

Delphi 2007 and DBX4

Version 4 of the Database Express Framework

by Bob Swart

In this paper, I will cover some of the new features in the latest edition of dbExpress (also called DBX4) which ships as part of Delphi 2007 and C++Builder 2007.

DBX4?

The first thing I asked myself was: if Delphi 2007 includes DBX4, where did the other versions belong? Since dbExpress was first released as part of Delphi 6 to offer cross-platform data access for Delphi and Kylix, I assume that's where version 1 started.

Delphi 2006 shipped with dbExpress drivers that had 30 in the name (like dbxint30.dll and dbxmss30.dll for example), so that would have been version 3. Delphi 2005 and lower has no such filename indicator, and uses dbexpint.dll and dbexpmss.dll for example. The fact that the dbExpress drivers use the same name for Delphi 6, 7, 8 and 2005 (as well as C++Builder 6), caused some problems from time to time, when an older version of the driver was found in the path and loaded before the newer driver (especially when Delphi 7 was installed after Delphi 6 and hence later in the search path).

We have to look inside the dbexpXXX.dll files in these versions of Delphi (and C++Builder) in order to get the actual version number. This tells us that Delphi 7 through 2005 is using dbExpress version 2.5. We cannot use this to verify the version number of dbExpress in Delphi 6 and C++Builder 6, since their drivers do not contain version information, yet.

Delphi 7 dbExpress 2.5

Delphi 8 dbExpress 2.5

Delphi 2005 dbExpress 2.5

Delphi 2006 dbExpress 3.0

Delphi 2007 dbExpress 3.0 as well as dbExpress 4.0 aka DBX4

Note that the dbExpress drivers that ship with Delphi 2007 are so-called dual interfaced drivers, that implement both the older dbExpress 3.0 and the new dbExpress 4 (DBX4) interfaces. This means that you can use the dbxint30.dll and dbxmss30.dll from Delphi 2007 also with Delphi 2006 for example.

The important part is to know that Delphi 2006 uses dbExpress version 3 (with a "30" in the driver filename), and Delphi 2007 uses DBX4 (with dbExpress 3 supported by the drivers for additional backwards compatibility), and all older versions of Delphi use dbExpress 2.5 or lower. The reason why this is important to know is that DBX4 contains backwards compatibility for dbExpress 3 drivers, but not for older versions. So if you are using a specific – say third party – driver for your special database, then you can still use that driver with Delphi 2007 and DBX4 if that driver was written for Delphi 2006 (or 2007, of course). If you still have a dbExpress 2.5 driver, then you will have to upgrade the driver and get a new version.

Single Source

DBX4 is written in Delphi, and no longer uses COM (which was the basis for dbExpress 3) or C/C++. In fact, DBX4 is a complete rewrite of the previous version(s) of dbExpress. And this time, the team has produced a single source version, which gets compiled to a Win32 and a .NET target (and Kylix hasn't been mentioned for years now, so it's unlikely this version of DBX will have Linux support in it). For each database, the team now only has to support a single DBX4 driver. And given the fact that for .NET the Borland Data Provider (BDP) will be replaced by DBX4 as well, this results in much cleaner and easier to maintain solution for the CodeGear engineers.

But what about us? What about existing code for example? For Win32, not a lot will change, since DBX4 still contains support for dbExpress 3 drivers. So you can take your Delphi 2006 application and recompile it with Delphi 2007 and it will just work. For a .NET application using BDP, the changes will be more drastic. Delphi 2007 for .NET – or Highlander as the codename goes – will support version 2.0 of the .NET Framework, and no longer .NET version 1.1. The Borland Data Provider, available in Delphi 8 for .NET, Delphi 2005 and Delphi 2006 is based on ADO.NET 1.1. The .NET Framework 2.0 introduces ADO.NET 2.0 which deprecates

ADO.NET 1.1 and BDP as well, so a new solution was needed anyway (you cannot simply take your ADO.NET 1.1 applications and use them in .NET 2.0 either). Although Highlander is still work-in-progress, rest assured that DBX4 will be the new standard for Delphi 2007 for .NET (2.0) developers. More information and details will follow as Highlander gets closer to its release.

Delegate Drivers

One of the most powerful new features of DBX4 is the ability to define delegate drivers. Using delegate drivers we can build a stack of drivers, where the bottom one is the one actually talking to the database, and the ones on top are used for tracing, connection pooling, auditing or any other purpose you can think of. The application makes use of a single connection component, but the connection itself (as well as all other public methods and properties) will bubble through the delegate driver stack. This can be quite useful at times, and will be demonstrated in more detail shortly.

There are a number of delegate drivers already available with DBX4 in Delphi 2007. These include the TDBXPool delegate driver and the TDBXTrace delegate driver. The TDBXPool delegate driver can be used for connection pooling to other DBX4 drivers. The TDBXTrace delegate driver can be used to trace methods and events of other DBX4 drivers. This can be done at different levels of detail.

There is also a special DBX3 adapter driver in dbxadapter30.dll which ensures that dbExpress 3.0 drivers are loaded and used by the framework as if they are DBX4 drivers. You can use this delegate driver to use any dbExpress 3.0 driver that shipped with Delphi 2006, or – more likely – one that was provided by a third-party dbExpress driver vendor like CoreLab or Upscene.

Practice

In practice, when you start a new Delphi 2007 project, you can still find the dbExpress components in the dbExpress category of the component palette. The same “old” components are still there, including the TSimpleDataSet (a pre-combined merger of the TSQLDataSet, TDataSetProvider and TClientDataSet – something which you should only use for small and simple projects). In fact, if you load an older Delphi project that uses dbExpress, you often only have to recompile the project in order to make it work with DBX4. This is certainly the case with Delphi 2006 projects, since these mention the dbXXXX30.dll drivers in the TSQLConnection component, which are still present in Delphi 2007 (this time supporting both the dbExpress 3.0 and DBX4 interfaces). For older projects, you may have to use the Connections Editor for the TSQLConnection component to upgrade the connection information.

The Connections Editor is actually the first place an actual difference can be observed compared to earlier versions of Delphi. When you open up the Connections Editor for the TSQLConnection component you will see two new entries: the DBXPool and DBXTrace delegate drivers.

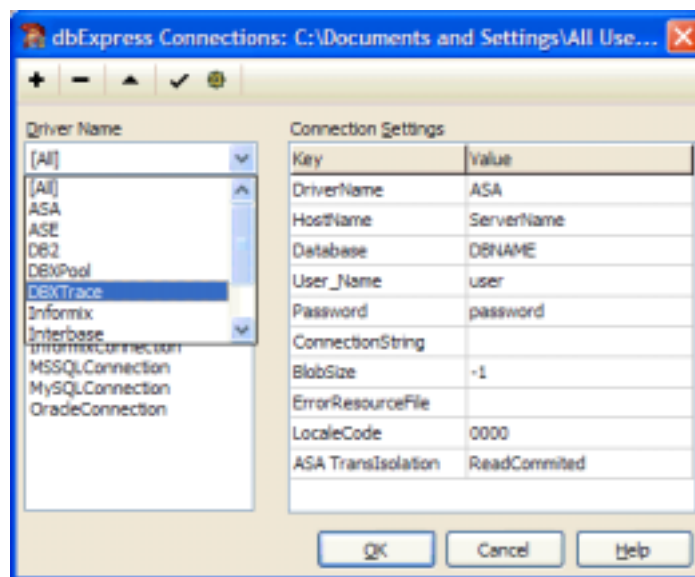


Figure A.

If you select the DBXPool driver, you will see the pre-defined DBXPoolConnection which has four attributes visible in the Connections Editor: DriverName (DBXPool, don't modify that one), MaxConnections, MinConnections and ConnectTimeout. The DBXTraceConnection in the DBXTrace category has only two attributes defined by default: DriverName and TraceFlags, the latter set to NONE by default.

Let's build a new connection, and see some of the features of the delegate drivers in action. For this example, we can use any database (I will use SQL Server in the screenshots, but anything for which you have a DBX4 driver is OK).

In the Connections Editor, click on the button with the plus sign to create a new connection. As the driver name, I pick MSSQL, and I specify DGConnection as the name of the new dbExpress connection.

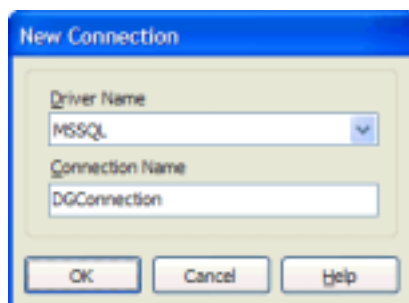


Figure B.

This will give me a new DGConnection item in the list of MSSQL drivers. The following screenshot shows all attributes that we can configure for this new MSSQL connection.

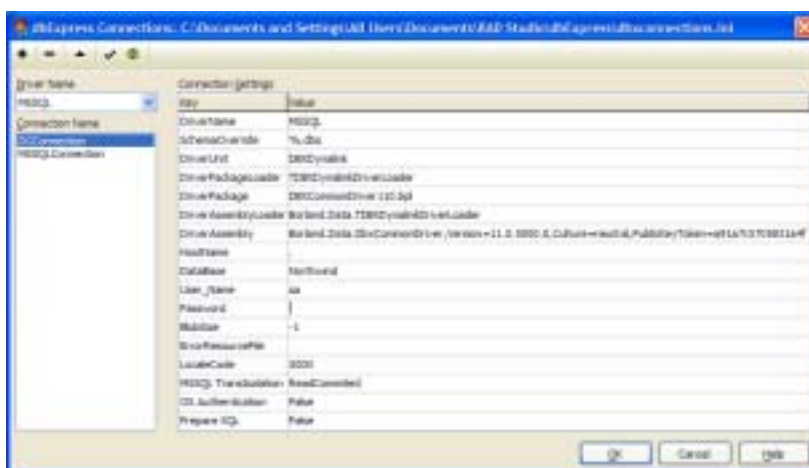


Figure C.

There are a number of attributes here that do not appear in the default MSSQLConnection. Specifically, we now get DriverUnit, DriverPackageLoader, DriverPackage, DriverAssemblyLoader and DriverAssembly which are not visible in the MSSQLConnection entry. You don't want to change them, by the way. The DriverPackage is the one relevant for Win32 applications, and illustrates that we need to deploy the DBXCommonDriver110.bpl package with our DBX4 application. The funny thing is that a file with this name cannot be found on your machine. Instead, we must deploy DBXCommonDriver100.bpl (the "100" is used to ensure version number compatibility with Delphi 2006, since Delphi 2007 for Win32 was said to be a non-breaking release). I'll get back to more deployment details at the end of this article.

To get a list of the actual dbExpress drivers that have been installed in your copy of Delphi 2007, click on the button with the little clockwork icon, which results in the following list on my system:

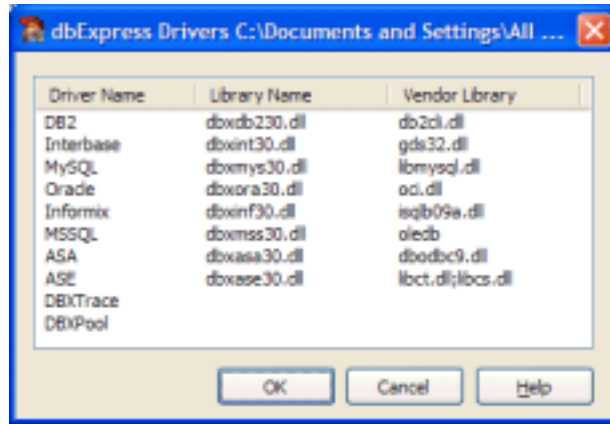


Figure D.

These are the actual dbExpress drivers that also have to be deployed. Unfortunately, dbExpress drivers can no longer be linked into our Delphi executable, as was previously the case. The units DBExpDB2, DBExpINT, DBExpMySQL and DBExpORA which were present in Delphi 2006 are no longer available.

Delegate Connection

If we take a look back at the screenshot in figure C, you may notice that it contains some old and new attributes, but no way to specify a delegate connection chain. For this, we need an attribute with key string "DelegateConnection". You cannot add this key with the Connections Editor, but once you have build a connection, you can use the Params property of the TSQLConnection component to edit the list of parameters. Here, you can add the DelegateConnection parameter with a value of DBXPoolConnection or DBXTraceConnection. However, this will only modify the specific instance of DGConnection that is used by this TSQLConnection component. If we want to make changes to the DGConnection itself, and add or remove attributes for example, we need to manually edit the file at C:\Documents and Settings\All Users\Documents\RAD Studio\dbExpress\dbxconnections.ini (a filename which is mentioned in the caption of the dbExpress Connections Editor, otherwise I would have searched for ages for the new location of this file).

For the DGConnection connection, the properties defined are as follows:

```
[DGConnection]
DriverName=MSSQL
SchemaOverride=% .dbo
DriverUnit=DBXDynalink
DriverPackageLoader=TDBXDynalinkDriverLoader
DriverPackage=DBXCommonDriver110.bpl
DriverAssemblyLoader=Borland.Data.TDBXDynalinkDriverLoader
DriverAssembly=Borland.Data.DbxCommonDriver,Version=11.0.5000.0,Culture=neutral,PublicKeyToken=a91a7c5705831a4f
HostName=.
DataBase=Northwind
User_Name=sa
Password=*****
BlobSize=-1
ErrorResourceFile=
LocaleCode=0000
MSSQL TransIsolation=ReadCommitted
OS Authentication=False

Prepare SQL=False
```

Note that this list includes the DriverUnit, DriverPackageLoader, DriverPackage, DriverAssemblyLoader and DriverAssembly attributes we already saw in figure C. Since these five attributes can also be found in the [MSSQL] entry inside the dbxdrivers.ini file, they are actually of no use here and can be removed!

And while we're editing dbxconnections.ini anyway, if we want to add another Delegate Driver to this connection, we have to manually add a line with key DelegateConnection and value DBXPoolConnection or DBXTraceConnection. In order to add trace capabilities (which are easier to demonstrate here than pooling capabilities), change the DGConnection section as follows:

```

[DGConnection]
DelegateConnection=DBXTraceConnection
DriverName=MSSQL
SchemaOverride=%.dbo
HostName=.
DataBase=Northwind
User_Name=sa
Password=*****
BlobSize=-1
ErrorResourceFile=
LocaleCode=0000
MSSQL TransIsolation=ReadCommitted
OS Authentication=False

Prepare SQL=False

```

Note that you have to open the Connections Editor in order to “refresh” the parameters of the TSQLConnection component. But at least we won’t have to use the Params property to modify the parameters afterwards.

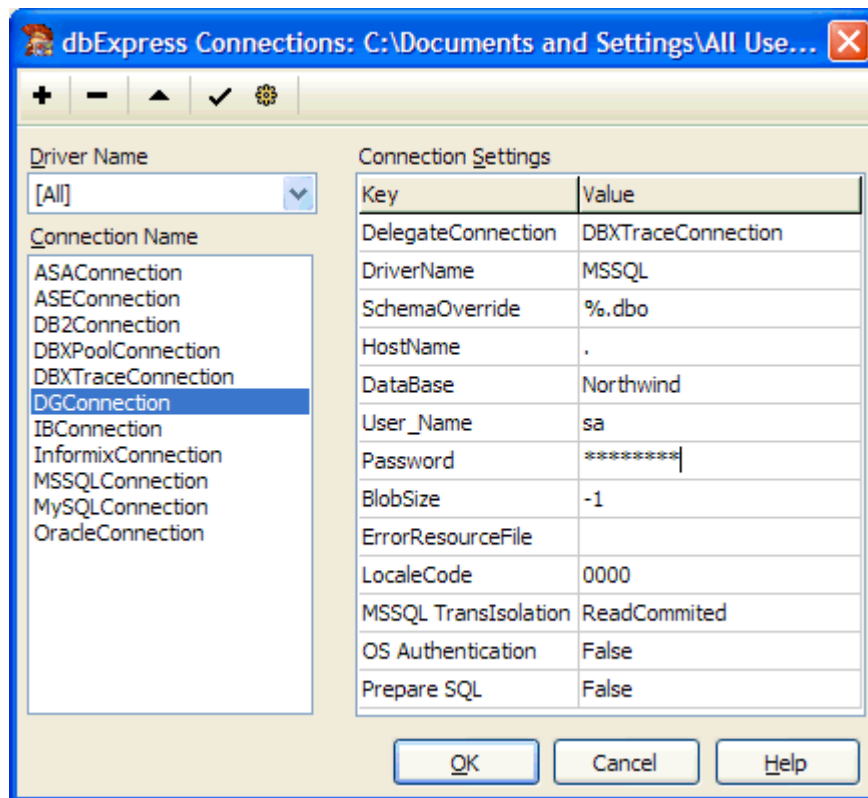


Figure E.

The dbxconnections.ini file also contains the information for the DBXPoolConnection and DBXTraceConnection. For DBXPoolConnection, the attributes that we saw earlier are defined as follows:

```

[DBXPoolConnection]
DriverName=DBXPool
MaxConnections=16
MinConnections=0
ConnectTimeout=0
For DBXTraceConnection, there are some more attributes, placed in comments, which help to get
an idea what the trace delegate driver is capable of doing.
[DBXTraceConnection]
DriverName=DBXTrace
;TraceFile=c:\temp\dbxtrace.txt
;TraceFlags=PARAMETER;ERROR;EXECUTE;COMMAND;CONNECT;TRANSACT;BLOB;MISC;VENDOR;READER;DRIVER_LOAD;METADATA
;TraceDriver=true

TraceFlags=NONE

```

If you do not specify a filename in the TraceFile attribute, then the output is shown on the console window. This will only work for a console application, or for an application that has the {\$APPTYPE CONSOLE} defined in the project file. Note that Delphi 2007 itself is not a console application, so using the DBXTraceConnection at design-time will give you I/O Error 105 errors (meaning file not open for output). Something to keep in mind when trying to open a TClientDataSet at design-time.

The output from the DBXTraceConnection delegate driver does not consists of simple log statements, but shows the actual Delphi source code that is being executed. This often has more meaning to us as Delphi developers than simple log statements that say something has been opened, prepared, etc.

```
Log Opened =====
{CONNECT      } ConnectionC1.Open;
{COMMAND      } CommandC1_1 := ConnectionC1.CreateCommand;
{COMMAND      } CommandC1_1.CommandType := 'Dbx.SQL';
{COMMAND      } CommandC1_1.RowSetSize := 20;
{COMMAND      } CommandC1_1.CommandType := 'Dbx.SQL';
{COMMAND      } CommandC1_1.Text := 'select * from Employees';
{PREPARE      } CommandC1_1.Prepare;
{COMMAND      } ReaderC1_1_1 := CommandC1_1.ExecuteQuery;
{COMMAND      } CommandC1_2 := ConnectionC1.CreateCommand;
{COMMAND      } CommandC1_2.CommandType := 'Dbx.MetaData';
{COMMAND      } CommandC1_2.Text := 'GetIndexes "Northwind"."dbo"."Employees" ` `';
{COMMAND      } ReaderC1_2_1 := CommandC1_2.ExecuteQuery;
{READER       } { ReaderC1_2_1 closed. 2 row(s) read }
{READER       } FreeAndNil(ReaderC1_2_1);
{COMMAND      } FreeAndNil(CommandC
```

If you want to see the log statements at the driver level, then you need to set the TraceDriver attribute to true. This will also include the non-Delphi log messages, and result in a larger logfile or contents of your console debug window (very handy if you have multiple monitors you can use).

Source Code Compatibility

When migrating code that uses older versions of dbExpress, you may encounter a few compatibility issues. Originally, the unit dbExpress contained the core of dbExpress, but this unit has been deprecated and instead we should use DBXCommon or SqlExpr.

Apart from the units, there are some changes in properties and methods as well. Transaction support for example. Prior to DBX4, we had to call the StartTransaction, Commit, and RollBack methods of TSQConnection. However, these methods are now deprecated. That doesn't mean that transaction support itself is deprecated (as some people incorrectly thought), but that the way to deal with transactions has changed. We should now use the new methods called BeginTransaction, CommitFreeAndNil, and RollBackFreeAndNil of the TSQConnection component instead of the deprecated methods.

```
DBXTransaction := SQLConnection1.BeginTransaction(TDBXIsolations.ReadCommitted);

try
  // do your work...
  SQLConnection1.CommitFreeAndNil(DBXTransaction);
except
  SQLConnection1.RollbackFreeAndNil(DBXTransaction);
  raise
end;
```

The source code for the DBX4 framework units can be found in C:\Program Files\CodeGear\RAD Studio\5.0\source\database.

Deployment: No More Static Linking

One downside of DBX4 is that it no longer supports static linking of drivers inside the executable. The MidasLib unit is still supported, fortunately, so we can still link the MIDAS.dll with our Win32 executables when needed. But the dbExpress drivers themselves need to be deployed.

Apart from the dbExpress driver, we also have to deploy the dbxdrivers.ini and dbxconnections.ini file when we use the DBXTraceConnection or DBXPoolConnection. The reason is that neither of these two delegate drivers are directly connected to a TSQLConnection component that holds their parameters, so the DBX4 framework has no way to know what the DBXTraceConnection consists of, for example. So far, I found no other way to let the application other than to include the dbxdrivers.ini and dbxconnections.ini files to the list of files to deploy. The good news is that you can make them almost empty, leaving only the [DBXTrace] and [DBXTraceConnection] entries in respectively the dbxdrivers.ini and dbxconnections.ini files.

Also, when working with packages, you need to deploy the DBXCommonDriver100.bpl package, as well as at least the rtl100.bpl and dbrtl100.bpl. These two additional packages are needed because the DBXCommonDriver100.bpl uses them, and this new package cannot use the older packages from Delphi 2006 with the same name (but different contents).

More Enhancements and Coverage

Apart from being placed in-between an actual database driver and your Delphi application, delegate drivers also offer the ability to have extensible commands. This means that if you build your own delegate driver, then you can add your own commands to the driver to perform special operations. This is meant for non-SQL operations, such as the “show pools” command in the DBXPool delegate driver.

Next time, I’ll cover DBX4 delegate drivers in more detail when we write our own one. This and other DBX4 techniques will also be covered at the Database Development Essentials masterclass which will be held on Monday November 26th in Upavon, right before the Web Development Essentials masterclass on Tuesday November 27th. Mark your diary if you want to be there!

References

<http://blogs.codegear.com/steveshaughnessy/archive/2007/02/16/31865.aspx>



Bob Swart b.swart@chello.nl (aka Dr Bob - www.drbob42.net) is a software developer, author, trainer, consultant and webmaster for his own one-man company, Bob Swart Training & Consultancy in Helmond, The Netherlands. He writes for numerous computing magazines as well as his own training material, and is also webmaster to the Developers Group.