

Create an ADO.NET ConnectionString

*Here's how to convert existing ADO
connection strings to work under
ADO.NET.*

You can create an ADO.NET connection string easily if you already have an ADO “classic” (ADOc) ConnectString property to start from. You don't *really* want to use the OleDb .NET provider unless you have to, and you *can't* use an ODBC connection string with OleDb (the old default), so you'll want to transmogrify your existing string to use it with the most suitable ADO.NET data provider.

When it comes time to choose an appropriate ADO.NET data provider, you should pick a “native,” “managed” provider whenever possible. A DBMS manufacturer such as Microsoft or Oracle, or a third-party vendor such as Data Direct, supplies such providers. Managed, native providers are best because they don't require COM interop transitions and deploy problematic COM DLLs.

For example, I use the built-in SqlClient data provider when accessing Microsoft SQL Server 7 and later (including MSDE and SQL Server 2000). For ODBC or any of the litany of other data sources, install the provider from the Web and add a reference to it in your application.

Now you're ready to convert your ADOc ConnectionString to ADO.NET. I'll step you through some examples, all of which are explained in detail in my new books (see Resources). Start from a typical OLE DB connection string, used by ADOc to connect to SQL Server:

```
cn.Open "provider=sqloledb;" _  
    & "data source=betav9; initial catalog=biblio", _  
    "admin", "pw"
```

This poses only a couple challenges. First, you must drop the OLE DB “provider=sqloledb;” keyword setting because you're targeting SQL Server 2000 and using the SqlClient .NET data provider. You also must incorporate the login name and password into the ConnectionString, because the ADO.NET Open method doesn't accept the security keywords. Use this new string in VB.NET:

```
Dim strConnect As String = "Data Source=betav9;" _  
    & "initial catalog=biblio;" _  
    & "uid=admin;pwd=pw"
```

Unless you're accessing ODBC data sources, you can continue to use existing ADOc ConnectionString settings when it's really necessary to use the OLE DB .NET data provider—for example, when you access Access/Jet databases or other non-ODBC data

sources lacking native providers. You *must* download and use the new Microsoft.Data.Odbc .NET data provider if you need to access ODBC data sources. See www.msdn.microsoft.com/downloads for all these new .NET data providers. There's even a beta of the Oracle .NET provider there.

If you get into trouble creating a working `ConnectionString`, you can get VS.NET to generate it for you. From your Windows Form application, open the Toolbox | Data pane and drag an `OleDbConnection` or `SqlConnection` to your form. Click on the `ConnectionString` property in the Properties window and copy the string into your application code. If necessary, build a new connection with the dialogs, choosing the appropriate OLE DB provider. This string contains lots of extraneous keywords as shown in this VS.NET-generated Jet `ConnectionString`:

```
Provider=Microsoft.Jet.OLEDB.4.0;Password="";User
ID=Admin;Data Source=C:\NEW40.MDB;Mode=Share
Deny None;Extended Properties="";Jet
OLEDB:System database="";Jet OLEDB:Registry
Path="";Jet OLEDB:Database Password="";Jet
OLEDB:Engine Type=5;Jet OLEDB:Database Locking
Mode=1;Jet OLEDB:Global Partial Bulk Ops=2;Jet
OLEDB:Global Bulk Transactions=1;Jet OLEDB:New
Database Password="";Jet OLEDB:Create System
Database=False;Jet OLEDB:Encrypt
Database=False;Jet OLEDB:Don't Copy Locale on
Compact=False;Jet OLEDB:Compact Without
Replica Repair=False;Jet OLEDB:SFP=False
```

Pretty bulky. But with a little judicious editing, you can boil it down to this—just the facts, ma'am:

```
Dim strConnect As String = _
    "Provider=Microsoft.Jet.OLEDB.4.0;Password='';_
    & "User ID=Admin;Data Source=C:\NEW40.MDB;"
```

Notice that you change the double quotes (") in the `Password=` keyword to single quotes. You can use a Universal Data Link (UDL) file to manage your `OleDb` connection string if you like. However, other providers don't support UDLs, so you must consider other ways to persist the `ConnectionString` value in cases where you can't (or don't want to) hard-code it in your app.

ODBC `ConnectionString` settings are even easier. You don't have to change anything—and yes, you can use DSNs or “DSN-less” connection strings. Frankly, when a native provider isn't available, using the Microsoft.Data.Odbc .NET data provider is always my next choice. Either of these `ConnectionString` settings will work with this new `Odbc` .NET data provider; the first uses a registered DSN (with the usual issues), while the second uses a “DSN-less” connection:

```
Dim strConnect As String = _
    & "DSN=Betav9DSN;database=biblio;" _
    & "connection timeout=2;UID=fred;pwd=xx"
Dim strConnect As String = _
    & "SERVER=betav9;Driver={Sql Server};" _
    & "UID=fred;pwd=xx;"
```

You can set only two of the ADO.NET Connection object (such as `OleDbConnection` or `SqlConnection`) properties before opening the connection—the `ConnectionString` and the `ConnectionTimeout` (which you can set in the `ConnectionString`). After the Connection is

open, the `ConnectionString` property is frozen. You can change the current “default” database post open using the `ChangeDatabase` method, but that’s about it. The remaining properties reflect `ConnectionString` settings and are read-only.

While this content is a bit dated, a more recent treatment can be found on my blog www.betav.com/blog/billva. Just search for “Connecting”.