Working with files and the FileSystemObject

By Darren Green
Version 7.0/2000
Level Intermediate

The ability to work with files gives us great flexibility in how we use DTS. It allows us to add some intelligence into a package, and cope with unexpected situations automatically. The key to working with files is the Scripting Run Time as this gives us the FileSystemObject, which can be fully exploited through ActiveX Script Tasks, Workflow scripts, and even transformations, inside your DTS packages. We start by demonstrating some simple file manipulation procedures, and then move on to how we can fully integrate them into a DTS package and make our solutions more flexible.

Full documentation from Microsoft on the FileSystemObject object and related objects can be found here

Copy File

This shows a simple file copy operation, using hardcoded source and destination filenames.

```
' Copy File
Option Explicit

Function Main()

          Dim oFSO
          Dim sSourceFile
          Dim sDestinationFile

          Set oFSO = CreateObject("Scripting.FileSystemObject")

          sSourceFile = "C:\SourceFile.txt"
          sDestinationFile = "C:\DestinationFile.txt"

          oFSO.CopyFile sSourceFile, sDestinationFile

          ' Clean Up
          Set oFSO = Nothing

          Main = DTSTaskExecResult_Success
End Function
```

Move or Rename File

This shows a simple file move operation, using hardcoded source and destination filenames. There is no explicit rename method in the scripting object, but a move is just the same.

```
' Move File
Option Explicit

Function Main()

          Dim oFSO
          Dim sSourceFile
          Dim sDestinationFile

          Set oFSO = CreateObject("Scripting.FileSystemObject")

          sSourceFile = "C:\SourceFile.txt"
          sDestinationFile = "C:\Folder\DestinationFile.txt"
```

```
          oFSO.MoveFile sSourceFile, sDestinationFile

          ' Clean Up
          Set oFSO = Nothing

          Main = DTSTaskExecResult_Success
End Function
```

Delete File

This shows a slightly more advanced delete operation. When using the DeleteFile method, if the file does not exist, an error is raised (File not found). We use the FileExists method to check for the file, and only call the DeleteFile when required. For another example of using FileExists see the article How can I check if a file exists?

```
' Delete File
Option Explicit

Function Main()

          Dim oFSO
          Dim sSourceFile

          Set oFSO = CreateObject("Scripting.FileSystemObject")

          sSourceFile = "C:\SourceFile.txt"

          ' Check if file exists to prevent error
          If oFSO.FileExists(sSourceFile) Then
                    oFSO.DeleteFile sSourceFile
          End If

          ' Clean Up
          Set oFSO = Nothing

          Main = DTSTaskExecResult_Success
End Function
```

Using Global Variables

To try and make our packages more flexible we should avoid using hard coded filenames as much as possible. One good method of doing this is to store all parameters required in global variables. In this way, no matter how many times we use a filename or file path inside our package, we only have to make one change. This sample is a move operation again, and uses two global variables. The source filename is held in one global variable, and the destination path or folder is held in a second. We are only supplying the destination folder in this example, so the existing filename will be preserved. It is important that the path is qualified with a backslash, so we use a helper function QualifyPath to ensure this is the case.

```
' Global Variable Move
Option Explicit

Function Main()

          Dim oFSO
          Dim sSourceFile
          Dim sDestinationFile

          Set oFSO = CreateObject("Scripting.FileSystemObject")

          sSourceFile = DTSGlobalVariables("SourceFileName").Value
          sDestinationFile = QualifyPath(DTSGlobalVariables("DestinationPath").Value)
```

```
          MsgBox sSourceFile
          MsgBox sDestinationFile
          oFSO.MoveFile sSourceFile, sDestinationFile

          ' Clean Up
          Set oFSO = Nothing

          Main = DTSTaskExecResult_Success
End Function

Function QualifyPath(ByVal sPath)
          If Right(sPath, 1) = "\" Then
                    QualifyPath = sPath
          Else
                    QualifyPath = sPath & "\"
          End If
End Function
```

File Date & Time

A simple validation check we may wish to do would be to check the data and time of a file to ensure it has been created within a defined time period. In this example we check that the file has been created today, otherwise we return a failure result from the task. In some case you may not want to fail the task, which in turn leads to a failed package result, in which case the code should be placed inside a Workflow script instead. For an example of using Workflow scripts to avoid failure conditions, see the article How can I check if a file exists?

```
' File Date & Time
Option Explicit

Function Main()

          Dim oFSO
          Dim oFile
          Dim sSourceFile

          Set oFSO = CreateObject("Scripting.FileSystemObject")

          sSourceFile = DTSGlobalVariables("SourceFileName").Value

          Set oFile = oFSO.GetFile(sSourceFile)

          If oFile.DateCreated < Date Then
                    Main = DTSTaskExecResult_Success
          Else
                    Main = DTSTaskExecResult_Failure
          End If

          ' Clean Up
          Set oFile = Nothing
          Set oFSO = Nothing
End Function
```

File Size

Another simple validation check similar to the date check shown above uses the Size property. In this example we will return an error if the file is empty or 0 bytes in size.

```
' File Size
Option Explicit

Function Main()
```

```
        Dim oFSO
        Dim oFile
        Dim sSourceFile

        Set oFSO = CreateObject("Scripting.FileSystemObject")

        sSourceFile = DTSGlobalVariables("SourceFileName").Value

        Set oFile = oFSO.GetFile(sSourceFile)

        If oFile.Size > 0 Then
                Main = DTSTaskExecResult_Success
        Else
                Main = DTSTaskExecResult_Failure
        End If

        ' Clean Up
        Set oFile = Nothing
        Set oFSO = Nothing
End Function
```

Using Connections

The majority of your file manipulations will be related to files used by connections within your DTS package. A common requirement is to be import a file of a changing name, for which many people would first think of using the move or rename method shown above, but why not just change the connection instead? You can see how to do this in the article How can I change the filename for a text file connection?

The obvious integration of connections into your file manipulations is to read or set a filename on a connection, and use that within the rest of your script. Connections have a DataSource property, and for Text File connections and Jet (Excel, Access) connections, this property is the filename. For DBase and FoxPro connections then DataSource is the folder, the filename or table will be selected in the task that uses the connection.

In this simple example we will read the filename from a connection and then rename the file, such as you may want to do after importing a file to indicate that the file has been dealt with. To get reference to the connection object we use the name, this is what you see as the label in the designer and it is also shown in the "Existing connection" drop-down in the connection properties dialog.

```
' Rename File from Connection
Option Explicit

Function Main()

        Dim oPkg
        Dim oConn
        Dim oFSO
        Dim oFile
        Dim sFilename

        ' Get reference to the current Package object
        Set oPkg = DTSGlobalVariables.Parent

        ' Get reference to the named connection
        Set oConn = oPkg.Connections("Text File (Source)")

        ' Get the filename from the connection
        sFilename = oConn.DataSource

        ' Rename the file
        Set oFSO = CreateObject("Scripting.FileSystemObject")
        oFSO.MoveFile sFilename, sFilename & ".bak"
```

```
            ' Clean Up
            Set oConn = Nothing
            Set oPkg = Nothing
            Set oFSO = Nothing

            Main = DTSTaskExecResult_Success
End Function
```