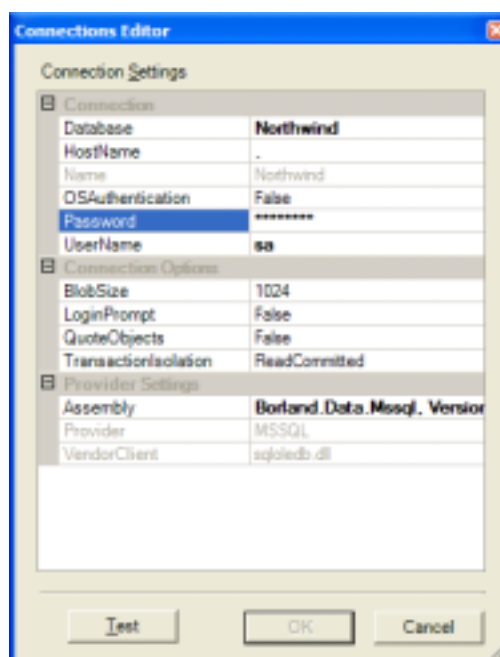
### ASP.NET Web Forms Example

Since this article is about deployment, let's not spend too much time on building an ASP.NET example. Do File | New - ASP.NET Web Application - Delphi for .NET, which gives you the New ASP.NET Web Application wizard. Specify the name of the project, like DevelopersGroup:



Click on OK to create a new ASP.NET Web Forms project.

Go to the Data Explorer tab (in the upper-right corner of the IDE), and open the MSSQL node in the Providers treeview. For this example, I'm using the Northwind database, for which I have an existing connection defined with a named user and password to login to the database (no Windows OS Authentication - see also the note at the end of this article).

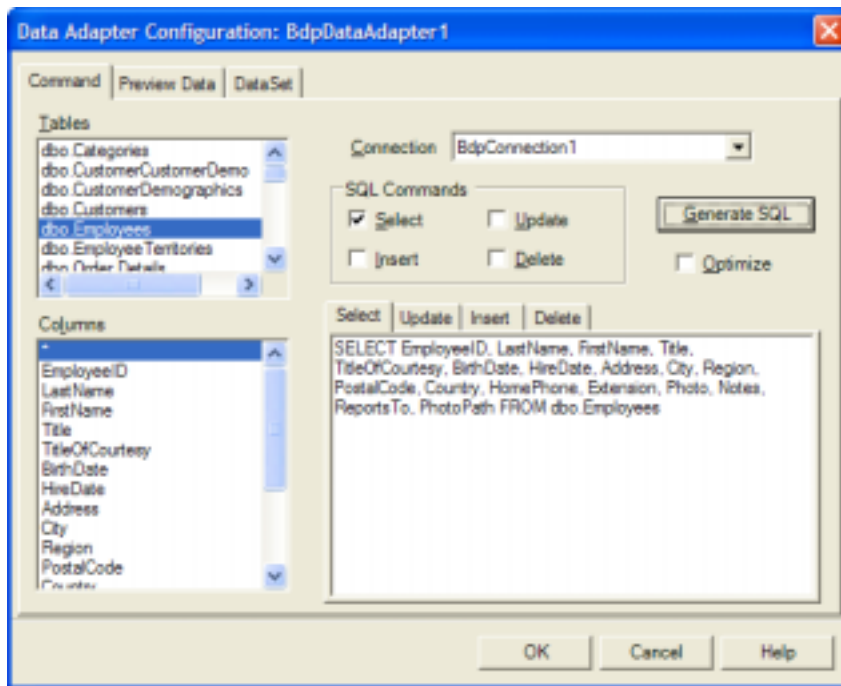


We will use the ASP.NET web page to display a DataGrid with overview information from the Employees table of the SQL Server Northwind example database. The SQL scripts to generate this example database can be found on the MSDN website at <http://www.microsoft.com/downloads/details.aspx?FamilyID=06616212-0356-46A0-8DA2-EEBC53A68034&displaylang=en>

Open the SQL Server Connection node to the Northwind database, drag the Employees table from the Data Explorer and drop it on your Form.

This will automatically create a BdpConnection and BdpDataAdapter component in the non-visual components area of the designer. Select the BdpDataAdapter and click on the Configure Data Adapter link at the bottom of the Object Inspector.

On the first tab of the Data Adapter Configuration dialog, uncheck the Insert, Update and Delete SQL Commands options, and click on the Generate SQL button to regenerate the Select command.



Optionally you can go to the second tab to get a preview of the data. Then go to the DataSet tab and select a New DataSet to put the result of the SELECT command.



Click on the OK button to close the dialog, and then set the Active property of the BdpDataAdapter to True. You will now have a BdpConnection, BdpDataAdapter and DataSet component in the non-visual components area of the ASP.NET forms designer.

Now, place a DataGrid on the Form, and set its DataSource property to dataSet1, and its DataMember property to Employees. Double-click on the Web Form area (below the DataGrid for example) to get to the Page\_Load event in the Code Editor, and write the following line of code:

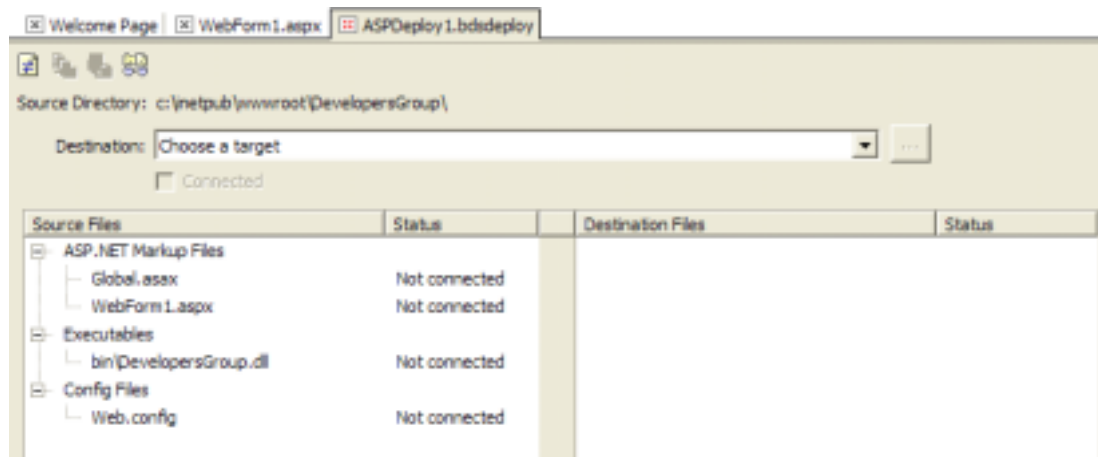
```
procedure TWebForm1.Page_Load(sender: System.Object; e: System.EventArgs);
begin
    if not Page.IsPostBack then DataGrid1.DataBind
end;
```

That's it. You can add something more if you wish (see Dr.Bob Examines #52 for example at <http://www.drbob42.com/examines/examin52.htm>), but this is the basis of an ASP.NET project using the Borland Data Provider to talk to the Northwind database. The remainder of this article is only concerned about the deployment of the project.

## Delphi 2005 Deployment Manager

Deploying ASP.NET applications before Delphi 2005 was a process of collecting all .aspx files, the .asax, web.config, assembly itself and all related assemblies (most of which were mentioned in the References list, fortunately). For each new version of the ASP.NET application, you had to collect everything all over again, or find a way to automate the process. Delphi 2005 has done just that: it includes a Deployment Manager that can be used on ASP.NET or IntraWeb projects - and is flexible enough to be used in other situations as well (once you find out how, but that's another story).

Go to the Project Manager, right-click on the Deployment node and select the "New ASP.NET Deployment" option. This will show the ASP.NET Deployment Manager page, listing all the files that are part of your ASP.NET project and need to be deployed. Or almost all these files, since you may notice that it doesn't appear to list all required assemblies. Specifically, it doesn't show the BDP assemblies in the screenshot below:



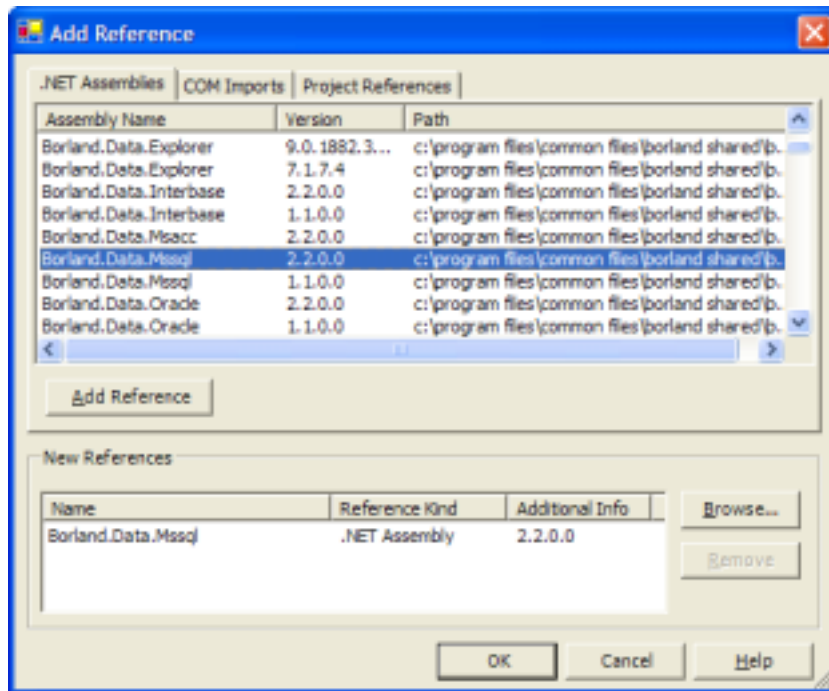
A way to justify this is to note that the BDP assemblies (like Borland.Data.Common.dll and Borland.Data.Provider.dll) are signed assemblies. Since ASP.NET doesn't officially support assemblies signed with a strong name to be deployed in the bin directory or a virtual directory, these officially need to be deployed in the Global Assembly Cache. You may have to contact your ISP with regard to those deployment needs. However, as we'll see in this article, it's not really required (the application will also work without having to access the GAC on the web server)

## BDP Assemblies

The real explanation why the Borland BDP assemblies don't show up in the Deployment Manager is because of the way they are referenced in our project. If you open up the References node for the DevelopersGroup project, you'll notice the Borland.Data.Common.dll and Borland.Data.Provider.dll assemblies. In order to add these two assemblies to the project directory, we must right-click on them and set the Copy Local option set to True (you can also do this in the Object Inspector by the way). As a result, when we compile the ASP.NET project, these two assemblies will be placed in the target directory (right next to the DevelopersGroup.dll in the Bin subdirectory of the DevelopersGroup virtual directory).

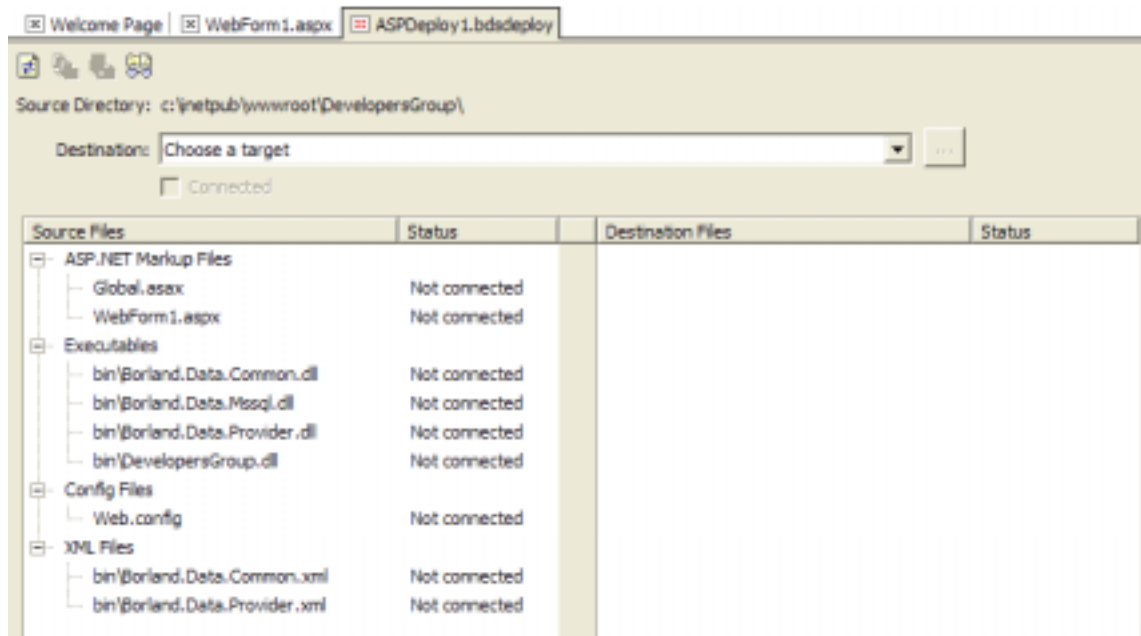
However, adding Borland.Data.Common.dll and Borland.Data.Provider.dll is still not enough. As you may remember, we use Microsoft SQL Server and a BdpConnection component connecting to SQL Server. And yet the list of files doesn't show a SQL Server specific BDP driver. The reason for that is the fact that the BDP

drivers are loaded dynamically based on their reference in the `ConnectionString` property of the `BdpConnection` component. This is the place where the reference is made to the `Borland.Data.Mssql.dll` assembly. In order to fix this, we should also add a reference to the `Borland.Data.Mssql` to our project, using the `Add Reference` dialog:



Note that other BDP assemblies also exist for InterBase, Oracle, IBM DB2, MS Access, etc. and also note that the version number is 2.2.0.0 (this is the version number you get after installing Update #2 for Delphi 2005 - the original version number of these BDP assemblies is 2.0.0.0).

After the reference to `Borland.Data.Mssql` is added, right-click on the new node to ensure the `Copy Local` option is enabled, and then recompile the project. The `Deployment Manager` should now have a fairly complete list of files to deploy:

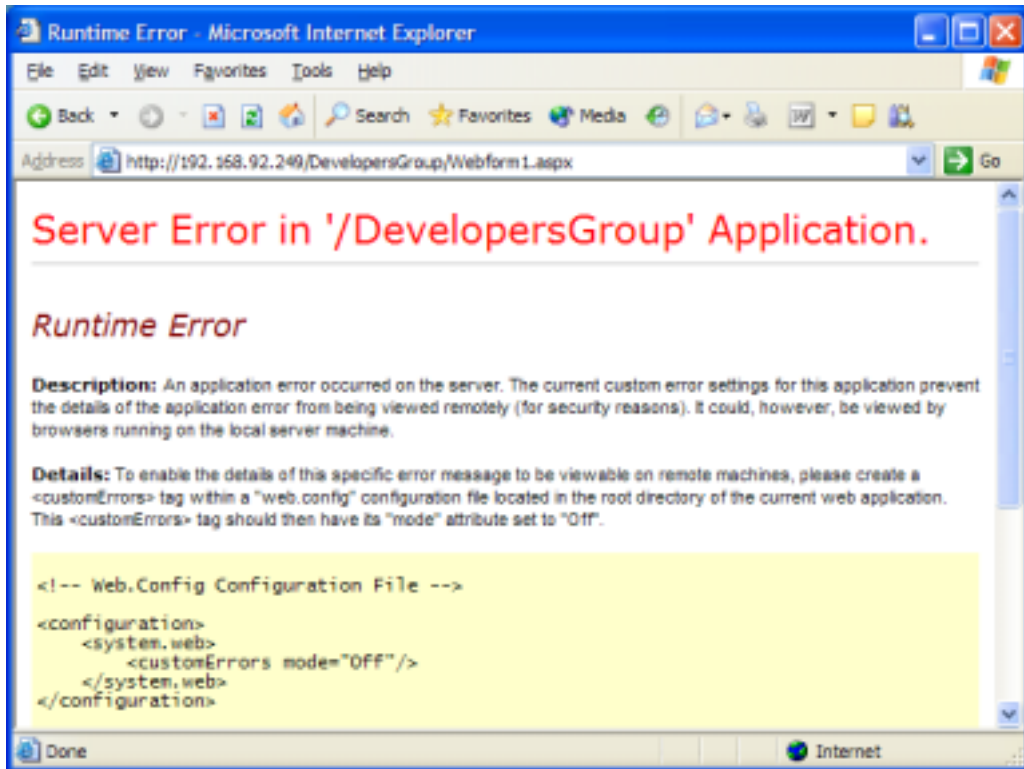


You can now connect to the deployment machine, and click on the “`deploy new and changed files`” button. It will then automatically copy or FTP upload all your new and modified files. That’s really a time-saver, especially in those deadline situations where all you want to do is focus on the changes you make to the code, without needing to worry about the actual files to deploy (so you will never accidentally forget to deploy any file that belongs to the project).

## Deployment Test

Once all these files are deployed, you can start the application by specifying the address of the deployment machine and the name of the main web form. Obviously, SQL Server and the database itself must also be deployed on the target machine (or on a separate database server), and you have to ensure that the `ConnectionString` property is updated to connect to the database on the new location. I just want to focus on the application and the required BDP-specific files now.

After we've deployed an ASP.NET application using the Deployment Manager as outlined above, we can try to view an ASP.NET page that uses the BDP components to connect to the database. Unfortunately, an error message appears to tell you that not everything works fine.



And even worse: it's not very clear from this error message what the problem is, since we get a so-called "user friendly error message". These can be controlled using the `customErrors` node in the `web.config` file:

---

```
<!-- CUSTOM ERROR MESSAGES

Set customErrors mode values to control the display of user-friendly
error messages to users instead of error details (including a stack
trace):

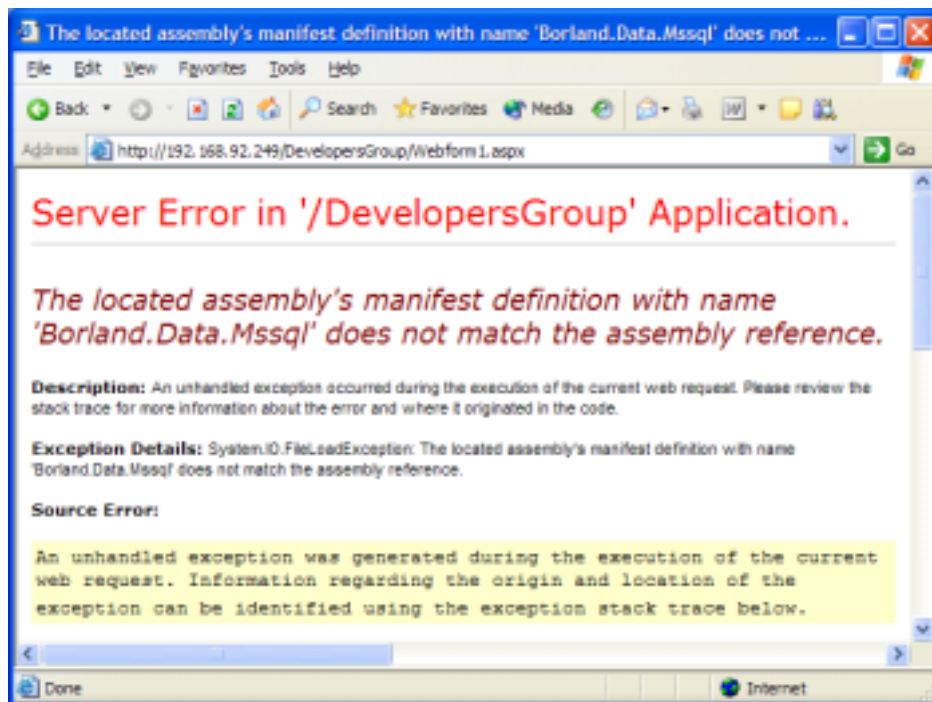
"On" Always display custom (friendly) messages
"Off" Always display detailed ASP.NET error information.
"RemoteOnly" Display custom (friendly) messages only to users not
running on the local Web server. This setting is recommended for
security purposes, so that you do not display application detail
information to remote clients.
-->
<customErrors
mode="RemoteOnly"
/>
```

---

In order to see the error message, you need to change `RemoteOnly` to `Off` - just for a short while! Note that the value is case-sensitive; "off" will not do it.

## Borland.Data.Mssql assembly reference

Once `customErrors` has been set to "Off", and the `web.config` file re-uploaded, reloading web form will produce the following more useful error information: the located assembly's manifest definition with the name `Borland.Data.Mssql` does not match the assembly reference:



If you read the detailed error information (not visible in the screenshot), you'll notice the problem, which seems to be in comparing the assembly name, resulting in a mismatch "Minor Version". This problem is due to the fact that I'm using Delphi 2005 Update #3, which - like Update #2 - uses version 2.2.0.0 of the BDP assemblies.

However, if you check the `bdpConnections.xml` or `BdpDataSources.xml` files on your development machine, you'll notice that the BDP assemblies are still listed with a 2.0.0.0 version number. This is the version number that's also used inside the `BdpConnection` components (specifically, their `ConnectionString` used to load the BDP assemblies). But since we're deploying the 2.2.0.0 versions of these assemblies, the deployment server complains.

Our development machine doesn't complain, since it contains a policy definition that allows 2.2.0.0 to be used when 2.0.0.0 is mentioned (without that, our pre-Update #2 Delphi 2005 projects still mentioning 2.0.0.0 would no longer compile or work with the new 2.2.0.0 assemblies).

## First Workaround

One workaround is to modify the 2.0.0.0 version number in the `ConnectionString` of the `BdpConnection` components, and change them to 2.2.0.0. Then, recompile the application, and redeploy to make it this error go away.

## GAC Solution

An alternative solution (brought to my attention by Peter Sawatzki) is to apply the assembly policies to the deployment web server too. Unfortunately, the policy files are hidden in the Delphi 2005 Update #2, but you can get the policy DLLs and config files from the GAC of your development machine by going to the `\Windows\assembly\GAC\policy.2.0.Borland.Data.Common`, `\Windows\assembly\GAC\policy.2.0.Borland.Data.Provider`, and `\Windows\assembly\GAC\policy.2.0.Borland.Data.XXX` (with XXX being the name of the database driver; DB2, Interbase, Msacc, Mssql, Oracle, or Sybase), and optionally also the `\Windows\assembly\GAC\policy.2.0.Borland.Data.DataSync` directories. You need to deploy both the `.dll` and `.config` files from these directories to the GAC on the web server.

## web.config Solution

However, often you do not have access to the GAC on the web server. In those cases, you need to copy the policy references from the `.config` files in the policy directories, and append them to the `web.config` file. In our case, for the `Borland.Data.Mssql` assembly, we need to copy the contents of the `policy.2.0.Borland.Data.Mssql.config` file, and place it at the bottom of the `web.config` file (integrating the configuration tags) as follows:

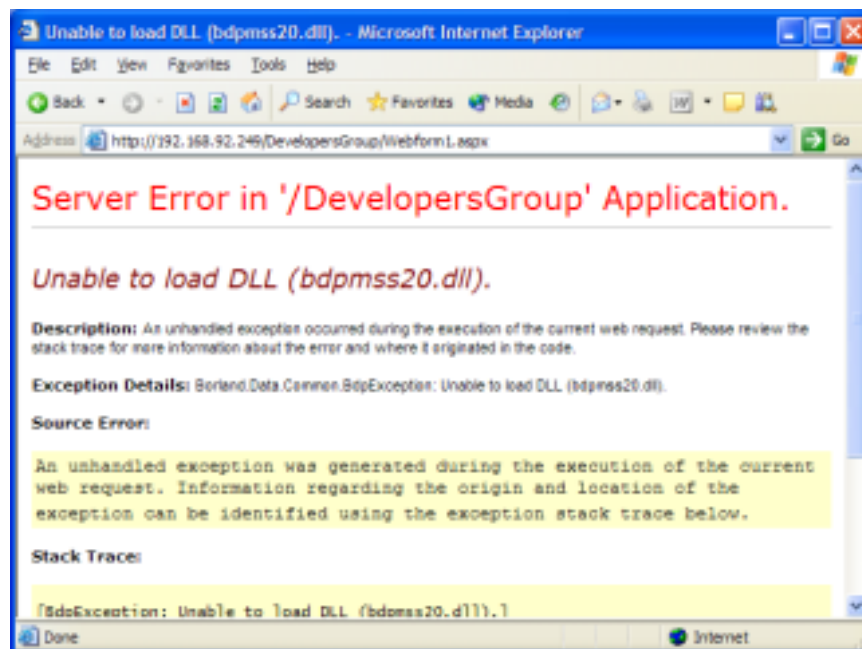
```

<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <system.web>
    ... // normal contents of web.config
  </system.web>

  <runtime>
    <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
      <dependentAssembly>
        <assemblyIdentity name="Borland.Data.Mssql"
          publicKeyToken="91d62ebb5b0d1b1b"
          culture="neutral" />
        <bindingRedirect oldVersion="2.0.0.0"
          newVersion="2.2.0.0" />
      </dependentAssembly>
    </assemblyBinding>
  </runtime>
</configuration>

```

After you've modified the web.config, you can redeploy the application, which will then give you another error message (the same error message that you will get if you apply any other of the mentioned solutions or workarounds).

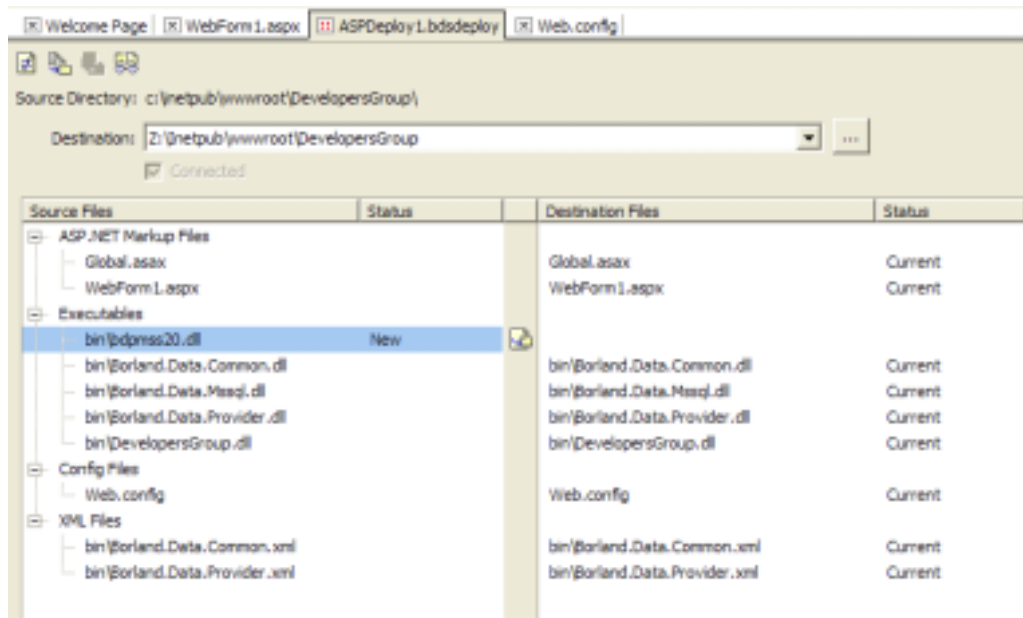


## Missing bdpmss20.dll

It appears that the file bdpmss20.dll is also needed by the ASP.NET application. In fact, it is needed by any application using the Borland Data Provider (if you use SQL Server as DBMS, that is). This was not the case for Delphi 2005 ASP.NET applications before Update #2, but since Update #2, we need to deploy this additional file. Note that you may have got this error message even before the Borland.Data.Mssql.dll related assembly reference error message – the order can differ from project to project.

The easiest way to fix this problem is to add bdpmss20.dll to the deployment set of files, by copying bdpmss20.dll to the DevelopersGroup's Bin directory on the development machine. Bdpms20.dll can be found in the BDS\3.0\Bin directory - where the other bdpXXX20.dll files can be found as well - but not in the Common Files\Borland Shared\BDS\3.0\Shared Assemblies directory. Copying bdpmss20.dll to the project Bin directory will add it to the list of deployment files in the Deployment Manager, so we can deploy it as well.

Once everything is redeployed again, the ASP.NET application will load and run just fine. No more error messages related to the Borland Data Provider for .NET.



## ASP.NET Database Access

Of course, you may still get an error about other missing assemblies (like the Borland.Data.Web.dll), but these are no longer related to the Borland Data Provider, and should just be added to the list of files to deploy.

Another error message that you may encounter is a problem connecting to your database. When using Microsoft SQL Server you may get a message that tells you that NT NETWORK NT AUTHORITY/NETWORK SERVICE is not allowed to access the SQL Server database. The latter problem can be experienced when using SQL Server MSDE as database on Windows XP or Windows Server 2003. The error is due to the fact that MSDE by default only allows Windows OS Authentication to be used, yet an ASP.NET user runs under the NT AUTHORITY/NETWORK SERVICE account which has no access permission to your databases. It's possible to solve this problem by granting NT AUTHORITY/NETWORK SERVICE access to your databases, but a better idea is to avoid using Windows OS Authentication in the first place, and start using a specific login and password for your database. The only trouble is that MSDE by default only allows OS Authentication to be used, unless you installed MSDE with the SECURITYMODE=SQL parameter, which most people don't know about until after they installed MSDE. With this parameter, MSDE will be installed with mixed mode authentication: both Windows OS Authentication and SQL Server Authentication.

In order to turn on Mixed Authentication after you've already installed MSDE, you need to manually modify the registry (at your own risk!). Assuming MSDE was installed as default instance, you need the registry value at HKLM\Software\Microsoft\MSSqlserver\MSSqlServer\LoginMode and change it (carefully!) from 1 (Windows Authentication only) to 2 (Mixed Authentication).

Note that you need to restart MSDE to allow the new setting to become effective.

## Summary

In this article, I've explained and demonstrated how we need to deploy the Borland.Data.Common.dll and Borland.Data.Provider.dll as well as the Borland.Data.xxx.dll for the specific BDP driver, plus a bdpxxx20.dll file (found in the BDS\3.0\Bin directory). Furthermore, we may need to fix the fact that the BDP driver version number in the ConnectionString of the BdpConnection component is not equal to the actual BDP driver version number (either by modifying the ConnectionString, or making sure the web server knows about the relevant BDP assembly policy, as I've shown in detail).

These steps have to be taken for any application (WinForms or ASP.NET Web Forms) using the Borland Data Provider to access databases. For ASP.NET applications you also need to ensure that you can access the database correctly. Once you know the drill, it's easy.

For an actual demo of these techniques, see <http://www.eBob42.com/courseware> where I've deployed my Delphi 2005 ASP.NET application selling Delphi 2005 courseware manuals online.