

MFG500067

Vault Advanced Administration

Kimberley Hendrix
D3 Technologies

Learning Objectives

- Learn how to analyze a vault for data issues
- Learn how to set up corrective actions to clean the vault using PowerShell
- Learn how to create some weekly tasks for reporting
- Learn how to utilize the job processor to help maintain a clean healthy Vault

Description

Basic administration of Vault software is pretty clean, but what about when your data over the years has gotten dirty. Ever feel like you'd like to run your data through a power wash? Maintaining a clean functioning Vault takes time if done manually. In this class, we'll explore ways to use PowerShell and the API to clean, rearrange, and add metadata to your data. Duplicate file names, missing information, other data sources, or maybe a new file structure. You can handle all of these things and more with PowerShell and the API.

Speaker

Based in Tulsa, Oklahoma, Kimberley Hendrix provides custom solutions for lean engineering using Autodesk, Inc., products, and industry knowledge to streamline design and engineering departments. Hendrix has worked in the manufacturing industry for over 30 years and she specialized in automated solutions for manufacturers. She has worked with Autodesk products since 1984. Hendrix is associated with D3 Technologies as the Manager of Data Management, focusing on data management and the customization of applications to streamline access to data across an organization

Analyzing your Vault

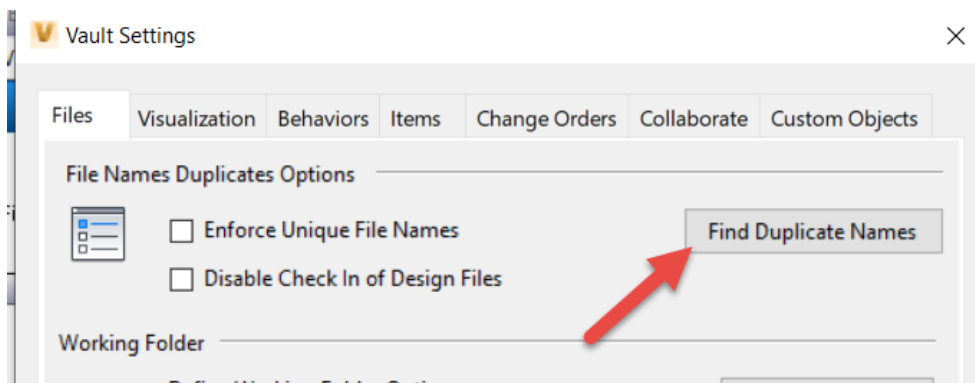
The first step to a clean healthy vault is to know what to look for. There are some useful built in reports and then some queries we can do using powerShell.

Vault built in reporting

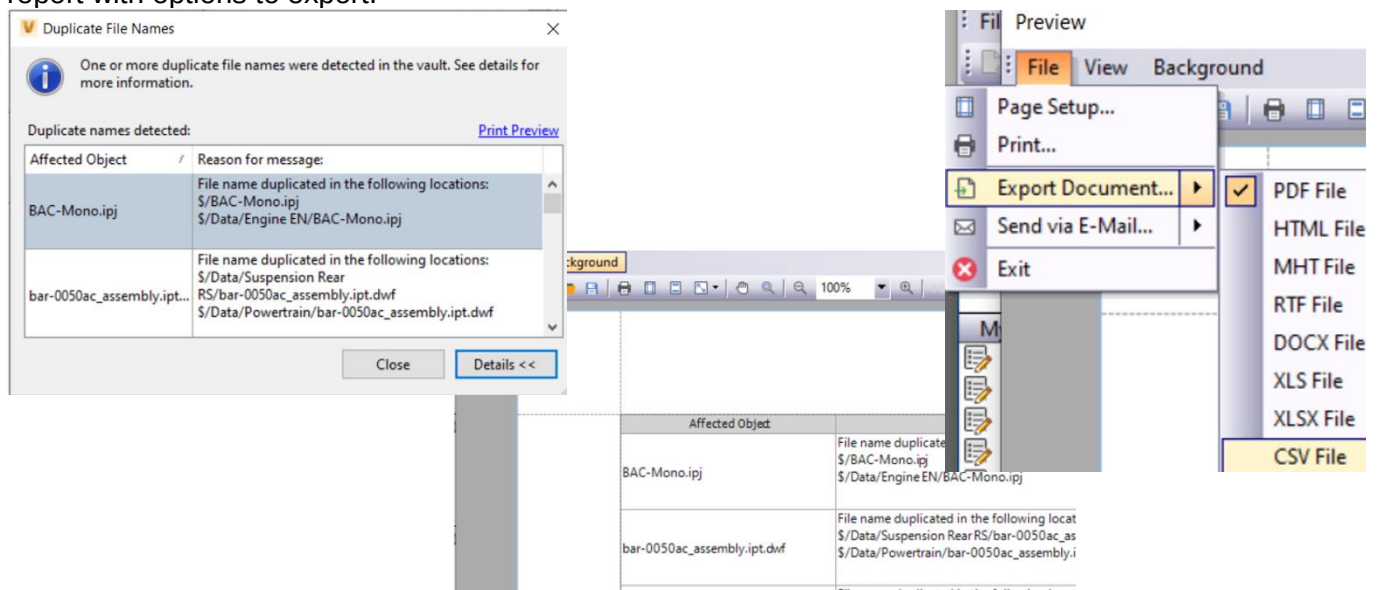
Most vaults started out not configured in the best possible way, or had a large data dump added, and the *Enforce Unique File Names* could not be checked.

If that is the case with your vault, this handy report, will identify all duplicate names, and even export a file that you can act on.

The duplicate files report is a quick way to check the health of your vault. It is accessed through the Tools -> Administration -> Vault Settings Menu.



The system view will look like the following, but if you hit the details button, you get a full screen report with options to export.



AUTODESK UNIVERSITY

Be sure to save the exported file from above, we will discuss some actions we can take using that data.

Vault Advanced Find Queries

Advanced Find in the Vault client is very powerful, you can also export findings for email notifications or data to work against with scripting.

First let's look at some of the advanced finds I do when cleaning and maintaining a vault.

1. Files checked out for an extended period of time.
 - a. This is a good search for a Design & Engineering Manager to run to keep track of the progress of files, or the habits of designers and keeping their data clean.
 - b. Properties to look at for this are:
 - i. Checked out by – not empty
 - ii. Checked Out – On or Before – date of your choice
2. Change Orders not modified for a period of time
 - a. Properties to look at for this are:
 - i. Look for: Change Orders
 - ii. Date Modified – before – date of your choice
3. Orphaned Files – part files (ipt) without a parent
 - a. A typical part file would have at least one parent, either an Assembly (iam) or a Drawing (idw/dwg)
 - b. If the file does not have a parent, it may need to be removed.
4. Files without Visualizations files
 - a. Properties to look at for this are:
 - i. Checked out by – is empty
 - ii. Visualization attachment – is none
 - iii. Provider – (either AutoCAD or Inventor)

With any of the above queries you can act on that data

Let's clean that Vault

We will go in depth on how to clear out duplicate file names using the export from earlier, then move into some interesting scripts to manipulate that data that is in the vault.

Duplicate File Names

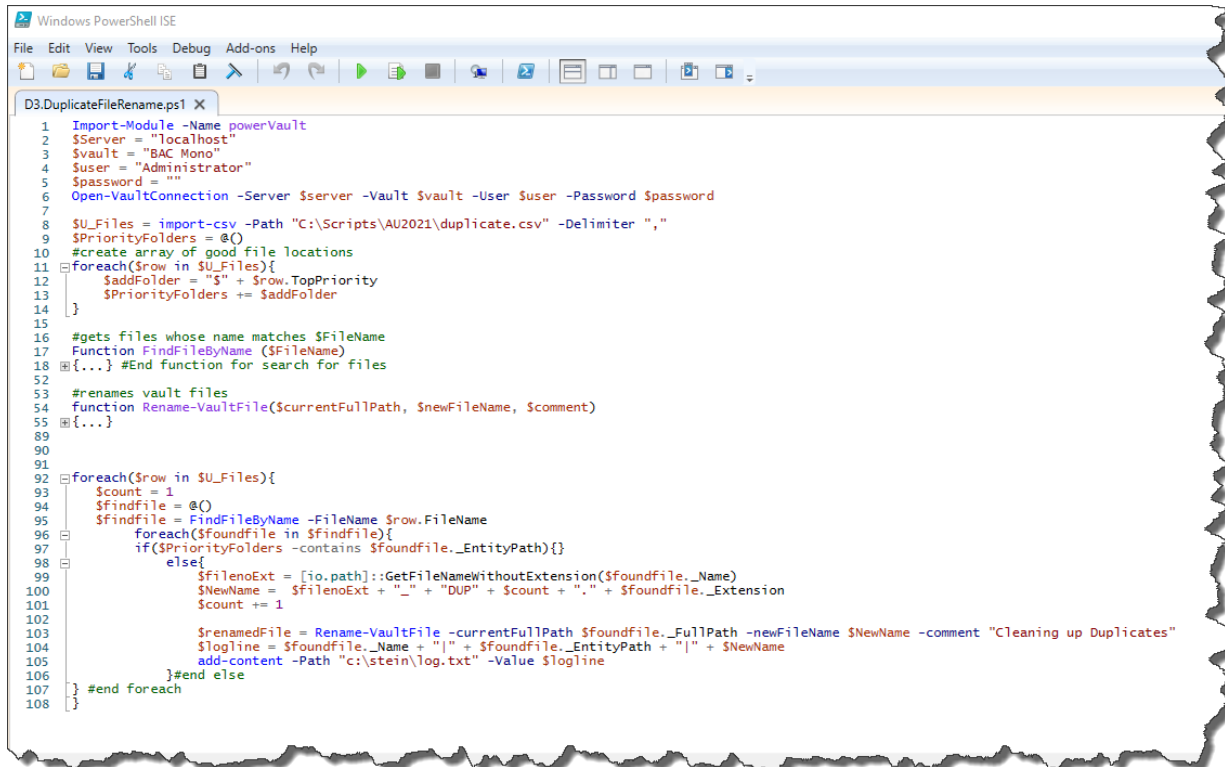
Duplicate file names in your vault can create various errors and eat up a designer's time trying to figure out which is the correct file. While this script doesn't remove the files, it will rename them with an extension, sorted by date, so you know that a file referenced in your assembly or that you are working on is duplicated with a new version and you can clean it up intelligently.

AUTODESK UNIVERSITY

Create a script that does the following,
(Pseudo Code)

- 1) Log in to the Vault with administrative rights
- 2) Import the CSV file from the Duplicate file search export
- 3) For each line in the CSV file
 - a) Search for all the files with that name
 - b) Sort those files by date
 - c) Skip the newest file (do not rename it)
 - d) For each of the remaining files rename each one of them with a suffix of _Dup#
- 4) Write a log file of each file touched (optional)

Converting that to actual PowerShell looks like this (actual PS1 file included in the materials section of this class)



```
D3.DuplicateFileRename.ps1 X
1 Import-Module -Name powerVault
2 $Server = "localhost"
3 $Vault = "BAC Mono"
4 $User = "Administrator"
5 $Password = ""
6 Open-VaultConnection -Server $Server -Vault $Vault -User $User -Password $Password
7
8 $U_Files = import-csv -Path "C:\Scripts\AU2021\duplicate.csv" -Delimiter ","
9 $PriorityFolders = @()
10 #create array of good file locations
11 foreach($row in $U_Files){
12     $addFolder = "$" + $row.TopPriority
13     $PriorityFolders += $addFolder
14 }
15
16 #gets files whose name matches $FileName
17 Function FindFileByName ($FileName)
18 {
19     [...] #End function for search for files
20 }
21
22 #renames vault files
23 function Rename-VaultFile($currentFullPath, $newFileName, $comment)
24 {
25     [...]
26 }
27
28 foreach($row in $U_Files){
29     $count = 1
30     $findFile = @()
31     $findFile = FindFileByName -FileName $row.FileName
32     foreach($foundFile in $findFile){
33         if($PriorityFolders -contains $foundFile._EntityPath){}
34         else{
35             $fileNoExt = [io.path]::GetFileNameWithoutExtension($foundFile._Name)
36             $NewName = $fileNoExt + "_" + "DUP" + $count + "." + $foundFile._Extension
37             $count += 1
38
39             $renamedFile = Rename-VaultFile -currentFullPath $foundFile._FullPath -newFileName $NewName -comment "Cleaning up Duplicates"
40             $logline = $foundFile._Name + "|" + $foundFile._EntityPath + "|" + $NewName
41             add-content -Path "c:\stein\log.txt" -Value $logline
42         }
43     }
44 }
45 #end foreach
```

When writing code to manipulate data in the Vault I always run this in a test environment first, and if a test environment is not available, take your CSV file and reduce it to one file that you can easily reverse any data changes that are not good.

AUTODESK UNIVERSITY

Data from another source

Many times, after an initial load of data there is data any other forms about the files, excel files, list, schedules etc. that if this data was applied to the files in vault these extra tracking files could be eliminated.

In the following example we will look at applying a series of custom vault property values to files from an external source. I prefer to work with CSV files instead of Excel simply because they read in faster with extraneous overhead.

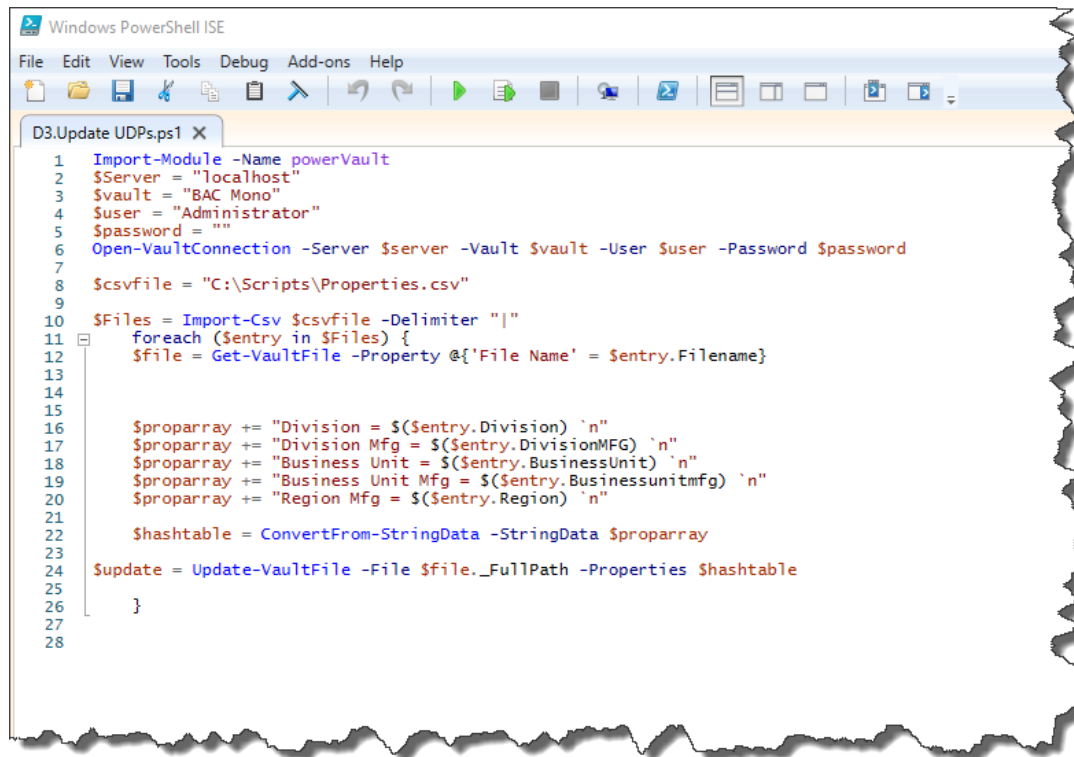
Create a script that does the following,
(Pseudo Code)

- 1) Log in the Vault with administrative rights
- 2) Import the CSV file with the FileName and properties
- 3) For each line in the CSV file
 - a. Search for and get the file object for the referenced file
 - b. Extract each of the property values from the CSV
 - c. Create a hashtable of the property values
 - d. Perform an Update-VaultFile
 - i. Update-VaultFile is a function of coolOrange's PowerVault utility

Converting that to actual PowerShell looks like this (actual PS1 file included in the materials section of this class)

The CSV file for this example looks like this

```
Filename|Division|DivisionMFG|BusinessUnit|Businessunitmg|Region  
Abcde.ipt|Tulsa|Tubing|Heat Exchangers|North Tulsa|South west
```



```
1 Import-Module -Name powerVault
2 $Server = "localhost"
3 $Vault = "BAC Mono"
4 $User = "Administrator"
5 $Password = ""
6 Open-VaultConnection -Server $Server -Vault $Vault -User $User -Password $Password
7
8 $Csvfile = "C:\Scripts\Properties.csv"
9
10 $Files = Import-Csv $Csvfile -Delimiter "|"
11 foreach ($Entry in $Files) {
12     $File = Get-VaultFile -Property @{ 'File Name' = $Entry.FileName }
13
14
15
16     $propparray += "Division = $($Entry.Division) `n"
17     $propparray += "Division Mfg = $($Entry.DivisionMfg) `n"
18     $propparray += "Business Unit = $($Entry.BusinessUnit) `n"
19     $propparray += "Business Unit Mfg = $($Entry.Businessunitmfg) `n"
20     $propparray += "Region Mfg = $($Entry.Region) `n"
21
22     $Shashtable = ConvertFrom-StringData -StringData $propparray
23
24     $update = Update-VaultFile -File $File._FullPath -Properties $Shashtable
25
26 }
27
28
```

Keeping a clean vault

Now that you have your vault all cleaned up, how do you maintain that environment? As an administrator you may need to keep track of some items for maintenance as well. In this section we will explore using the job processor to tackle some of these tasks on a weekly basis. In order to run a PowerShell script from the job processor, powerJobs from coolOrange will be required, a link to coolOrange is included in the presentation.

Visualization files

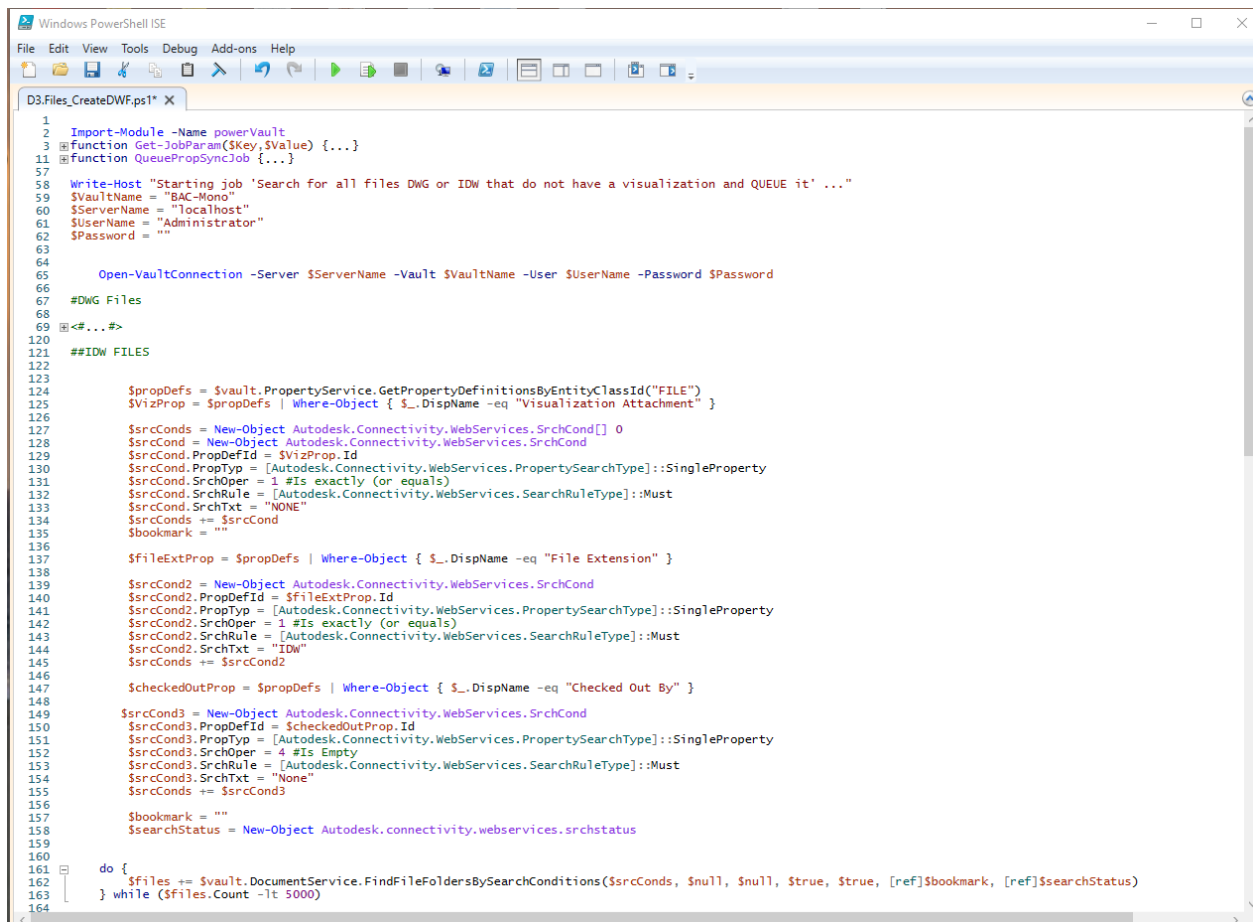
In a clean healthy vault, all the Visualization files (DWF's) are generated as needed, on lifecycle change, upon check in etc. sometimes however they may fail with a non-tip error meaning the file was touched or checked out prior to the DWF creation, or a property was edited, and vault automatically disassociates the DWF with the file. For whatever reason sometimes they need to be recreated.

A Cron Job (timed job) can be added to powerJobs that executes every Saturday morning when the processor is not being used to search the vault for all files that the Visualization attachment is "NONE" and the file is not checked out and execute a job to generate the associated DWF.

AUTODESK UNIVERSITY

Create a Job script that does the following,
("C:\ProgramData\coolOrange\powerJobs\Jobs\D3.Files_CreateDWF.ps1")
(Pseudo Code)

- 1) Log into the Vault with administrative rights
- 2) Perform a search with the following criteria
 - a. Visualization Attachment = NONE
 - b. Checked out by – is Empty
 - c. Provider = Inventor or AutoCAD
- 3) For Each file in the search
 - a. Queue a Sync Properties Job (it will subsequently queue the DWF creation)
- 4) Limit each weekend job to a number that can complete over the weekend usually less than 5000



```
1  Import-Module -Name powerVault
2  function Get-JobParam($Key,$Value) {...}
3  function QueuePropSyncJob {...}
11 function QueuePropSyncJob {...}
57
58 Write-Host "Starting job 'Search for all files DWG or IDW that do not have a visualization and QUEUE it' ..."
59 $VaultName = "BAC-Mono"
60 $ServerName = "localhost"
61 $UserName = "Administrator"
62 $Password = ""
63
64
65 Open-VaultConnection -Server $ServerName -Vault $VaultName -User $UserName -Password $Password
66
67 #DWG Files
68
69 <#...#>
120
121 ##IDW FILES
122
123
124 $propDefs = $Vault.PropertyService.GetPropertyDefinitionsByEntityClassId("FILE")
125 $VizProp = $propDefs | Where-Object { $_.DispName -eq "Visualization Attachment" }
126
127 $srcConds = New-Object Autodesk.Connectivity.WebServices.SrchCond[] 0
128 $srcCond = New-Object Autodesk.Connectivity.WebServices.SrchCond
129 $srcCond.PropDefId = $VizProp.Id
130 $srcCond.PropType = [Autodesk.Connectivity.WebServices.PropertySearchType]::SingleProperty
131 $srcCond.SrchOper = 1 #Is exactly (or equals)
132 $srcCond.SrchRule = [Autodesk.Connectivity.WebServices.SearchRuleType]::Must
133 $srcCond.SrchTxt = "NONE"
134 $srcConds += $srcCond
135 $bookmark = ""
136
137 $fileExtProp = $propDefs | Where-Object { $_.DispName -eq "File Extension" }
138
139 $srcCond2 = New-Object Autodesk.Connectivity.WebServices.SrchCond
140 $srcCond2.PropDefId = $fileExtProp.Id
141 $srcCond2.PropType = [Autodesk.Connectivity.WebServices.PropertySearchType]::SingleProperty
142 $srcCond2.SrchOper = 1 #Is exactly (or equals)
143 $srcCond2.SrchRule = [Autodesk.Connectivity.WebServices.SearchRuleType]::Must
144 $srcCond2.SrchTxt = "IDW"
145 $srcConds += $srcCond2
146
147 $checkedOutProp = $propDefs | Where-Object { $_.DispName -eq "Checked Out By" }
148
149 $srcCond3 = New-Object Autodesk.Connectivity.WebServices.SrchCond
150 $srcCond3.PropDefId = $checkedOutProp.Id
151 $srcCond3.PropType = [Autodesk.Connectivity.WebServices.PropertySearchType]::SingleProperty
152 $srcCond3.SrchOper = 4 #Is Empty
153 $srcCond3.SrchRule = [Autodesk.Connectivity.WebServices.SearchRuleType]::Must
154 $srcCond3.SrchTxt = "None"
155 $srcConds += $srcCond3
156
157 $bookmark = ""
158 $searchStatus = New-Object Autodesk.connectivity.webservices.srchstatus
159
160
161 do {
162     $files += $Vault.DocumentService.FindFileFoldersBySearchConditions($srcConds, $null, $null, $true, $true, [ref]$bookmark, [ref]$searchStatus)
163 } while ($files.Count -lt 5000)
164
```

AUTODESK UNIVERSITY

```
165 write-host "file count: '$($files.Count)'"
166
167 #lets do 5000 a night
168 $limit = 5000
169 $counter = 1
170
171 foreach($filefolder in $files)
172 {
173     If($files.Count -lt 2) {break}
174     #Write-Host "file name: '$($filefolder.file.name)'"
175     #ScoFiles = Get-VaultFile -FileId $filefolder.file.id
176     $vdffile = QueuePropSyncJob -File $filefolder.file -QueueDwfJob
177     $counter = $counter +1
178     write-host "Number of Files Queued $($counter) of Limit $($limit) "
179     if ($counter -eq $limit){exit}
180 }
181 #
182
183
184
185
186
```

Reporting on the Health and Wellbeing of your server

The vault just sits in the background and runs right? Ever worry about drive space, SQL server database size or if the backups ran overnight.

Put the job processor to work checking and reporting on those things at least weekly. Again, for PowerShell and Cron to work you will need powerJobs by coolOrange.

```
D3.ServerInfo.ps1 X
1 Import-Module -Name powerVault
2
3 $emailText = $null
4
5 $today = Get-Date -Format "dddd MM/dd/yyyy"
6 $CR = " `r`n"
7
8 $emailText = "Server Status for Date: " + $today + $CR + $CR
9
10 #Get the Database Size(s)
11 $dbSizes = Invoke-Sqlcmd -query "sp_databases" -Database 'BAC Mono' -serverInstance 'D3-0752\AUTODESKVAULT'
12 $bacmono = $dbSizes | Where-Object {$_.DATABASE_NAME -EQ 'BAC Mono'}
13
14 $DBSize = $bacmono.DATABASE_SIZE / 1GB
15
16 if ([int]$DBSize -lt 1){
17     $DBSize = "Less than 1GB"
18 }
19
20 $emailText = $emailText + "Database size for Vault BAC MONO is: " + $DBSize + $CR + $CR
21
22 #Get the Drive Size where the FileStore is stored
23
24 $disk = Get-WmiObject Win32_LogicalDisk -ComputerName D3-0752 -Filter "DeviceID='C:'" | Select-Object Size,FreeSpace
25
26 $dsize = $disk.Size / 1GB
27 $dfree = $disk.FreeSpace / 1GB
28 $emailText = $emailText + "Disk Size / Free Space on Server with Filestore is: " + [int]$dsize + "GB / " + [int]$dfree + "GB" + $CR
29
30 #send report to Admin
31 #BEGIN USER INPUT
32
33 $SMTPServer="192.168.1.206" #SMTP Server
34 $From="kim.hendrix@teamd3.com" #Email "From"
35 $To="kim.hendrix@teamd3.com" #Email "To"
36
37 $EmailSubject = "Vault Server Report for " + $today
38
39 Send-MailMessage -SMTPServer $SMTPServer -From $From -To $To -Subject $EmailSubject -Body $emailText
40
```

AUTODESK UNIVERSITY

Summary

Maintenance and upkeep of your data is an important and can be a time-consuming job. However, with the use of PowerShell, coolOrange's powerJobs and some intuitive scripts you can make this much easier and somewhat self-cleaning.

References

- Autodesk AKN is a valuable resource of examples and tutorials
- coolOrange product PowerVault is invaluable and is a great entry into the Vault API <http://www.coolorange.com>
- Markus Koechl's GitHub is a wealth of information: <https://github.com/koechlm>
- D3 Technologies Web site and Blog: <http://www.teamd3.com>
- Code snippets from this class will be available on the AU website

Contact information

KIMBERLEY HENDRIX
MANAGER OF DATA MANAGEMENT

D3 TECHNOLOGIES
Kim.Hendrix@TeamD3.com
LinkedIn
<https://www.linkedin.com/in/kimberleyhendrix/>