BACKUPSCRIPT

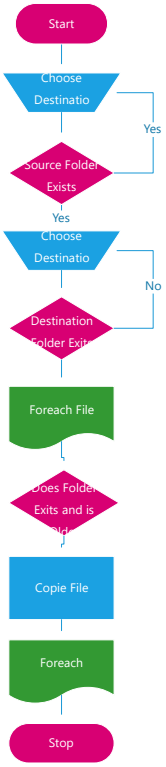
Skript zum Testen erstellen, welches Verzeichnisse übergibt. Erwartet Rückgabewert, vor Backup zählen der Dateien/Ordner im Src Verzeichnis,
nach Backup zählen der Dateien im Dest Verzeichniss.
Ausgabe in Test_LOG Datei mit Src, Dest, Anzahl Dateien, Stimmt überein?

Skript Backup so abändern, dass der Folder Picker nur kommt wenn keine Verzeichnisse übergeben wurden.

SELEKTIVER BACKUP
Erweiter Backupscript mit Frage ob Verzeichnisse ausgeschlossen werden sollen, -> Folder Picker wenn nichts übergeben.

INKREMENTELLER BACKUP
Vor dem Kopieren wird unüberprüft ob die Datei bereits in einem Alten Backup vorhanden ist und falls ja ob sie verändert wurde.

PAP anpassen und weitere Versionen erstellen für selektiv bzw. inkrementell




```
#########################################################
# Name: BackupScript.ps1
# Creator: Ratcorp
# CreationDate: 21.11.2019
# LastModified: 16.12.2019
# Version: 0.3
#
#########################################################
```

```powershell
Function Get-FolderName($initialDirectory, $m) {
    [System.Reflection.Assembly]::LoadWithPartialName("System.windows.forms") | Out-Null

    $OpenFolderDialog = New-Object System.Windows.Forms.FolderBrowserDialog
    if($m -eq "src"){$OpenFolderDialog.Description = "Select the folder to backup"}
    else {$OpenFolderDialog.Description = "Select the destination folder"}
    $Topmost = New-Object System.Windows.Forms.Form
    $Topmost.TopMost = $True
    $Topmost.MinimizeBox = $True

    $OpenFolderDialog.RootFolder = $initialDirectory
    $OpenFolderDialog.ShowDialog($Topmost) | Out-Null
    return $OpenFolderDialog.SelectedPath
}

#Variables, only Change here
$Destination=$args[0]
if(!$Destination){
$Destination=Get-FolderName Desktop dest #Copy the Files to this Location
}
#$Destination="C:\Users\seimi\Downloads"
$Staging="C:\Users\seimi\Downloads\Staging"
$ClearStaging=$true # When $true, Staging Dir will be cleared
$Versions="5" #How many of the last Backups you want to keep
$BackupDirs=$args[1]
if(!$BackupDirs){
$BackupDirs=Get-FolderName Desktop src #What Folders you want to backup
}

$ExcludeDirs="C:\Users\seimi\OneDrive - Seidl Michael\0-Temp","C:\Users\seimi\OneDrive -
Seidl Michael\0-Temp\Dir2" #This list of Directories will not be copied

$LogName="Log.txt" #Log Name
$LoggingLevel="3" #LoggingLevel only for Output in Powershell Window, 1=smart, 3=Heavy


#STOP-no changes from here
#STOP-no changes from here
#Settings - do not change anything from here

$ExcludeString=""
#[string[]]$excludedArray = $ExcludeDirs -split ","
foreach ($Entry in $ExcludeDirs)
{
    $Temp="^"+$Entry.Replace("\","\\")
    $ExcludeString+=$Temp+"|"
}
$ExcludeString=$ExcludeString.Substring(0,$ExcludeString.Length-1)
#$ExcludeString
[RegEx]$exclude = $ExcludeString

if ($UseStaging -and $Zip)
{
    #Logging "INFO" "Use Temp Backup Dir"
    $Backupdir=$Staging +"\Backup-"+ (Get-Date -format yyyy-MM-dd)+"-"+(Get-Random -
Maximum 100000)+"\"
}
else
{
    #Logging "INFO" "Use orig Backup Dir"
    $Backupdir=$Destination +"\Backup-"+ (Get-Date -format yyyy-MM-dd)+"-"+(Get-Random -
Maximum 100000)+"\"
}



#$BackupdirTemp=$Temp +"\Backup-"+ (Get-Date -format yyyy-MM-dd)+"-"+(Get-Random -
Maximum 100000)+"\"
$Log=$Backupdir+$LogName
$Log
$Items=0
$Count=0
$ErrorCount=0
$StartDate=Get-Date #-format dd.MM.yyyy-HH:mm:ss

#FUNCTION
#Logging
Function Logging ($State, $Message) {
    $Datum=Get-Date -format dd.MM.yyyy-HH:mm:ss
```

```
    if (!(Test-Path -Path $Log)) {
        New-Item -Path $Log -ItemType File | Out-Null
    }
    $Text="$Datum - $State"+":"+" $Message"

    if ($LoggingLevel -eq "1" -and $Message -notmatch "was copied") {Write-Host $Text}
    elseif ($LoggingLevel -eq "3") {Write-Host $Text}

    add-Content -Path $Log -Value $Text

}


#Create Backupdir
Function Create-Backupdir {
    New-Item -Path $Backupdir -ItemType Directory | Out-Null
    sleep -Seconds 5
    Logging "INFO" "Create Backupdir $Backupdir"
}

#Delete Backupdir
Function Delete-Backupdir {
    $Folder=Get-ChildItem $Destination | where {$_.Attributes -eq "Directory"} | Sort-
Object -Property CreationTime -Descending:$false | Select-Object -First 1

    Logging "INFO" "Remove Dir: $Folder"

    $Folder.FullName | Remove-Item -Recurse -Force
}


#Check if Backupdirs and Destination is available
function Check-Dir {
    Logging "INFO" "Check if BackupDir and Destination exists"
    if (!(Test-Path $BackupDirs)) {
        return $false
        Logging "Error" "$BackupDirs does not exist"
    }
    if (!(Test-Path $Destination)) {
        return $false
        Logging "Error" "$Destination does not exist"
    }
}

#Save all the Files
Function Make-Backup {
    Logging "INFO" "Started the Backup"
    $Files=@()
    $SumMB=0
    $SumItems=0
    $SumCount=0
    $colItems=0
    $Count=0
    Logging "INFO" "Count all files and create the Top Level Directories"

    foreach ($Backup in $BackupDirs) {
        $colItems = (Get-ChildItem $Backup -recurse | Where-Object {$_.mode -notmatch
"h"} | Measure-Object -property length -sum)
        $Items=0
        $FilesCount += Get-ChildItem $Backup -Recurse | Where-Object {$_.mode -notmatch
"h"}
        Copy-Item -Path $Backup -Destination $Backupdir -Force -ErrorAction
SilentlyContinue
        $SumMB+=$colItems.Sum.ToString()
        $SumItems+=$colItems.Count
    }
    $Folders = Get-ChildItem -Directory $Backup -Recurse
    $FoldersCount = $Folders.Count

    $TotalMB="{0:N2}" -f ($SumMB / 1MB) + " MB of Files"
    Logging "INFO" "There are $FoldersCount Folders with $SumItems Files with  $TotalMB
to copy"

    foreach ($Backup in $BackupDirs) {
        $Index=$Backup.LastIndexOf("\")
        $SplitBackup=$Backup.substring(0,$Index)
        $Files = Get-ChildItem $Backup -Recurse  | select * | Where-Object {$_.mode -
notmatch "h" -and $_.fullname -notmatch $exclude} | select fullname #$_.mode -notmatch
"h" -and
        foreach ($File in $Files) {
```

```
            $restpath = $file.fullname.replace($SplitBackup,"")
            try {
                Copy-Item  $file.fullname $($Backupdir+$restpath) -Force -ErrorAction
SilentlyContinue |Out-Null
                Logging "INFO" "$file was copied"
            }
            catch {
                $ErrorCount++
                Logging "ERROR" "$file returned an error an was not copied"
            }
            $Items += (Get-item $file.fullname).Length
            $status = "Copy file {0} of {1} and copied {3} MB of {4} MB: {2}" -f $count,
$SumItems,$file.Name,("{0:N2}" -f ($Items / 1MB)).ToString(),("{0:N2}" -f ($SumMB /
1MB)).ToString()
            $Index=[array]::IndexOf($BackupDirs,$Backup)+1
            $Text="Copy data Location {0} of {1}" -f $Index ,$BackupDirs.Count
            Write-Progress -Activity $Text $status ($Items / $SumMB*100)
            write-host
            $count++
        }
    }
    $SumCount+=$Count-$FoldersCount
    $SumTotalMB="{0:N2}" -f ($Items / 1MB) + " MB of Files"
    Logging "INFO" "----------------------"
    Logging "INFO" "Created $FoldersCount Folders and Copied $SumCount files with
$SumTotalMB"
    Logging "INFO" "$ErrorCount Files could not be copied"

}


#create Backup Dir



Create-Backupdir
Logging "INFO" "----------------------"
Logging "INFO" "Start the Script"

#Check if Backupdir needs to be cleaned and create Backupdir
$Count=(Get-ChildItem $Destination | where {$_.Attributes -eq "Directory"}).count
Logging "INFO" "Check if there are more than $Versions Directories in the Backupdir"

if ($count -gt $Versions)
{

    Delete-Backupdir
}


#Check if all Dir are existing and do the Backup
$CheckDir=Check-Dir

if ($CheckDir -eq $false) {
    Logging "ERROR" "One of the Directory are not available, Script has stopped"
} else {
    Make-Backup

    $Enddate=Get-Date #-format dd.MM.yyyy-HH:mm:ss
    $span = $EndDate - $StartDate
    $Minutes=$span.Minutes
    $Seconds=$Span.Seconds

    Logging "INFO" "Backupduration $Minutes Minutes and $Seconds Seconds"
    Logging "INFO" "----------------------"
    Logging "INFO" "----------------------"
   }
```