|  |  |  |
| --- | --- | --- |
| SharePoint 2010 Data Cleansing Suit v1.0 | 2011 | |
| This document contains all instructions and usage outlines for the SharePoint 2010 Data Cleansing Suite | |  |



Contents

[SharePoint 2010 Data Cleansing Suite 2](#_Toc294102043)

[Disclaimer 2](#_Toc294102044)

[Intended Usage 2](#_Toc294102045)

[How the Scripts works – What they actually do… 2](#_Toc294102046)

[How the scripts are intended to be used 3](#_Toc294102047)

[GeneralNotes 3](#_Toc294102048)

[Perquisitesbeforerunningthescript 4](#_Toc294102049)

[Running the Reporting Scripts 5](#_Toc294102050)

[Running the Change Illegal Characters Script 7](#_Toc294102051)

[Annex 1 – The Scripts 9](#_Toc294102052)

[Annex 1a - Check Default Blocked File Types: 9](#_Toc294102053)

[Annex 1b - Check Illegal File Names 12](#_Toc294102054)

[Annex 1c - Check Illegal File Names and Blocked File Types 15](#_Toc294102055)

[Annex 1d - Change Illegal Characters 18](#_Toc294102056)

[Annex B – Blocked File Types 24](#_Toc294102057)

# SharePoint 2010 Data Cleansing Suite

# Disclaimer

These tools are provided as is and are used and implemented at the users own risk. You may use this tool for any purpose as long as any communication/feedback accredits Heath Groves of [www.SundownSolutions.co.uk](http://www.SundownSolutions.co.uk) with the original source code.

# Intended Usage

These tools are designed to be used for:

1. Scanning File Shares to Discover:
   * + - *Illegal File Names*
       - *Illegal File Types*
2. Scanning Local Drives to Discover:
   * + - *Illegal File Names*
       - *Illegal File Types*
3. Scanning local and shared drives and replacing all illegal characters with characters of your choosing

# How the Scripts works – What they actually do…

Reporting Scripts

The reporting scripts are used as an aid to planning so you can see the size and severity of the data cleansing task.

Check Default Blocked File Types

This script starts by scanning your selected location and then comparing the file extensions against the default list of SharePoint blocked file types. The Default List of Blocked File Types can be Found at the end of this document in Annex B – Blocked File Types.

If you have custom blocked file types then these can be added or removed from the script as you require.

It then outputs a txt file to c:\SundownSolutions

Check Illegal Filenames

This script starts by scanning your selected location and the comparing the file names against the default illegal characters in SharePoint which are: **{ } & % ~ #**

It then outputs a txt file to c:\SundownSolutions

Check Illegal Filenames and Blocked File Types

This report combines the blocked file type check and the illegal character check together and produces one report stored in c:\SundownSolutions

Change Illegal Characters

This script runs against your selected drive and scans all the file names for illegal characters – it then gives you the option to replace each character with a known good value of you’re choosing.

The script replaces the characters with your selection and outputs to text files located here c:\Sundown Solutions\RenamedFiles

Each Character that is renamed receives its own log file so you can see which characters have been replaced in a more granular fashion

# How the scripts are intended to be used

It is envisioned that you will use the scripts multiple times during a data cleansing operation to help you identify and understand how “ready” your data is to be moved to a SharePoint environment.

First all you would scan the drive in question using the Reporting scripts and use the log files to identify how many problem filenames and blocked file types exist.

Then you should run the change illegal Characters script – this will clean all illegal characters accordingly.

You would then re-run the reporting scripts which should help you identify if any problems still exist.

If the log files are empty then no problems exists! ☺ repeat as necessary, your mileage may vary!

# GeneralNotes

This script is version one and as such it is to be modified and updated accordingly with increasing functionality – feel free to add and rationalise the code as necessary.

All code here has been written “long hand” to allow the maximum amount of customisation.

There are two types of illegal filename that are not renamed these are “..” and “…” – any file name that has multiple periods side by side i.e. .. or … will not be renamed – this is due to the script also including the .txt (File extension period) in its scan. The reporting scripts will allow you to identify these files though.

It takes approximately 15 minutes to scan 90 GB of data – this is a rough guide.

The bigger the data the longer the scan will take.

# Perquisitesbeforerunningthescript

You must be an administrator to the run the script and your PowerShell environment must be configured to allow remote scripts to run – this can be achieved by running the command:

set-executionpolicy remotesigned

\*see this link for details:

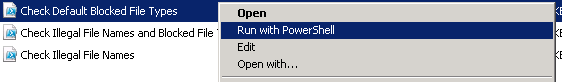
<http://technet.microsoft.com/en-us/library/dd347628.aspx>

Please ensure that you run the script from a server – this will guarantee that PowerShell is enabled.

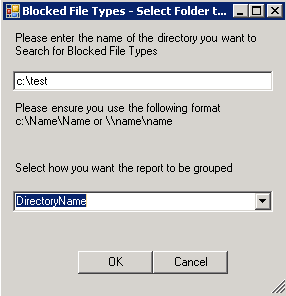
# Running the Reporting Scripts

There are two ways to run the script, directly from the command shell, or using the Integrated Scripting Environment (ISE). This guide will show you how to use the shell version but using the ISE is recommended if you require trouble shooting information.

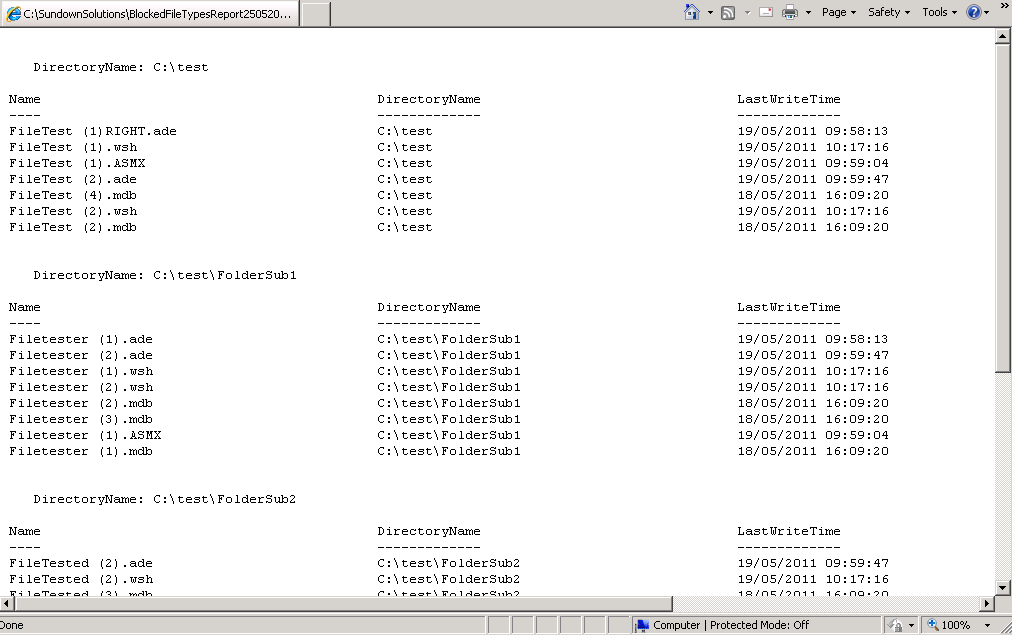
1. Copy the script to your server – the desktop is fine. Right click the script and select **Run with PowerShell**



1. You will be presented with the first dialog box – enter the path to the drive you wish to scan and also enter how you wish to group the report



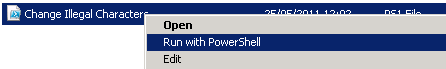
1. You will be presented with the report – this report will also be stored in c:\sundownsolutions\



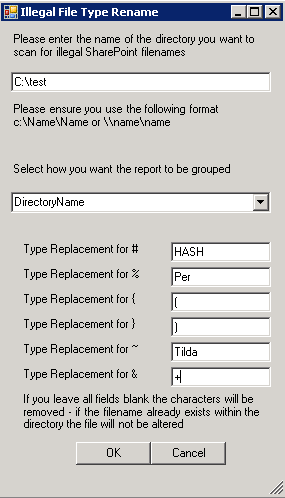
1. Repeat this procedure for all reporting scripts as necessary

# Running the Change Illegal Characters Script

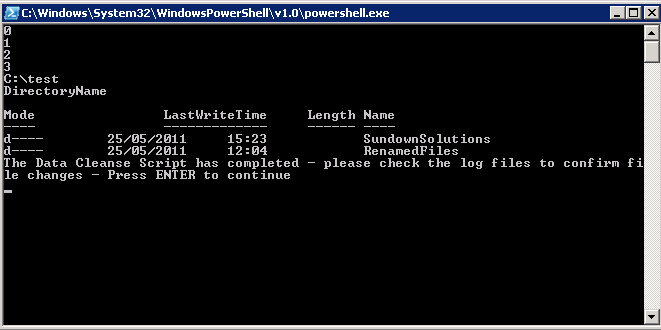
1. Copy the script to your server – the desktop is fine. Right click the script and select **Run with PowerShell**



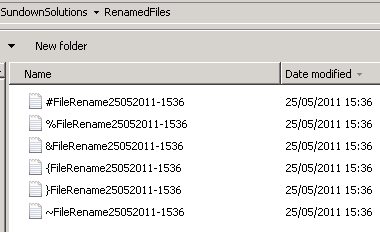
1. Next, enter the directory you wish to scan, how you want the reports to be group and your corresponding replacement characters



1. After a period of time (Depending on the size of the drive that is being scanned) the script will stop at the command screen waiting for you to press Enter to continue – this is to aid troubleshooting in case anything goes wrong. Press enter to close the PowerShell window



1. The folder containing all the log files will be displayed.



# Annex 1 – The Scripts

# **Annex 1a - Check Default Blocked File Types:**

#----------------------------------------------------------------------------------------------------Dialog Box 1 - Folder To be Searched

[void] [System.Reflection.Assembly]::LoadWithPartialName("System.Drawing")

[void] [System.Reflection.Assembly]::LoadWithPartialName("System.Windows.Forms")

[array]$DropDownArray = "DirectoryName","Name","Length","LastWriteTime" #----------------------------The Group By Array

#----------------------------------------------------------------------------------------------------This Function Returns the Selected Value and Closes the Form

function Return-DropDown {

$Choice = $DropDown.SelectedItem.ToString()

$Form.Close()

Write-Host $Choice

}

$objForm = New-Object System.Windows.Forms.Form

$objForm.Text = "Blocked File Types - Select Folder to Check"

$objForm.Size = New-Object System.Drawing.Size(290,300)

$objForm.StartPosition = "CenterScreen"

$objForm.KeyPreview = $True

$objForm.Add\_KeyDown({if ($\_.KeyCode -eq "Enter")

{$x=$objTextBox.Text;$objForm.Close()}})

$objForm.Add\_KeyDown({if ($\_.KeyCode -eq "Escape")

{$objForm.Close()}})

$DropDownLabel = new-object System.Windows.Forms.Label

$DropDownLabel.Location = new-object System.Drawing.Size(10,140)

$DropDownLabel.size = new-object System.Drawing.Size(280,30)

$DropDownLabel.Text = "Select how you want the report to be grouped"

$objform.Controls.Add($DropDownLabel)

$DropDown = new-object System.Windows.Forms.ComboBox

$DropDown.Location = new-object System.Drawing.Size(10,170)

$DropDown.Size = new-object System.Drawing.Size(260,20)

ForEach ($Item in $DropDownArray) {

$DropDown.Items.Add($Item)

}

$objform.Controls.Add($DropDown)

$OKButton = New-Object System.Windows.Forms.Button

$OKButton.Location = New-Object System.Drawing.Size(75,230)

$OKButton.Size = New-Object System.Drawing.Size(75,23)

$OKButton.Text = "OK"

$OKButton.Add\_Click({$DirectoryToSearch=$objTextBox.Text;$GroupBy=$DropDown.Text;$objForm.Close()})

$objForm.Controls.Add($OKButton)

$CancelButton = New-Object System.Windows.Forms.Button

$CancelButton.Location = New-Object System.Drawing.Size(150,230)

$CancelButton.Size = New-Object System.Drawing.Size(75,23)

$CancelButton.Text = "Cancel"

$CancelButton.Add\_Click({$objForm.Close()})

$objForm.Controls.Add($CancelButton)

$objLabel = New-Object System.Windows.Forms.Label

$objLabel.Location = New-Object System.Drawing.Size(10,10)

$objLabel.Size = New-Object System.Drawing.Size(280,40)

$objLabel.Text = "Please enter the name of the directory you want to Search for Blocked File Types"

$objForm.Controls.Add($objLabel)

$objLabel1 = New-Object System.Windows.Forms.Label

$objLabel1.Location = New-Object System.Drawing.Size(10,80)

$objLabel1.Size = New-Object System.Drawing.Size(280,40)

$objLabel1.Text = "Please ensure you use the following format c:\Name\Name or \\name\name"

$objForm.Controls.Add($objLabel1)

$objTextBox = New-Object System.Windows.Forms.TextBox

$objTextBox.Location = New-Object System.Drawing.Size(10,50)

$objTextBox.Size = New-Object System.Drawing.Size(260,20)

$objForm.Controls.Add($objTextBox)

$objForm.Topmost = $True

$objForm.Add\_Shown({$objForm.Activate()})

[void] $objForm.ShowDialog()

$DirectoryToSearch

$GroupBy

#----------------------------------------------------------------------------------------------------Create Output Folder and Date Stamped File name

[IO.Directory]::CreateDirectory("c:\SundownSolutions")

$Date = Get-Date

$Filename="BlockedFileTypesReport{0:d2}{1:d2}{2:d2}-{3:d2}{4:d2}.txt" -f $date.day,$date.month,$date.year,$date.hour,$date.minute

$Filename2 = "c:\SundownSolutions\" + $Filename

#----------------------------------------------------------------------------------------------------Conduct The Search and Output the File

$File = Get-ChildItem $DirectoryToSearch -include \*.ade,\*.adp,\*.app,\*.asa,\*.ashx,\*.asmx,\*.asp,\*.bas,\*.bat,\*.cdx,\*.cer,\*.chm,\*.class,\*.cmd,\*.cnt,\*.com,\*.config,\*.cpl,\*.crt,\*.csh,\*.der,\*.dll,\*.exe,\*.fxp,\*.gadget,\*.grp,\*.hlp,\*.hpj,\*.hta,\*.htr,\*.htw,\*.ida,\*.idc,\*.idq,\*.ins,\*.isp,\*.its,\*.jse,\*.ksh,\*.lnk,\*.mad,\*.maf,\*.mag,\*.mam,\*.maq,\*.mar,\*.mas,\*.mat,\*.mau,\*.mav,\*.maw,\*.mcf,\*.mda,\*.mdb,\*.mde,\*.mdt,\*.mdw,\*.mdz,\*.msc,\*.msh,\*.msh1,\*.msh1xml,\*.msh2,\*.msh2xml,\*.mshxml,\*.msi,\*.msp,\*.mst,\*.ops,\*.pcd,\*.pif,\*.pl,\*.prf,\*.prg,\*.printer,\*.ps1,\*.ps1xml,\*.ps2,\*.ps2xml,\*.psc1,\*.psc2,\*.pst,\*.reg,\*.rem,\*.scf,\*.scr,\*.sct,\*.shb,\*.shs,\*.shtm,\*.shtml,\*.soap,\*.stm,\*.svc,\*.url,\*.vb,\*.vbe,\*.vbs,\*.ws,\*.wsc,\*.wsf,\*.wsh -recurse | Sort-Object -property $GroupBy | format-table -property Name, DirectoryName, LastWriteTime, Length -groupby $GroupBy | Out-File $Filename2 -width 180

$File

#----------------------------------------------------------------------------------------------------Open File in IE

$ie = new-object -comobject "InternetExplorer.Application"

$ie.visible = $true

$ie.navigate($Filename2)

#----------------------------------------------------------------------------------------------------Dispose of all variables - Cleaning up :)

function Dispose-All {

Get-Variable -exclude Runspace |

Where-Object {

$\_.Value -is [System.IDisposable]

} |

Foreach-Object {

$\_.Value.Dispose()

Remove-Variable $\_.Name

}

}

# **Annex 1b - Check Illegal File Names**

#----------------------------------------------------------------------------------------------------Dialog Box 1 - Folder To be Searched

[void] [System.Reflection.Assembly]::LoadWithPartialName("System.Drawing")

[void] [System.Reflection.Assembly]::LoadWithPartialName("System.Windows.Forms")

[array]$DropDownArray = "DirectoryName","Name","Length","LastWriteTime" #----------------------------The Group By Array

#----------------------------------------------------------------------------------------------------This Function Returns the Selected Value and Closes the Form

function Return-DropDown {

$Choice = $DropDown.SelectedItem.ToString()

$Form.Close()

Write-Host $Choice

}

$objForm = New-Object System.Windows.Forms.Form

$objForm.Text = "Illegal File Types - Select Folder to Check"

$objForm.Size = New-Object System.Drawing.Size(290,300)

$objForm.StartPosition = "CenterScreen"

$objForm.KeyPreview = $True

$objForm.Add\_KeyDown({if ($\_.KeyCode -eq "Enter")

{$x=$objTextBox.Text;$objForm.Close()}})

$objForm.Add\_KeyDown({if ($\_.KeyCode -eq "Escape")

{$objForm.Close()}})

$DropDownLabel = new-object System.Windows.Forms.Label

$DropDownLabel.Location = new-object System.Drawing.Size(10,140)

$DropDownLabel.size = new-object System.Drawing.Size(280,30)

$DropDownLabel.Text = "Select how you want the report to be grouped"

$objform.Controls.Add($DropDownLabel)

$DropDown = new-object System.Windows.Forms.ComboBox

$DropDown.Location = new-object System.Drawing.Size(10,170)

$DropDown.Size = new-object System.Drawing.Size(260,20)

ForEach ($Item in $DropDownArray) {

$DropDown.Items.Add($Item)

}

$objform.Controls.Add($DropDown)

$OKButton = New-Object System.Windows.Forms.Button

$OKButton.Location = New-Object System.Drawing.Size(75,230)

$OKButton.Size = New-Object System.Drawing.Size(75,23)

$OKButton.Text = "OK"

$OKButton.Add\_Click({$DirectoryToSearch=$objTextBox.Text;$GroupBy=$DropDown.Text;$objForm.Close()})

$objForm.Controls.Add($OKButton)

$CancelButton = New-Object System.Windows.Forms.Button

$CancelButton.Location = New-Object System.Drawing.Size(150,230)

$CancelButton.Size = New-Object System.Drawing.Size(75,23)

$CancelButton.Text = "Cancel"

$CancelButton.Add\_Click({$objForm.Close()})

$objForm.Controls.Add($CancelButton)

$objLabel = New-Object System.Windows.Forms.Label

$objLabel.Location = New-Object System.Drawing.Size(10,10)

$objLabel.Size = New-Object System.Drawing.Size(280,40)

$objLabel.Text = "Please enter the name of the directory you want to Search for illegal SharePoint Filenames"

$objForm.Controls.Add($objLabel)

$objLabel1 = New-Object System.Windows.Forms.Label

$objLabel1.Location = New-Object System.Drawing.Size(10,80)

$objLabel1.Size = New-Object System.Drawing.Size(280,40)

$objLabel1.Text = "Please ensure you use the following format c:\Name\Name or \\name\name"

$objForm.Controls.Add($objLabel1)

$objTextBox = New-Object System.Windows.Forms.TextBox

$objTextBox.Location = New-Object System.Drawing.Size(10,50)

$objTextBox.Size = New-Object System.Drawing.Size(260,20)

$objForm.Controls.Add($objTextBox)

$objForm.Topmost = $True

$objForm.Add\_Shown({$objForm.Activate()})

[void] $objForm.ShowDialog()

$DirectoryToSearch

$GroupBy

#----------------------------------------------------------------------------------------------------Create Output Folder and Date Stamped File name

[IO.Directory]::CreateDirectory("c:\SundownSolutions")

$Date = Get-Date

$Filename="IllegalFileNameReport{0:d2}{1:d2}{2:d2}-{3:d2}{4:d2}.txt" -f $date.day,$date.month,$date.year,$date.hour,$date.minute

$Filename2 = "c:\SundownSolutions\" + $Filename

#----------------------------------------------------------------------------------------------------Conduct The Search and Output the File

$File = Get-ChildItem $DirectoryToSearch -include '\*&\*','\*#\*','\*%\*','\*{\*','\*}\*','\*~\*','\*..\*','\*...\*' -recurse | Sort-Object -property $GroupBy | format-table -property Name, DirectoryName, LastWriteTime, Length -groupby $GroupBy | Out-File $Filename2 -width 180

$File

#----------------------------------------------------------------------------------------------------Open File in IE

$ie = new-object -comobject "InternetExplorer.Application"

$ie.visible = $true

$ie.navigate($Filename2)

#----------------------------------------------------------------------------------------------------Dispose of all variables - Cleaning up :)

function Dispose-All {

Get-Variable -exclude Runspace |

Where-Object {

$\_.Value -is [System.IDisposable]

} |

Foreach-Object {

$\_.Value.Dispose()

Remove-Variable $\_.Name

}

}

# **Annex 1c - Check Illegal File Names and Blocked File Types**

#----------------------------------------------------------------------------------------------------Dialog Box 1 - Folder To be Searched

[void] [System.Reflection.Assembly]::LoadWithPartialName("System.Drawing")

[void] [System.Reflection.Assembly]::LoadWithPartialName("System.Windows.Forms")

[array]$DropDownArray = "DirectoryName","Name","Length","LastWriteTime" #----------------------------The Group By Array

#----------------------------------------------------------------------------------------------------This Function Returns the Selected Value and Closes the Form

function Return-DropDown {

$Choice = $DropDown.SelectedItem.ToString()

$Form.Close()

Write-Host $Choice

}

$objForm = New-Object System.Windows.Forms.Form

$objForm.Text = "Select Folder to Check"

$objForm.Size = New-Object System.Drawing.Size(290,300)

$objForm.StartPosition = "CenterScreen"

$objForm.KeyPreview = $True

$objForm.Add\_KeyDown({if ($\_.KeyCode -eq "Enter")

{$x=$objTextBox.Text;$objForm.Close()}})

$objForm.Add\_KeyDown({if ($\_.KeyCode -eq "Escape")

{$objForm.Close()}})

$DropDownLabel = new-object System.Windows.Forms.Label

$DropDownLabel.Location = new-object System.Drawing.Size(10,140)

$DropDownLabel.size = new-object System.Drawing.Size(280,30)

$DropDownLabel.Text = "Select how you want the report to be grouped"

$objform.Controls.Add($DropDownLabel)

$DropDown = new-object System.Windows.Forms.ComboBox

$DropDown.Location = new-object System.Drawing.Size(10,170)

$DropDown.Size = new-object System.Drawing.Size(260,20)

ForEach ($Item in $DropDownArray) {

$DropDown.Items.Add($Item)

}

$objform.Controls.Add($DropDown)

$OKButton = New-Object System.Windows.Forms.Button

$OKButton.Location = New-Object System.Drawing.Size(75,230)

$OKButton.Size = New-Object System.Drawing.Size(75,23)

$OKButton.Text = "OK"

$OKButton.Add\_Click({$DirectoryToSearch=$objTextBox.Text;$GroupBy=$DropDown.Text;$objForm.Close()})

$objForm.Controls.Add($OKButton)

$CancelButton = New-Object System.Windows.Forms.Button

$CancelButton.Location = New-Object System.Drawing.Size(150,230)

$CancelButton.Size = New-Object System.Drawing.Size(75,23)

$CancelButton.Text = "Cancel"

$CancelButton.Add\_Click({$objForm.Close()})

$objForm.Controls.Add($CancelButton)

$objLabel = New-Object System.Windows.Forms.Label

$objLabel.Location = New-Object System.Drawing.Size(10,10)

$objLabel.Size = New-Object System.Drawing.Size(280,40)

$objLabel.Text = "Please enter the name of the directory you want to Search for illegal SharePoint Filenames"

$objForm.Controls.Add($objLabel)

$objLabel1 = New-Object System.Windows.Forms.Label

$objLabel1.Location = New-Object System.Drawing.Size(10,80)

$objLabel1.Size = New-Object System.Drawing.Size(280,40)

$objLabel1.Text = "Please ensure you use the following format c:\Name\Name or \\name\name"

$objForm.Controls.Add($objLabel1)

$objTextBox = New-Object System.Windows.Forms.TextBox

$objTextBox.Location = New-Object System.Drawing.Size(10,50)

$objTextBox.Size = New-Object System.Drawing.Size(260,20)

$objForm.Controls.Add($objTextBox)

$objForm.Topmost = $True

$objForm.Add\_Shown({$objForm.Activate()})

[void] $objForm.ShowDialog()

$DirectoryToSearch

$GroupBy

#----------------------------------------------------------------------------------------------------Create Output Folder and Date Stamped File name

[IO.Directory]::CreateDirectory("c:\SundownSolutions")

$Date = Get-Date

$Filename="BothReports{0:d2}{1:d2}{2:d2}-{3:d2}{4:d2}.txt" -f $date.day,$date.month,$date.year,$date.hour,$date.minute

$Filename2 = "c:\SundownSolutions\" + $Filename

#----------------------------------------------------------------------------------------------------Conduct The Search and Output the File

$File = Get-ChildItem $DirectoryToSearch -include '\*&\*','\*#\*','\*%\*','\*{\*','\*}\*','\*~\*','\*..\*','\*...\*',\*.ade,\*.adp,\*.app,\*.asa,\*.ashx,\*.asmx,\*.asp,\*.bas,\*.bat,\*.cdx,\*.cer,\*.chm,\*.class,\*.cmd,\*.cnt,\*.com,\*.config,\*.cpl,\*.crt,\*.csh,\*.der,\*.dll,\*.exe,\*.fxp,\*.gadget,\*.grp,\*.hlp,\*.hpj,\*.hta,\*.htr,\*.htw,\*.ida,\*.idc,\*.idq,\*.ins,\*.isp,\*.its,\*.jse,\*.ksh,\*.lnk,\*.mad,\*.maf,\*.mag,\*.mam,\*.maq,\*.mar,\*.mas,\*.mat,\*.mau,\*.mav,\*.maw,\*.mcf,\*.mda,\*.mdb,\*.mde,\*.mdt,\*.mdw,\*.mdz,\*.msc,\*.msh,\*.msh1,\*.msh1xml,\*.msh2,\*.msh2xml,\*.mshxml,\*.msi,\*.msp,\*.mst,\*.ops,\*.pcd,\*.pif,\*.pl,\*.prf,\*.prg,\*.printer,\*.ps1,\*.ps1xml,\*.ps2,\*.ps2xml,\*.psc1,\*.psc2,\*.pst,\*.reg,\*.rem,\*.scf,\*.scr,\*.sct,\*.shb,\*.shs,\*.shtm,\*.shtml,\*.soap,\*.stm,\*.svc,\*.url,\*.vb,\*.vbe,\*.vbs,\*.ws,\*.wsc,\*.wsf,\*.wsh -recurse | Sort-Object -property $GroupBy | format-table -property Name, DirectoryName, LastWriteTime, Length -groupby $GroupBy | Out-File $Filename2 -width 180

$File

#----------------------------------------------------------------------------------------------------Open File in IE

$ie = new-object -comobject "InternetExplorer.Application"

$ie.visible = $true

$ie.navigate($Filename2)

#----------------------------------------------------------------------------------------------------Dispose of all variables - Cleaning up :)

function Dispose-All {

Get-Variable -exclude Runspace |

Where-Object {

$\_.Value -is [System.IDisposable]

} |

Foreach-Object {

$\_.Value.Dispose()

Remove-Variable $\_.Name

}

}

# **Annex 1d - Change Illegal Characters**

#----------------------------------------------------------------------------------------------------Dialog Box 1 - Folder To be Searched

[void] [System.Reflection.Assembly]::LoadWithPartialName("System.Drawing")

[void] [System.Reflection.Assembly]::LoadWithPartialName("System.Windows.Forms")

[array]$DropDownArray = "DirectoryName","Name","Length","LastWriteTime" #----------------------------The Group By Array

#----------------------------------------------------------------------------------------------------Thiscreates the dialog box and assigns all variables when the ok button is pressed

function Return-DropDown {

$Choice = $DropDown.SelectedItem.ToString()

$Form.Close()

Write-Host $Choice

}

$objForm = New-Object System.Windows.Forms.Form

$objForm.Text = "Illegal File Type Rename"

$objForm.Size = New-Object System.Drawing.Size(290,500)

$objForm.StartPosition = "CenterScreen"

$objForm.KeyPreview = $True

$objForm.Add\_KeyDown({if ($\_.KeyCode -eq "Enter")

{$x=$objTextBox.Text;$objForm.Close()}})

$objForm.Add\_KeyDown({if ($\_.KeyCode -eq "Escape")

{$objForm.Close()}})

$DropDownLabel = new-object System.Windows.Forms.Label

$DropDownLabel.Location = new-object System.Drawing.Size(10,140)

$DropDownLabel.size = new-object System.Drawing.Size(280,30)

$DropDownLabel.Text = "Select how you want the report to be grouped"

$objform.Controls.Add($DropDownLabel)

$DropDown = new-object System.Windows.Forms.ComboBox

$DropDown.Location = new-object System.Drawing.Size(10,170)

$DropDown.Size = new-object System.Drawing.Size(260,20)

ForEach ($Item in $DropDownArray) {

$DropDown.Items.Add($Item)

}

$objform.Controls.Add($DropDown)

$objLabel = New-Object System.Windows.Forms.Label

$objLabel.Location = New-Object System.Drawing.Size(10,10)

$objLabel.Size = New-Object System.Drawing.Size(280,40)

$objLabel.Text = "Please enter the name of the directory you want to scan for illegal SharePoint filenames"

$objForm.Controls.Add($objLabel)

$objLabel1 = New-Object System.Windows.Forms.Label

$objLabel1.Location = New-Object System.Drawing.Size(10,80)

$objLabel1.Size = New-Object System.Drawing.Size(280,40)

$objLabel1.Text = "Please ensure you use the following format c:\Name\Name or \\name\name"

$objForm.Controls.Add($objLabel1)

$objTextBox = New-Object System.Windows.Forms.TextBox

$objTextBox.Location = New-Object System.Drawing.Size(10,50)

$objTextBox.Size = New-Object System.Drawing.Size(260,20)

$objForm.Controls.Add($objTextBox)

$IFT1bL = New-Object System.Windows.Forms.Label

$IFT1bL.Location = New-Object System.Drawing.Size(20,220)

$IFT1bL.Size = New-Object System.Drawing.Size(130,20)

$IFT1bL.Text = "Type Replacement for #"

$objForm.Controls.Add($IFT1bL)

$IFT1b = New-Object System.Windows.Forms.TextBox

$IFT1b.Location = New-Object System.Drawing.Size(170,220)

$IFT1b.Size = New-Object System.Drawing.Size(100,20)

$objForm.Controls.Add($IFT1b)

$IFT2bL = New-Object System.Windows.Forms.Label

$IFT2bL.Location = New-Object System.Drawing.Size(20,245)

$IFT2bL.Size = New-Object System.Drawing.Size(130,20)

$IFT2bL.Text = "Type Replacement for %"

$objForm.Controls.Add($IFT2bL)

$IFT2b = New-Object System.Windows.Forms.TextBox

$IFT2b.Location = New-Object System.Drawing.Size(170,245)

$IFT2b.Size = New-Object System.Drawing.Size(100,20)

$objForm.Controls.Add($IFT2b)

$IFT3bL = New-Object System.Windows.Forms.Label

$IFT3bL.Location = New-Object System.Drawing.Size(20,270)

$IFT3bL.Size = New-Object System.Drawing.Size(130,20)

$IFT3bL.Text = "Type Replacement for {"

$objForm.Controls.Add($IFT3bL)

$IFT3b = New-Object System.Windows.Forms.TextBox

$IFT3b.Location = New-Object System.Drawing.Size(170,270)

$IFT3b.Size = New-Object System.Drawing.Size(100,20)

$objForm.Controls.Add($IFT3b)

$IFT4bL = New-Object System.Windows.Forms.Label

$IFT4bL.Location = New-Object System.Drawing.Size(20,295)

$IFT4bL.Size = New-Object System.Drawing.Size(130,20)

$IFT4bL.Text = "Type Replacement for }"

$objForm.Controls.Add($IFT4bL)

$IFT4b = New-Object System.Windows.Forms.TextBox

$IFT4b.Location = New-Object System.Drawing.Size(170,295)

$IFT4b.Size = New-Object System.Drawing.Size(100,20)

$objForm.Controls.Add($IFT4b)

$IFT5bL = New-Object System.Windows.Forms.Label

$IFT5bL.Location = New-Object System.Drawing.Size(20,320)

$IFT5bL.Size = New-Object System.Drawing.Size(130,20)

$IFT5bL.Text = "Type Replacement for ~"

$objForm.Controls.Add($IFT5bL)

$IFT5b = New-Object System.Windows.Forms.TextBox

$IFT5b.Location = New-Object System.Drawing.Size(170,320)

$IFT5b.Size = New-Object System.Drawing.Size(100,20)

$objForm.Controls.Add($IFT5b)

$IFT8bL = New-Object System.Windows.Forms.Label

$IFT8bL.Location = New-Object System.Drawing.Size(20,345)

$IFT8bL.Size = New-Object System.Drawing.Size(130,20)

$IFT8bL.Text = "Type Replacement for &&"

$objForm.Controls.Add($IFT8bL)

$IFT8b = New-Object System.Windows.Forms.TextBox

$IFT8b.Location = New-Object System.Drawing.Size(170,345)

$IFT8b.Size = New-Object System.Drawing.Size(100,20)

$objForm.Controls.Add($IFT8b)

$ReminderLabel = New-Object System.Windows.Forms.Label

$ReminderLabel.Location = New-Object System.Drawing.Size(20,370)

$ReminderLabel.Size = New-Object System.Drawing.Size(280,40)

$ReminderLabel.Text = "If you leave all fields blank the characters will be removed - if the filename already exists within the directory the file will not be altered"

$objForm.Controls.Add($ReminderLabel)

$OKButton = New-Object System.Windows.Forms.Button

$OKButton.Location = New-Object System.Drawing.Size(75,420)

$OKButton.Size = New-Object System.Drawing.Size(75,23)

$OKButton.Text = "OK"

$OKButton.Add\_Click({$DirectoryToSearch=$objTextBox.Text;$GroupBy=$DropDown.Text;$IFT1c=$IFT1b.Text;$IFT2c=$IFT2b.Text;$IFT3c=$IFT3b.Text;$IFT4c=$IFT4b.Text;$IFT5c=$IFT5b.Text;$IFT6c=$IFT6b.Text;$IFT7c=$IFT7b.Text;$IFT8c=$IFT8b.Text;$objForm.Close()})

$objForm.Controls.Add($OKButton)

$CancelButton = New-Object System.Windows.Forms.Button

$CancelButton.Location = New-Object System.Drawing.Size(150,420)

$CancelButton.Size = New-Object System.Drawing.Size(75,23)

$CancelButton.Text = "Cancel"

$CancelButton.Add\_Click({$objForm.Close()})

$objForm.Controls.Add($CancelButton)

$objForm.Topmost = $True

$objForm.Add\_Shown({$objForm.Activate()})

[void] $objForm.ShowDialog()

$DirectoryToSearch

$GroupBy

#----------------------------------------------------------------------------------------------------Define the illegal filenames

$IFT1 = '\*#\*'

$IFT1a ='#'

$IFT2 = '\*%\*'

$IFT2a = '%'

$IFT3 = '\*{\*'

$IFT3a = '{'

$IFT4 = '\*}\*'

$IFT4a = '}'

$IFT5 = '\*~\*'

$IFT5a = '~'

$IFT8 = '\*&\*'

$IFT8a = '&'

$IFT = $IFT1,$IFT2,$IF3,$IFT4,$IFT5,$IFT6,$IFT7,$IFT8

#----------------------------------------------------------------------------------------------------Create Output Folder and Date Stamped File name + the Rename parameter

[IO.Directory]::CreateDirectory("c:\SundownSolutions\")

[IO.Directory]::CreateDirectory("c:\SundownSolutions\RenamedFiles")

$Date = Get-Date

$Filename="FileRename{0:d2}{1:d2}{2:d2}-{3:d2}{4:d2}.txt" -f $date.day,$date.month,$date.year,$date.hour,$date.minute

$Filename0 = "c:\SundownSolutions\RenamedFiles\"+$IFT1a + $Filename

$Filename1 = "c:\SundownSolutions\RenamedFiles\"+$IFT2a + $Filename

$Filename2 = "c:\SundownSolutions\RenamedFiles\"+$IFT3a + $Filename

$Filename3 = "c:\SundownSolutions\RenamedFiles\"+$IFT4a + $Filename

$Filename4 = "c:\SundownSolutions\RenamedFiles\"+$IFT5a + $Filename

$Filename7 = "c:\SundownSolutions\RenamedFiles\"+$IFT8a + $Filename

#----------------------------------------------------------------------------------------------------Conduct The Search, Change the File Names and then output the results to the log files

$File1 = Get-ChildItem $DirectoryToSearch -include $IFT1 -recurse | Sort-Object -property $GroupBy | format-table -property Name, DirectoryName, LastWriteTime, Length -groupby $GroupBy | Out-File $Filename0 -width 180

$File1 = Get-ChildItem $DirectoryToSearch -include $IFT1 -recurse | Rename-Item -NewName {$\_.Name -replace $IFT1a,$IFT1c}

$File2 = Get-ChildItem $DirectoryToSearch -include $IFT2 -recurse | Sort-Object -property $GroupBy | format-table -property Name, DirectoryName, LastWriteTime, Length -groupby $GroupBy | Out-File $Filename1 -width 180

$File2 = Get-ChildItem $DirectoryToSearch -include $IFT2 -recurse | Rename-Item -NewName {$\_.Name -replace $IFT2a,$IFT2c}

$File3 = Get-ChildItem $DirectoryToSearch -include $IFT3 -recurse | Sort-Object -property $GroupBy | format-table -property Name, DirectoryName, LastWriteTime, Length -groupby $GroupBy | Out-File $Filename2 -width 180

$File3 = Get-ChildItem $DirectoryToSearch -include $IFT3 -recurse | Rename-Item -NewName {$\_.Name -replace $IFT3a,$IFT3c}

$File4 = Get-ChildItem $DirectoryToSearch -include $IFT4 -recurse | Sort-Object -property $GroupBy | format-table -property Name, DirectoryName, LastWriteTime, Length -groupby $GroupBy | Out-File $Filename3 -width 180

$File4 = Get-ChildItem $DirectoryToSearch -include $IFT4 -recurse | Rename-Item -NewName {$\_.Name -replace $IFT4a,$IFT4c}

$File5 = Get-ChildItem $DirectoryToSearch -include $IFT5 -recurse | Sort-Object -property $GroupBy | format-table -property Name, DirectoryName, LastWriteTime, Length -groupby $GroupBy | Out-File $Filename4 -width 180

$File5 = Get-ChildItem $DirectoryToSearch -include $IFT5 -recurse | Rename-Item -NewName {$\_.Name -replace $IFT5a,$IFT5c}

$File8 = Get-ChildItem $DirectoryToSearch -include $IFT8 -recurse | Sort-Object -property $GroupBy | format-table -property Name, DirectoryName, LastWriteTime, Length -groupby $GroupBy | Out-File $Filename7 -width 180

$File8 = Get-ChildItem $DirectoryToSearch -include $IFT8 -recurse | Rename-Item -NewName {$\_.Name -replace $IFT8a,$IFT8c}

#----------------------------------------------------------------------------------------------------Script Complete confirmation

Write-Host "The Data Cleanse Script has completed - please check the log files to confirm file changes - Press ENTER to continue"

$x = $host.UI.RawUI.ReadKey("NoEcho,IncludeKeyDown")

#----------------------------------------------------------------------------------------------------Open log folder in explorer

Start Explorer.exe 'c:\SundownSolutions\RenamedFiles'

#----------------------------------------------------------------------------------------------------Dispose of all variables - Cleaning up :)

function Dispose-All {

Get-Variable |

Where-Object {

$\_.Value -is [System.IDisposable]

} |

Foreach-Object {

$\_.Value.Dispose()

Remove-Variable $\_.Name

}

}

# Annex B – Blocked File Types

These are the files that are scanned by the blocked file types script:

|  |  |  |  |
| --- | --- | --- | --- |
| **ade** | **grp** | **maw** | **ps1** |
| **adp** | **hlp** | **mcf** | **ps1xml** |
| **app** | **hpj** | **mda** | **ps2** |
| **asa** | **hta** | **mdb** | **ps2xml** |
| **ashx** | **htr** | **mde** | **psc1** |
| **asmx** | **htw** | **mdt** | **psc2** |
| **asp** | **ida** | **mdw** | **reg** |
| **bas** | **idc** | **mdz** | **rem** |
| **bat** | **idq** | **msc** | **scf** |
| **cdx** | **ins** | **msh** | **scr** |
| **cer** | **isp** | **msh1** | **sct** |
| **chm** | **its** | **msh1xml** | **shb** |
| **class** | **jse** | **msh2** | **shs** |
| **cmd** | **ksh** | **msh2xml** | **shtm** |
| **cnt** | **lnk** | **mshxml** | **shtml** |
| **com** | **mad** | **msi** | **soap** |
| **config** | **maf** | **msp** | **stm** |
| **cpl** | **mag** | **mst** | **svc** |
| **crt** | **mam** | **ops** | **vb** |
| **csh** | **maq** | **pcd** | **vbe** |
| **der** | **mar** | **pif** | **vbs** |
| **dll** | **mas** | **pl** | **ws** |
| **fxp** | **mat** | **prf** | **wsc** |
| **gadget** | **mau** | **prg** | **wsf** |
|  | **mav** | **printer** | **wsh** |