<#

SearchPPTSDocs.ps1

.SYNOPSIS

Constructs a search of the PPTS doc class group

#>

Param(

[Parameter(Mandatory=$true)]

[string]$RECORD\_ID,

[Parameter(Mandatory=$true)]

[string]$RECORD\_NAME

)

# Load the M-Files API.

[Reflection.Assembly]::LoadWithPartialName( "Interop.MFilesAPI" )

#\*=============================================================================

#\* FUNCTIONS

#\*=============================================================================

#\*=============================================================================

#\* SCRIPT BODY

#\*=============================================================================

# Connect to the M-Files server with current Windows user (must be system administrator).

$mfclient = New-Object MFilesAPI.MFilesClientApplicationClass

$mfConnections = $mfclient.GetVaultConnections()

foreach ($mfvaultConnection in $mfConnections)

{

if ($mfvaultConnection.Name -eq "CCSF Prod")

{

# Write-Host "Got it"

break

}

}

if ($mfvaultConnection.IsLoggedIn())

{

$mfvault = $mfvaultConnection.BindToVault(0, $false, $false)

Write-Host "No login needed"

}

else

{

$mfvault = $mfvaultConnection.LoginAsUser(3, "PublicUser", "remember", $null, $null)

}

$mySearchCondition = New-Object MFilesAPI.SearchConditionClass

$mySearchCondition.ConditionType = 1 # 1 is equals, 9 is startswith

$prop = $mfVault.PropertyDefOperations.GetPropertyDefIDByAlias("M-Files.Property.Project")

$mySearchCondition.Expression.DataPropertyValuePropertyDef = $prop

$myDataType = 1

$value = $RECORD\_ID+"--"+$RECORD\_NAME **p1020=**

#$value = [System.Text.Encoding]::Unicode.GetBytes($command)

#[System.Convert]::ToBase64String($value)

$mySearchCondition.TypedValue.SetValue($myDataType, $value)

$mySearchConditions = New-Object MFilesAPI.SearchConditionsClass

$mySearchConditions.Add(-1, $mySearchCondition)

# Document Category NOT Working Document **p1044!=Working Document**

$mySearchCondition = New-Object MFilesAPI.SearchConditionClass

$mySearchCondition.ConditionType = 2 # 2 is NOT equals, 9 is startswith

$prop = $mfVault.PropertyDefOperations.GetPropertyDefIDByAlias("M-Files.Property.DocumentCategory")

$mySearchCondition.Expression.DataPropertyValuePropertyDef = $prop

$myDataType = 1

$value = "Working Document"

$mySearchCondition.TypedValue.SetValue($myDataType, $value)

$mySearchConditions.Add(-1, $mySearchCondition)

# Document Category NOT Confidential **p1044!=Confidential**

$mySearchCondition = New-Object MFilesAPI.SearchConditionClass

$mySearchCondition.ConditionType = 2 # 2 is NOT equals, 9 is startswith

$prop = $mfVault.PropertyDefOperations.GetPropertyDefIDByAlias("M-Files.Property.DocumentCategory")

$mySearchCondition.Expression.DataPropertyValuePropertyDef = $prop

$myDataType = 1

$value = "Confidential"

$mySearchCondition.TypedValue.SetValue($myDataType, $value)

$mySearchConditions.Add(-1, $mySearchCondition)

# Document Category NOT Background Documents **p1044!=Background Documents**

$mySearchCondition = New-Object MFilesAPI.SearchConditionClass

$mySearchCondition.ConditionType = 2 # 2 is NOT equals, 9 is startswith

$prop = $mfVault.PropertyDefOperations.GetPropertyDefIDByAlias("M-Files.Property.DocumentCategory")

$mySearchCondition.Expression.DataPropertyValuePropertyDef = $prop

$myDataType = 1

$value = "Background Documents"

$mySearchCondition.TypedValue.SetValue($myDataType, $value)

$mySearchConditions.Add(-1, $mySearchCondition)

# NOT Deleted **(don’t need to add this to query string)**

$mySearchCondition = New-Object MFilesAPI.SearchConditionClass

$mySearchCondition.ConditionType = 1 # 1 is equals, 9 is startswith

$mySearchCondition.Expression.DataStatusValueType = 5 #5 is deleted

$myDataType = 8 # boolean

$value = $false

$mySearchCondition.TypedValue.SetValue($myDataType, $value)

$mySearchConditions.Add(-1, $mySearchCondition)

$myResults = $mfVault.ObjectSearchOperations.SearchForObjectsByConditionsEx($mySearchConditions, 0, $false, 0)

# Set up list for files

$mFiles = @()

foreach ($mfResult in $myResults)

{

$myObjID = $mfVault.ObjectOperations.GetObjIDByGUID($mfResult.ObjectGUID)

$oResult = $mfVault.ObjectOperations.GetLatestObjectVersionAndProperties($myObjID, $true)

foreach ($oPropertyValue2 in $oResult.Properties)

{

# Find the property definition for the property.

$oPropertyDef = $mfVault.PropertyDefOperations.GetPropertyDef($oPropertyValue2.PropertyDef)

# Output the name and the value of the property.

if ($oPropertyDef.Name -eq "Name or Title") {

$vDocName = $oPropertyValue2.TypedValue.DisplayValue }

if ($oPropertyDef.Name -eq "Document Group") {

$vDocGroup = $oPropertyValue2.TypedValue.DisplayValue }

if ($oPropertyDef.Name -eq "Document Category") {

$vDocCategory = $oPropertyValue2.TypedValue.DisplayValue }

}

$myFiles = $mfResult.Files

foreach ($file in $myFiles)

{

if ($vDocName -eq $file.Title)

{

$vListName = $file.Title

}

else

{

$vListName = $vDocName + "--" + $file.Title

}

# Write-Host $mfVault.GetGUID() " has this object " $mfResult.ObjectGUID " with this file " $file.FileGUID

$mFile = New-Object PSObject -Property @{

"DocName" = $vListName

"VaultGUID" = $mfVault.GetGUID()

"ObjectGUID" = $mfResult.ObjectGUID

"FileGUID" = $file.FileGUID

"DocGroup" = $vDocGroup

"DocCategory" = $vDocCategory

"FileSize" = $file.LogicalSize

"LastModified" = $file.LastWriteTimeUtc.ToShortDateString()

}

$mFiles += $mFile

}

# $mFiles | Export-Csv c:\Scripts\testPPTSdocs.csv -delimiter "|"

}

return $mFiles