

# MUTUALLY BENEFICIAL

## Making DTS and SSIS Packages Portable Part 1 DTS



**David Lundell, MBA**  
**Principal Consultant**  
**& Trainer**  
**MCDBA, MCSE,**  
**MCSD, MCT**

**Abstract:** One of the most significant challenges to using DTS effectively is in migrating DTS packages between dev and test, and test and production without opening the packages and modifying connection strings and other configuration items particular to each server. The main focus of this article is to show you how to make migrating/promoting DTS packages and SSIS packages from dev to test to production seamless and easy. You will see demos showing how to make DTS packages portable using Dynamic properties, Active X Script tasks, Global Variables and DTSSrun command line parameters. You will see a demonstration of how this issue is solved in SQL 2005 with SSIS package configurations.

A further thorn in one's side has to do with DTS logs being distinctly unfriendly; you will see a demonstration of how to make your DTS logs a lot easier on the eyes. Lastly you will see a demo on how SSIS logging is done and its plethora of options.

This article is based on a presentation David Lundell made to the Arizona SQL Server Users Group in Feb. 2006.

To more fully examine the subject let us imagine that you are the DBA for the conglomerate that owns Northwind Traders and Pubs (the two fictional companies that have databases included in SQL Server 2000).

### **Scenario**

You want to be the hero, helping the Pubs Sales people cross sell to the Northwind Customers. To make that happen, you had to fix a few critical issues in the DTS package that executes the Cross Sell process. Good job! Your changes have flown through testing and all is well, the change has been approved through your change control process. Now you just have to move your package into production. Go Pubs!

### **Technical Goals**

- Move your freshly fixed, tested and approved DTS Package to Production
  - Without Changing It!
  - Have it work in Production
  - Have a reproducible process
  - Able to rollback

### **Environments**

The degree of difficulty (much like in many of the Olympic Sports) is dependent on the environment in which you operate. Naturally, the ideal environment is to have **separate servers** one for development, another for testing, and an inviolate server for production, where each SQL Server is installed as the default

instance and the database names are all the same. This holds true even if dev and test SQL Servers run on virtual PCs on the same hardware. Less ideal, but less taxing on your hardware is to use **named instances**. Even less ideal but sometimes forced upon us is the notion of **named databases**. That is to say that dev and test databases are installed on the same instance of SQL Server with the databases being named differently such as pubs\_dev and pubs\_test. Heaven forbid the production database should be on the same instance, alas this has been known to happen. Nevertheless, it is still better than just making the changes to the production database directly.

**Sidebar: Disconnected Edit**  
 Disconnected Edit is to a DTS Package what the regedit is to Windows – a chance to look behind the scenes and make changes without validation. To get to disconnected edit right click on some white-space in the package and then select disconnected edit from the context menu.

## Challenge

Connection specific information gets hard coded into the package. In Figure 1 Connection Properties, by using the Disconnected Edit feature we can see that the Server Name get hard coded into the DataSource Property and the Database Name is stored in the Catalog Property.

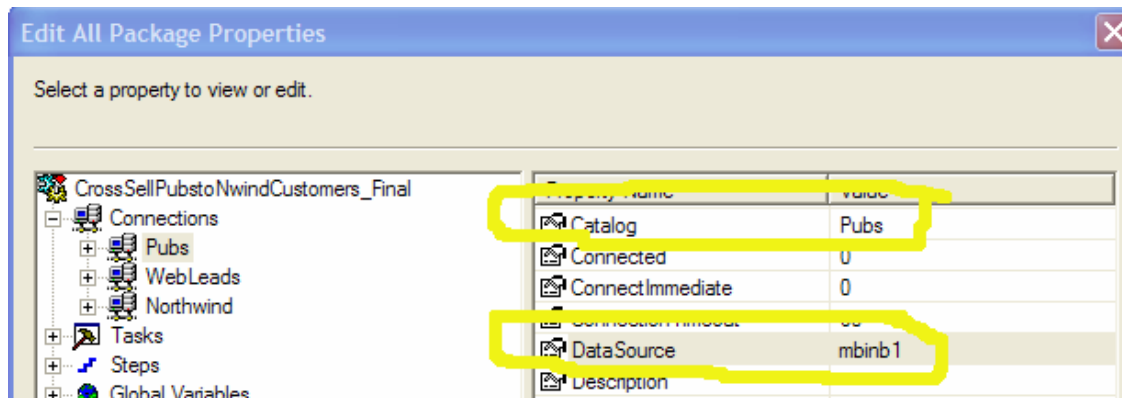


Figure 1 Connection Properties

If you are in a named databases environment (or if for some reason the databases have different names) then an even thornier problem arises. As a consequence of Data Pump tasks defaulting to use a three part name (database.owner.object i.e. pubs.dbo.authors), the database name is embedded in the DestinationObjectName and SourceObjectName properties as you can see in Figure 2 Data Pump Task Properties

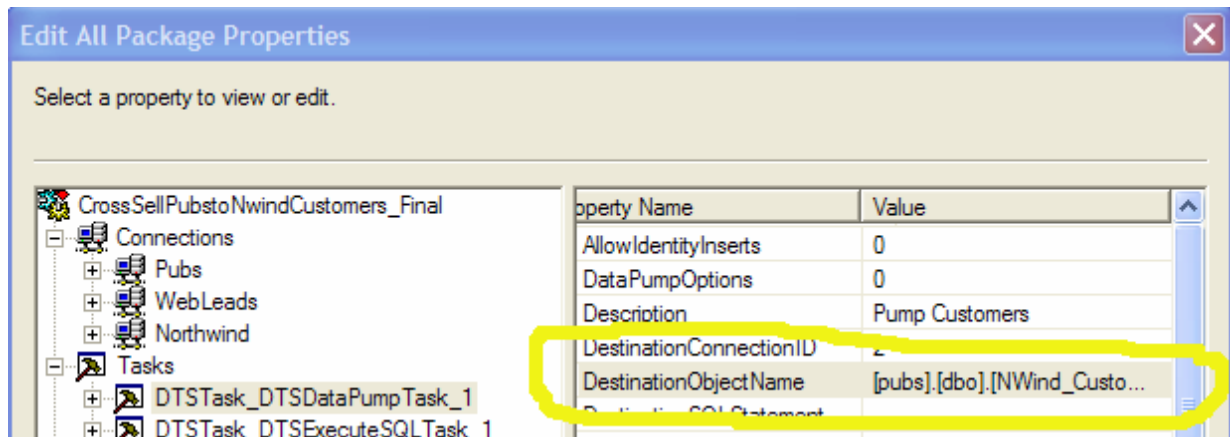


Figure 2 Data Pump Task Properties

Additionally, the Server Name is often referenced in the Package Properties to specify the logging, and path names are also specified. Hopefully, these paths exist on your other servers. (Hint: this is a best

practice to have a standard path for DTS log files I recommend c:\DTSLogs\ -- using environmental variables in the path does not work. The path %windir%\DTSLog.Log will create a file in the current working directory called %windir%\DTSLog.Log ).

## Process for Promoting Packages

- 1) You will need to make your packages portable (this is the thrust of the article and will be covered in the next section).
- 2) Use DTSBackup (a wonderful utility created by Darrell Green and available for free at sqldots.com) to copy the package to the next server (be it test or production). DTSBackup will keep the Package GUID, and the version GUID the same. Keeping the GUID the same is very useful for the next step (When you save a package to a COM Structured file it preserves its GUID, however when you save it to the other SQL Server you are given a new GUID).
- 3) Use DTSCmpare from Red Gate Software to ensure that the packages are identical
- 4) Save the DTS package as a COM structured file and check it into source control with appropriate comments. – This provides ultimate rollback capability (while the version history is kept in the msdb database as the sysdtspackages table simple adds a new row every time you hit save
- 5) Save DTS package as a VB file and check it into source control, again with comments. This gives you the ability to do a difference on the file and “see” the changes that have taken place between versions. Otherwise some of the properties are buried to deep to see.

## The Solution

If you have the ideal **separate servers** environment and your packages are only referencing the local server you can use the (local) alias in each of your connections or use the dot “.” (skip the quotes). In the **named instance** environment this will result in the use of the default instance on the box and not the named instance where the package is stored. So you will still need to use Dynamic Properties.

If you are in the **named databases** environment then you need to simplify life by using disconnected edit to remove the database name from the DestinationObjectName and SourceObjectName properties on the Data Pump Tasks as shown in Figure 3 Two Part Name. Then you only need worry about the connection objects.

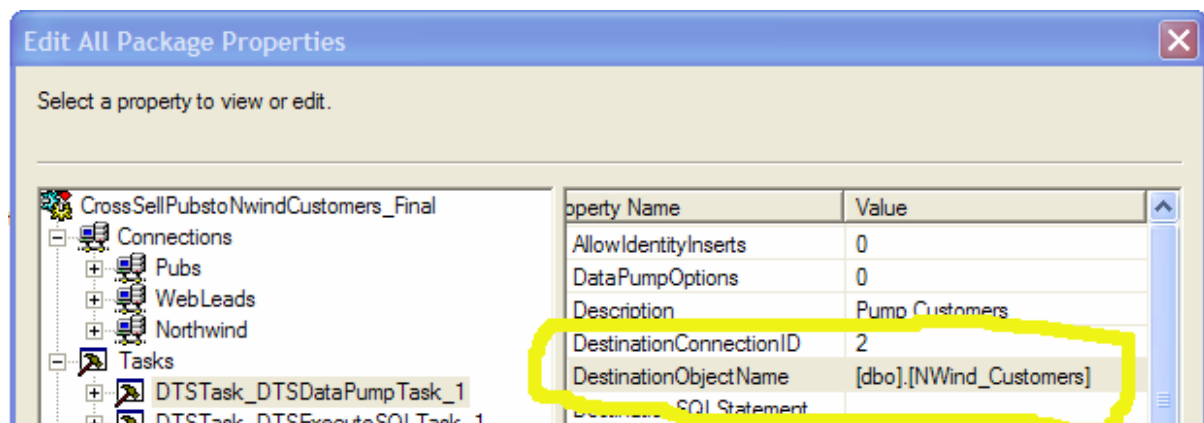


Figure 3 Two Part Name

Your connection objects can be handled using a Dynamic Properties task. Setting up the Dynamic Properties Task is very similar to using Disconnected Edit. Add a Dynamic Properties task by dragging it from the toolbox and then double click on the Disconnected Edit task (see the red 1 in Figure 4) and you get the Dynamic Properties task window as shown in Figure 4 with the red 2. When click Add or Edit you get the dialog denoted by the red 3 which bears a striking resemblance to Disconnected Edit. Then after you select the property you want to modify you get the Add/Edit Assignment dialog box shown by the red 4.



**Figure 4 Dynamic Properties**

There are several options for grabbing the value to plug into the property:

- 1) Global Variables
- 2) Ini file
- 3) Data file
- 4) Constant
- 5) Environmental Variable
- 6) Query

I recommend the Global Variables as these can be set at run time based on the command line. Ini files work great as long as you are in the **separate servers** environment, otherwise how are you going to designate which ini file is for dev and which is for test? I have not used the data file. Constants are only useful as placeholders when you are creating your package and don't yet know which method to get the properties. Environmental variables are a tempting possibility in the **separate server** environment

(otherwise what is test and what is for dev?) but under stress they do not hold up. Queries are a challenge because you have to have a working connection in order to make a query, but if your connection objects are being modified by the dynamic property how are they going to execute their queries?

**Ouch that smarts: Using Environmental Variables in Dynamic Properties**

I built some packages that had two dynamic properties – one to get the environmental variable (ComputerName), then an ActiveX Script to setup the Global variables based on the retrieved environmental variable, followed by a Dynamic Properties task to setup the connections. This worked great 97 times out of 100. However since these packages were being executed every ten minutes all day long every day that meant getting about 4 failure pages a day. I quickly returned to using Global variables and setting them from the command line.

**Code Listing 1: DTSrun.exe Global Variable at runtime**

```
Dtsrun /S Server /E /N PackageName
/A PUBSSERVER:8="MBINB1"
/A PUBSCATALOG:8="PUBS"
/L C:\DTSLogs\PackageLog.Log
```

Then to modify the variables at runtime you use DTSRun.exe The /A option allows you to specify variables at

runtime. I always use 8 to indicate string type as I have never been able to get the others to work. Additionally, Global variable names are case sensitive so please standardize. See the “Ouch that smarts” sidebar to see why.

Logging to your SQL Server is great except when you are in the **Named instance** environment. Here’s why: When you log to server the logging starts right as the package begins to execute – before your dynamic properties task begins. So in a Named Instance environment you are better off only logging to a file because that can be controlled from the command line using the /L option demonstrated in Code Listing 1.

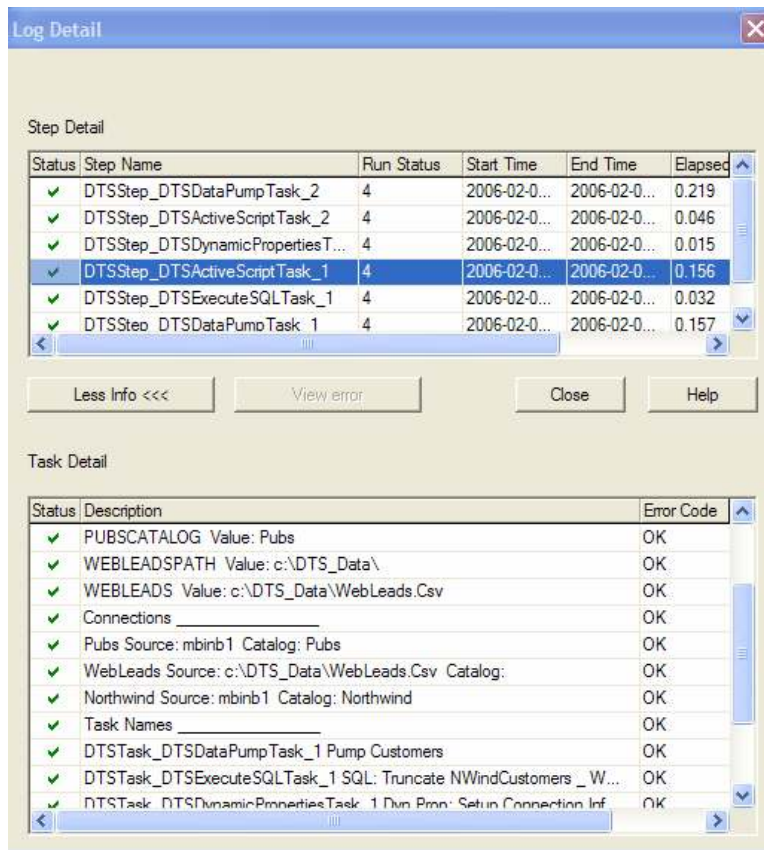
Viewing DTS logs are painful, more so the more complex your package is. By default all you get is the top portion (step detail) of what is shown in Figure 5 DTS Log. If there is an error how are you supposed to tell DTS\_Data\_Pump\_Task\_23 from DTS\_Data\_Pump\_Task\_22, which one is Pumping the customer’s addresses and which one their sales info?

Using a technique I first learned from Trey Johnson at VSLive 2003 we can make use of two methods on the DTSPackageLog object: WriteStringToLog and WriteTaskRecord. They both accomplish the same basic purpose: they write to either the log file (if logging to file) and/or to the sysdtstasklog table (if logging to server). Which one to use?

**Ouch that smarts: Global Variables are Case Sensitive – Package firing on Prod or is that Test?**

I moved the packages to Production and ran them for the first time (fortunately the system had not been made available to the users yet), and I truncated data and copied it from test.

What happened? In the package my variable was “PubsServer” and at the Command line is was “Pubsserver”. This actually created two variables (the one already defined in the package and then the one created at runtime. The Dynamic Properties task was using the one already defined in the Package which was a test value. How did I figure this out? I used some very user friendly logging – coming up next.



WriteStringToLog is better if you are logging to file as it just writes exactly what you tell it to write. When logging to the Server it puts a red X in the Status column in the task detail instead of the lovely green check mark. So when logging to Server WriteTaskRecord is better because you can control whether you get a lovely green checkmark (give a positive number for the first parameter) or whether you get an ugly red X (put a zero or a negative number for the first parameter). However when logging to file it will preface everything you write with Error = 1 (00000001), Description = "

I then use one of these two methods to write out the value of the Global variables at run time, the datasource and catalog properties of every connection object as well as the Task Names

alongside their descriptions. See Code Listing 2 or [http://www.mutuallybeneficial.com/code/DTS\\_ActiveX\\_Logging\\_Script.txt](http://www.mutuallybeneficial.com/code/DTS_ActiveX_Logging_Script.txt)

**Figure 5 DTS Log**

**Code Listing 2: VB Script for Documenting Package Variables, Connections, and Task Names**

```
'*****
' Visual Basic ActiveX Script
'*****

Function Main()
    Dim P ' Package2

    Set P = DTSGlobalVariables.Parent

    DTSPackageLog.WriteStringToLog "Global Variables _____"
    For EACH GV IN P.GlobalVariables
        'DTSPackageLog.WriteStringToLog is neater in the text
        file (it does not preface it with "Error = 1 (00000001), Description = "
        'DTSPackageLog.WriteTaskRecord 1 is neater in the SQL
        DTS Package Log -- each step shows up with a green check instead of a red X
        DTSPackageLog.WriteTaskRecord 1, ifNull(GV.NAME) & "
        Value: " & ifNull(GV.Value)
    Next

    DTSPackageLog.WriteTaskRecord 1, "Connections _____"
    For EACH cn IN P.Connections
        DTSPackageLog.WriteTaskRecord 1, cn.Name & " Source: " &
        ifNull(cn.DataSource) & " Catalog: " & ifNull(cn.Catalog)
    Next

    DTSPackageLog.WriteTaskRecord 1, "Task Names _____"
    For EACH tsk IN P.Tasks
        DTSPackageLog.WriteTaskRecord 1, tsk.Name & " " &
        tsk.Description
    Next

    Main = DTSTaskExecResult_Success
End Function

Function ifNull(test)
    IF ISNULL( test) THEN
        DTSPackageLog.WriteStringToLog "Found a null"
        ifNull = ""
    ELSE
        ifNull = test
    END IF
End Function
```

Stick around for Part II Making SSIS packages portable.

#### About the Author

David received a BS in Computer Engineering from the University of Arizona and an MBA from the Eller College of Business at the University of Arizona. He is also a member of World at Work. Furthermore, he possesses an astounding array of technical certifications from Microsoft, Cisco, Novell, Sun and CompTIA: MCSE, MCSA, MCSA, MCSD, MCDBA, MCT, CCNA, CNA, Sun Certified Programmer Java, A+, Network+, Server+. He also holds a Community College Teaching Certificate for Computer Information Systems in the State of Arizona.

David is also the author of a published text on Microsoft Windows Server 2003.

He can be reached at [David@MutuallyBeneficial.com](mailto:David@MutuallyBeneficial.com) (480) 682-7437