**DTUtil.exe /?**

Essentially, this tool can be used to do many of the things that you do in Management Studio, and to a lesser extent in [SQL Server Data Tools](https://mindmajix.com/ssis/sql-server-data-tools-for-visual-studio-2013). The next sections describe creative ways to use DTUtil.exe.

**Re-encrypting All Packages in a Directory**

By default, SSIS files in development are encrypted to prevent an unauthorized person from seeing your SSIS package. The type of encryption is seamless behind the scenes and is applied at a workstation and user level.

Earlier in development you can set the ProtectionLevel property to EncryptSensitiveWithUserKey (default option) to lock down password information in Connection Managers and other sensitive data. You can also set a password on the package by changing the ProtectionLevel property to EncryptSensitiveWithPassword.

By default, if you were to send a package that you’re developing to another developer on your team, he or she would not be able to open it. The same would apply if you logged in with a different user account. You would receive the following error:

**There were errors while the package was being loaded. The package**  
**might be  
corrupted. See the Error List for details.**

This error message is very misleading. In truth, you can’t open the package, because the originating user encrypted it, whether intentionally or not. To fix this, the owner of the package can open it and select a different option in the Properties pane (like a package password) for the ProtectionLevel option. The default option is EncryptSensitiveWithUserKey. To protect the entire package with a password, select the EncryptAllWithPassword option.

Another useful option enables SSIS designers to encrypt all packages with the default option, and when it is time to send them to production, they can develop a batch file to loop through a directory’s .**dtsx** file and set a password. The batch file would use DTUtil.exe and look like this:

**for %%f in (\*.dtsx) do Dtutil.exe /file %%f /encrypt  
file;%%f;3;newpassword**

This would loop through each .dtsx file in your directory and assign the password of newpassword. The production support group could then use the same batch file to reset the password to a production password. The number 3 before the word newpassword sets the ProtectionLevel property of the package to EncryptAllWithPassword.

**Handling a Corrupt Package**

Occasionally when you copy objects in and out of a container, you may corrupt a given task in the package. In that case, you can’t delete the task or move it outside the container or link it in the container. This doesn’t happen often, but when you suspect you have a corrupt package or object, you can use DTUtil.exe to regenerate the package’s XML. To do so, you can use the –I switch to generate a new PackageID and regenerate the XML, like this:

**DTUtil.exe -I -File dbsnapshots.dtsx**

After you do this, the package may look different when you open it because the XML has been regenerated. For example, some of your containers may be smaller than they were originally and placed in areas they weren’t originally located. You can also use this command to generate a new PackageID when you find that the developer has copied and pasted the package in SQL Server Data Tools.

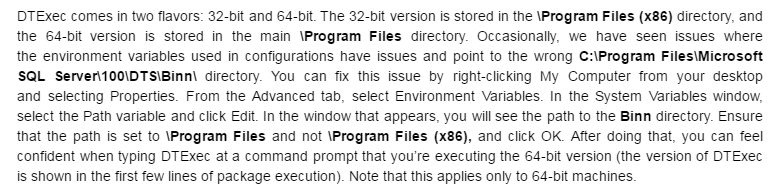
You can also create a batch file to loop through the directory and regenerate the ID for every package in the directory. The batch file will loop through every .dtsx file and execute DTUtil.exe. The **batch file looks like this:**

**for %%f in (\*.dtsx) do dtutil.exe /I /FILE “%%f”**

**Case**  
When you copy a SSIS package, the GUID that identifies the package, remains the same. So far no problem, but when you enable logging this value is logged as the SourceID. If you see a package error in your log and a couple of packages have the same GUID, it's hard to see which package caused the error. How do I change duplicate GUIDS?

As mentioned before, DTExecUI.exe is a 32-bit application. Therefore, whenever you execute a package from DTExecUI.exe, it will execute in 32-bit mode and potentially take longer to execute than if you were executing it on your development machine. Much of the reason for this is data must be marshaled back and forth between 32-bit mode and 64-bit mode, and the amount of memory available differs. Also, Visual Studio is a 32-bit process, so you will need to run your packages out of the Visual Studio debug mode to get a true 64-bit run. To get around this problem, you can go to the [Command Line](https://mindmajix.com/ssis/command-line-utilities) page of this tool, copy the command out of the window, and paste it into a command prompt, prefixing **dtexec.exe** in front of it.

Learn how to use SSIS, from beginner basics to advanced techniques. Enroll for Free [SSIS Training](https://mindmajix.com/ssis-training) Demo!



A particularly annoying quirk (at the time of publication) is the lack of an MDAC driver for the 64-bit architecture. The impact of this is that you can’t execute packages in 64-bit mode if they refer to anything that uses a Jet driver, in particular, Access and Excel. If you need to do this, you can execute the package using the 32-bit version of DTExec.exe. Another option in [SQL Server Data Tools](https://mindmajix.com/ssis/sql-server-data-tools-for-visual-studio-2013) is to right-click the project, select Properties, and set the Run 64Bit Runtime to false in the Debugging page. This will set packages inside the project to run in 32-bit mode when debugging.

Aside from the few quirks of the 64-bit architecture and SSIS, the benefits are incredible. Keep in mind that SSIS is very memory intensive. If you’re able to scale up the memory on demand with a 64-bit architecture, you have a truly compelling reason to upgrade. Even though tools like DTExecUI are not 64- bit ready, scheduled packages will run under 64-bit mode. If you want a package to run under 32-bit mode, you have to add the step to run the 32-bit DTExec from the scheduled job by going to the runtime option in the Execution Options tab in SQL Server Agent.