Rename and Move a File with PowerShell in a SQL Server Agent Job

Written By: Tim Smith -- 4/21/2014

Problem

We import a file once a day and after importing the file, we need it moved to an archive folder with a date stamp appended to the file name. We have had trouble in the past with this process, so we'd like this to run as its own step to help troubleshoot it if there are issues.

Solution

We can find a multitude of solutions for this problem, but the key is to make it its own process so that with process of elimination we can determine what part of our job failed. PowerShell makes it easy to complete a task like this because the jobs steps in SQL Server Agent allow us to use PowerShell.

PowerShell code to Move and Append Date to File Name

First, we want to obtain today's date. In PowerShell, the command below will obtain it and store it in the variable $d:

```
$d = Get-Date -uFormat "%Y%m%d"
```

Second, we want to rename the current file and append the date to the filename. For testing purposes, let's create a text file called "test" and put it on the "C" drive under the folder "files" and finally create a folder called "Archive" within the "files" folder. To make this into a function later, we'll break this up piece-by-piece, creating separate variables that will later become parameters. Let's create the variable $location, which will store the drive letter and folder location of the file, $file, which will store the file name, and $extension, which will store the extension. From there, let's create two new variables $old, which will hold the full file path of the file, and $new which will hold the full file path of the file after the date has been appended to the file name. Note that $new adds an underscore and the date ("_" + $d):

```
$date = Get-Date -uFormat "%Y%m%d"
## These will become parameters in our function later
$locationPath = "C:\files\"
$fileName = "test"
$extension = ".txt"
$old = $locationPath + $fileName + $extension
$new = $locationPath + $fileName + "_" + $date + $extension
Rename-Item $old $new
```

The result is that "C:\files\test.txt" becomes "C:\files\test_03242013.txt" (or whatever today's date is when you're reading this). Now we can add another variable ($archive), which will later become a parameter for our function, and another command (Move-Item) to archive the file:

```
$date = Get-Date -uFormat "%Y%m%d"
$locationPath = "C:\files\"
$fileName = "test"
$extension = ".txt"
$old = $locationPath + $fileName + $extension
$new = $locationPath + $fileName + "_" + $date + $extension
$archive = $locationPath + "Archive\"
Rename-Item $old $new
```

```
Move-Item $new $archive
```

The variable $archive adds the folder "Archive\" to the path "C:\files", and the Move-Item command moves the file. The final step is replacing this with a function, where we can re-use it for other files and job steps:

```
Function RenameMoveFile($locationPath, $fileName, $extension, $archiveFolder)
{
    $date = Get-Date -uFormat "%Y%m%d"
    $old = $locationPath + $fileName + $extension
    $new = $locationPath + $fileName + "_" + $date + $extension
    $archiveFolder = $locationPath + $archiveFolder + "\"
    Rename-Item $old $new
    Move-Item $new $archiveFolder
}
## This is the only part that we'd edit
RenameMoveFile -locationPath "C:\files\" -fileName "test" -extension ".txt" -archiveFolder "Archive"
```

As an final note, we never need to re-write this function again. We can now add it to a job step in SQL Server Agent, editing only the parameters as shown above in the last line of the code.

Summary

Although my purpose for writing this was to rename and move files for an import process, this can also be used for any number of things like moving and renaming backup files, moving and renaming export files, etc. I hope this little script comes in handy.

Next Steps

- Test the PowerShell script locally or in a development environment.
- Ensure that your SQL Server Agent account has the appropriate permissions to execute PowerShell scripts.