AUTODESK. UNIVERSITY

# Vault – Configure don’t Customize – the Power of Data Standards (Part 2 of 2)

**Kimberley Hendrix**

Solutions Consultant, D3 Technologies| @kimhendrix80


# About the speaker

## Kimberley Hendrix

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 30 years and she specialized in automated solutions for the heat exchanger industry. She has worked with Autodesk products since 1984. Hendrix is associated with D3 Technologies as a solutions consultant, focusing on data management, plant, automation, and mechanical issues

# Class Summary

## Vault – Configure don’t Customize – the Power of Data Standards (Part 2 of 2)

**Part two of a two-part series discussing the ways to Configure your vault without expensive custom code that must be compiled year over year. Part two will cover the use of Data Standards. We will explore various use cases and configurations of Data Standards to make your Autodesk Vault work for you in your environment. How does Data Standards work? How to effectively deploy Data Standards to your end users. And how to create effective utilities with Data Standards, solving irritating discrepancies in your data. How to utilize your Custom Objects from Part one: Vault-Configure don't Customize - the Power of Custom Objects**

# the Power of Data Standards

## Data Standards is provided with Vault Professional and Vault Workgroup

**Autodesk Definition - Vault Data Standard is a data control feature helping you to ensure design relevant information is captured in a standardized format during the data entry process in Vault Client, Inventor and AutoCAD. (AKN)**

**My Definition – an add-on tool with simplified access to the Vault API using PowerShell, allowing for applications and utilities to be created to streamline data management in a consistent manner. The programming behind Vault Data Standards is primarily done in PowerShell and therefore does not require complicated compiling year over year for upgrades.**

# Why Data Standards?

1. It is provided with Autodesk Vault Clients both Workgroup and Professional.
2. Customizations afforded with Data Standards do not require reworking and compiling year over year.
3. The language used to configure Data Standards automation is PowerShell, available with all Windows installations.
4. The user interface, or dialog boxes are written in XAML format
    1. XAML, which stands for eXtensible Application Markup Language, is Microsoft's variant of XML for describing a GUI. In previous GUI frameworks, like WinForms, a GUI was created in the same language that you would use for interacting with the GUI, e.g. C# or VB.NET and usually maintained by the designer (e.g. Visual Studio), but with XAML, Microsoft is going another way. Much like with HTML, you are able to easily write and edit your GUI in a text.
5. The next years upgrade will generally only require copying the files to a new location:
    1. For example, from C:\ProgramData\Autodesk\Vault 2020\Extensions\DataStandard

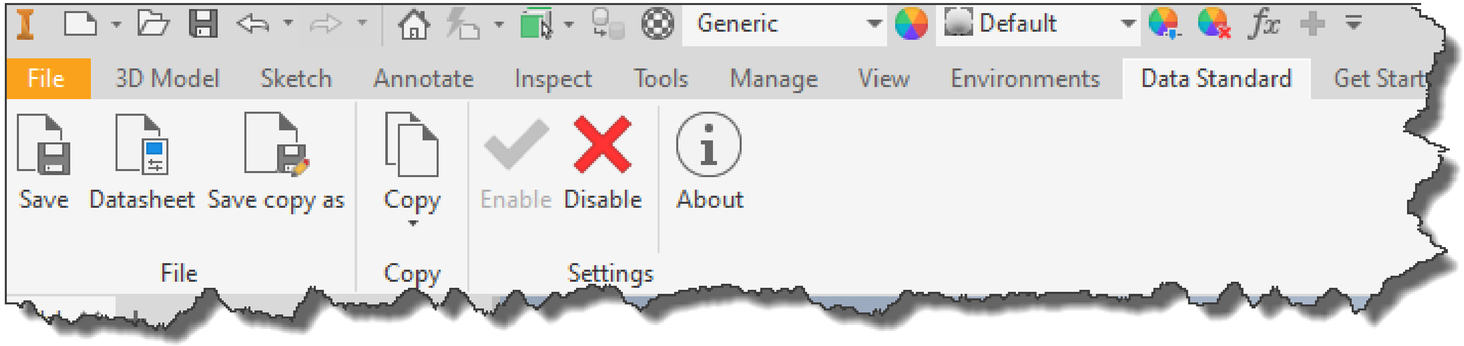
       to

       C:\ProgramData\Autodesk\Vault 2021\Extensions\DataStandard

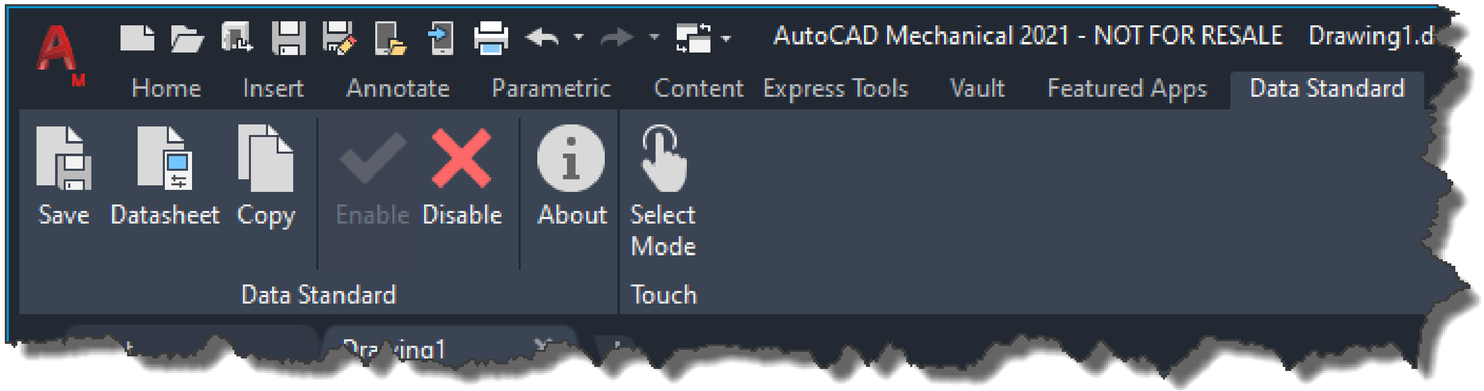
# Data Standards “OOB”
# Inventor & AutoCAD

**Out of the Box Data Standards without any modifications includes**

- **Save**
- **Datasheet**
- **Save Copy as (Inventor)**
- **Copy**
- **Disable**




Inventor Data Standard Ribbon

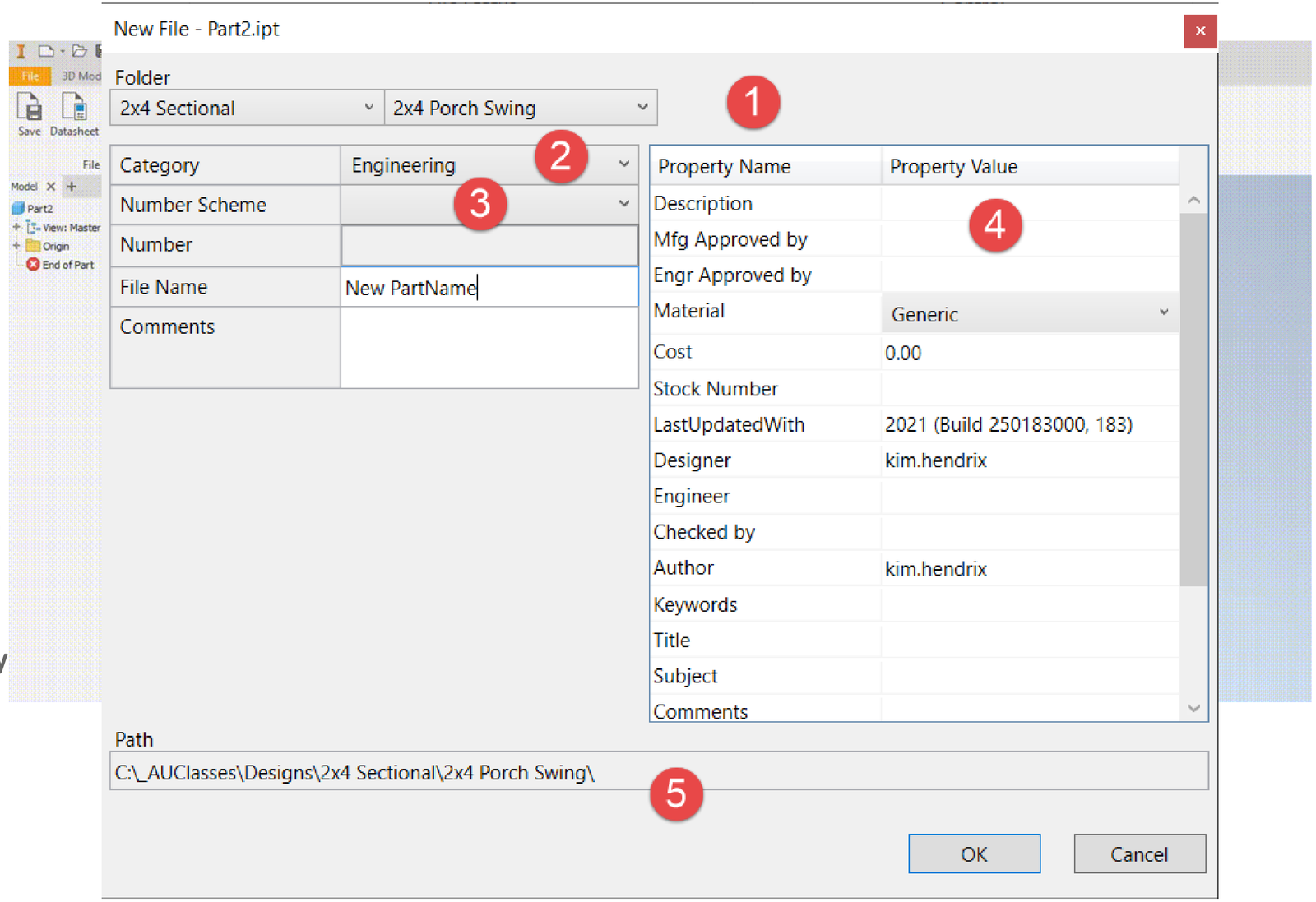


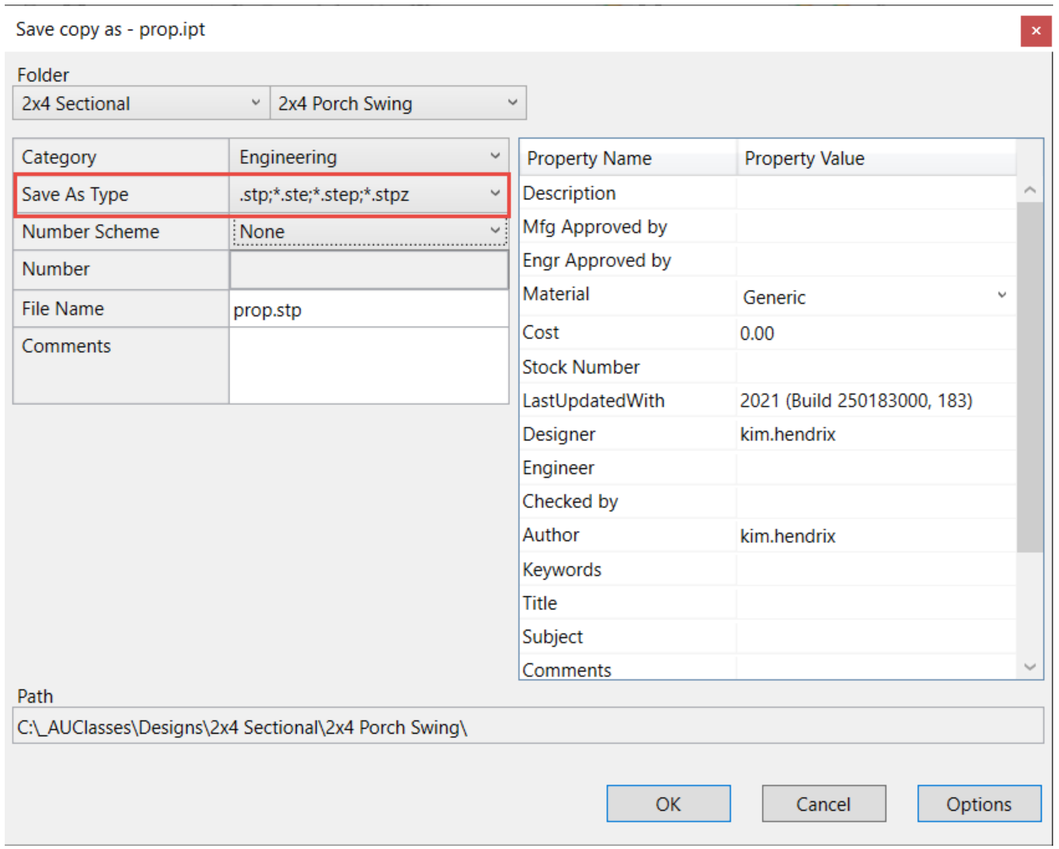

AutoCAD Data Standard Ribbon

# Data Standards “OOB” Save

The Inventor and AutoCAD VDS Save dialogs are very similar the difference is only the dynamic property grid.

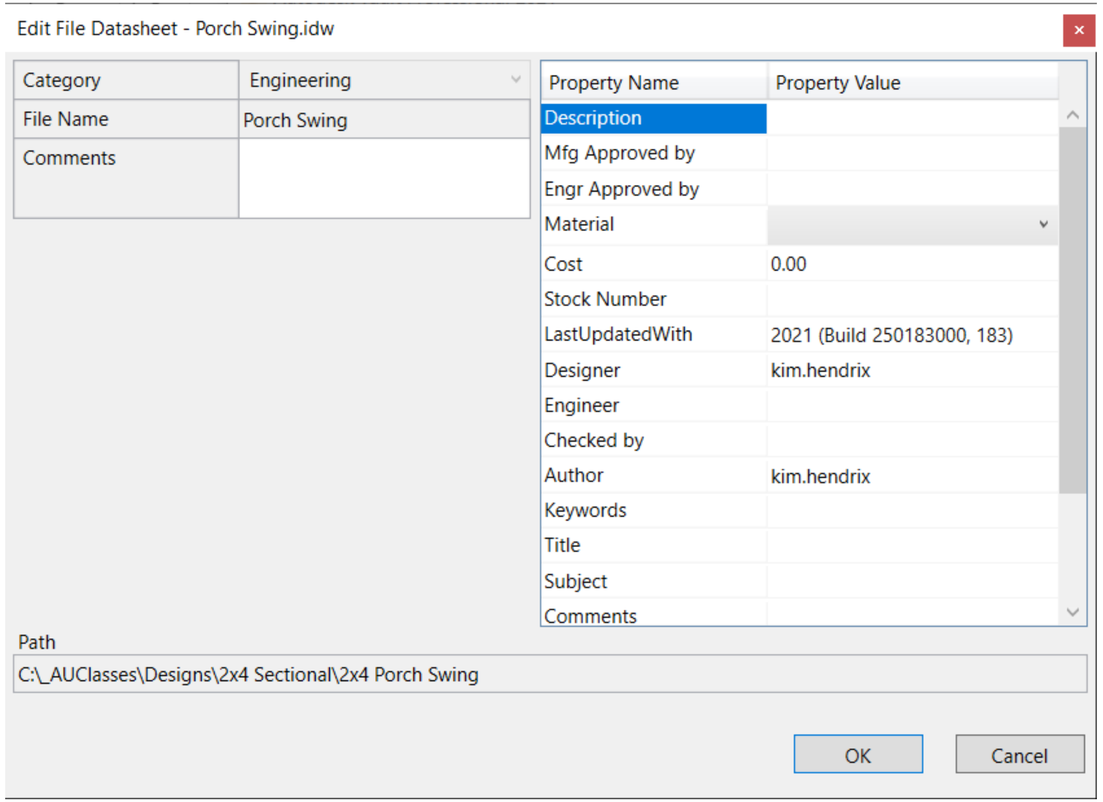
1. Folder: is dynamic and is reading from the Vault. Avoiding random folder creation in the Vault
2. Category: is dynamic and is reading from the Vault the property grid changes as the Category changes
3. Number Scheme: is dynamic and reading from the Vault.
4. Property Grid: dynamic and read from the configured properties of the Category
5. Local Path: Determined from the Vault folder picked in step 1






# Data Standards “OOB”
# Save Copy AS (Inventor Only)

**The Save Copy As dialog is very similar to the Save dialog  except it adds a Save as Type drop down, the options on this drop down are dependent on the type of file you are working with, a Part/Assembly will have Stp & JT, but a drawing file (IDW or DWG) will have AutoCAD DWG, DXF & PDF.**



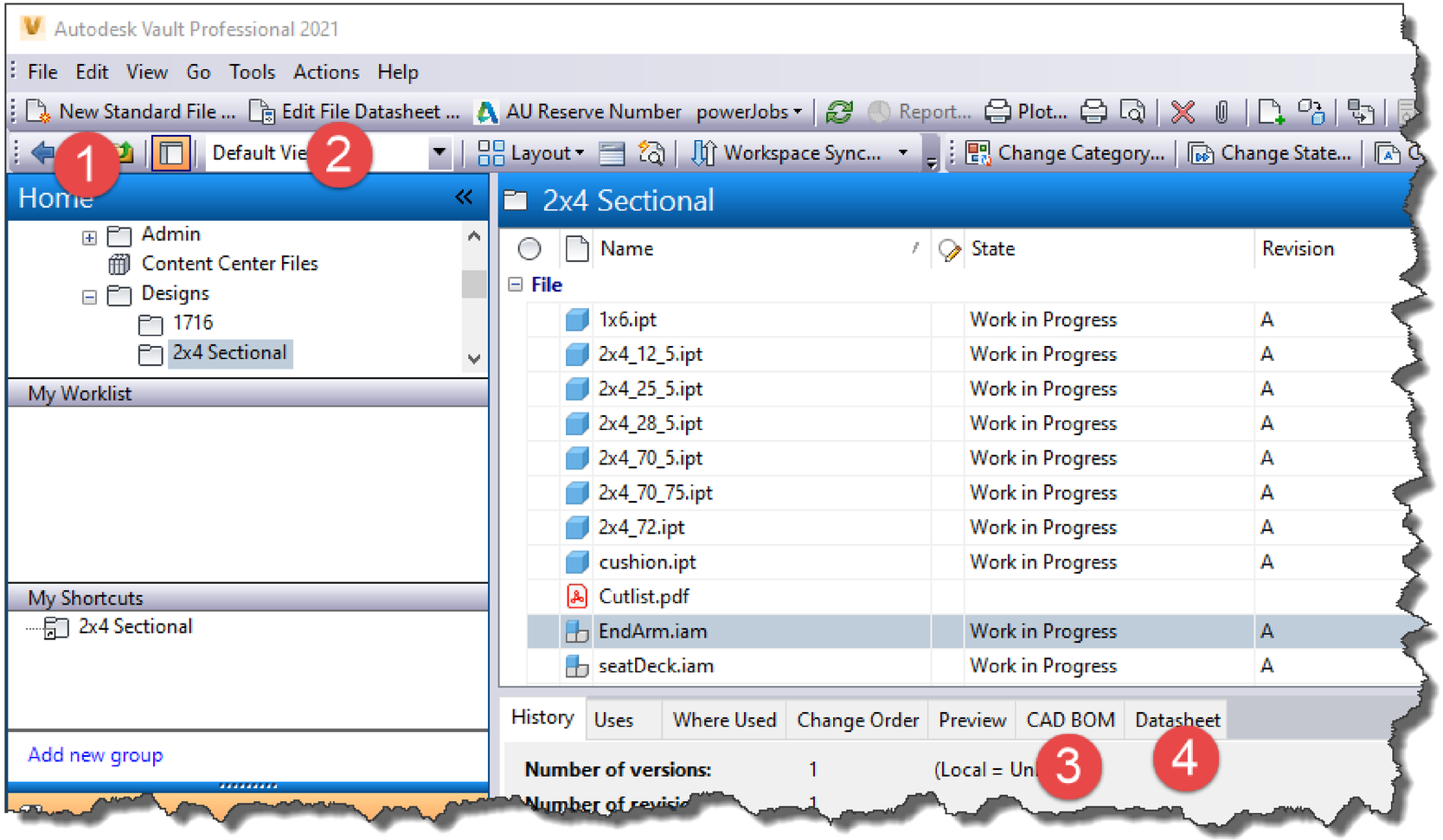

# Data Standards “OOB”
# Edit Data Sheet

**The Edit Data Sheet dialog is almost a duplicate of the Save dialog box, without the Naming system, it is designed to allow easy modifications of properties.**

# Data Standards “OOB”
# Vault Client

Vault Client Data Standards

1. New Standard File: same dialog as in the CAD applications except the addition of Template choice.
2. Edit File Datasheet: same dialog as in the CAD applications
3. CAD BOM: on Inventor Assembly files it will display the BOM information from the CAD file. Read Only
4. Datasheet: an easy few of properties



# The Anatomy of Vault Data Standards

# Three parts of Vault Data Standards

MENU D

Defines

Locatio

MenuDefinitions.xml - Notepad

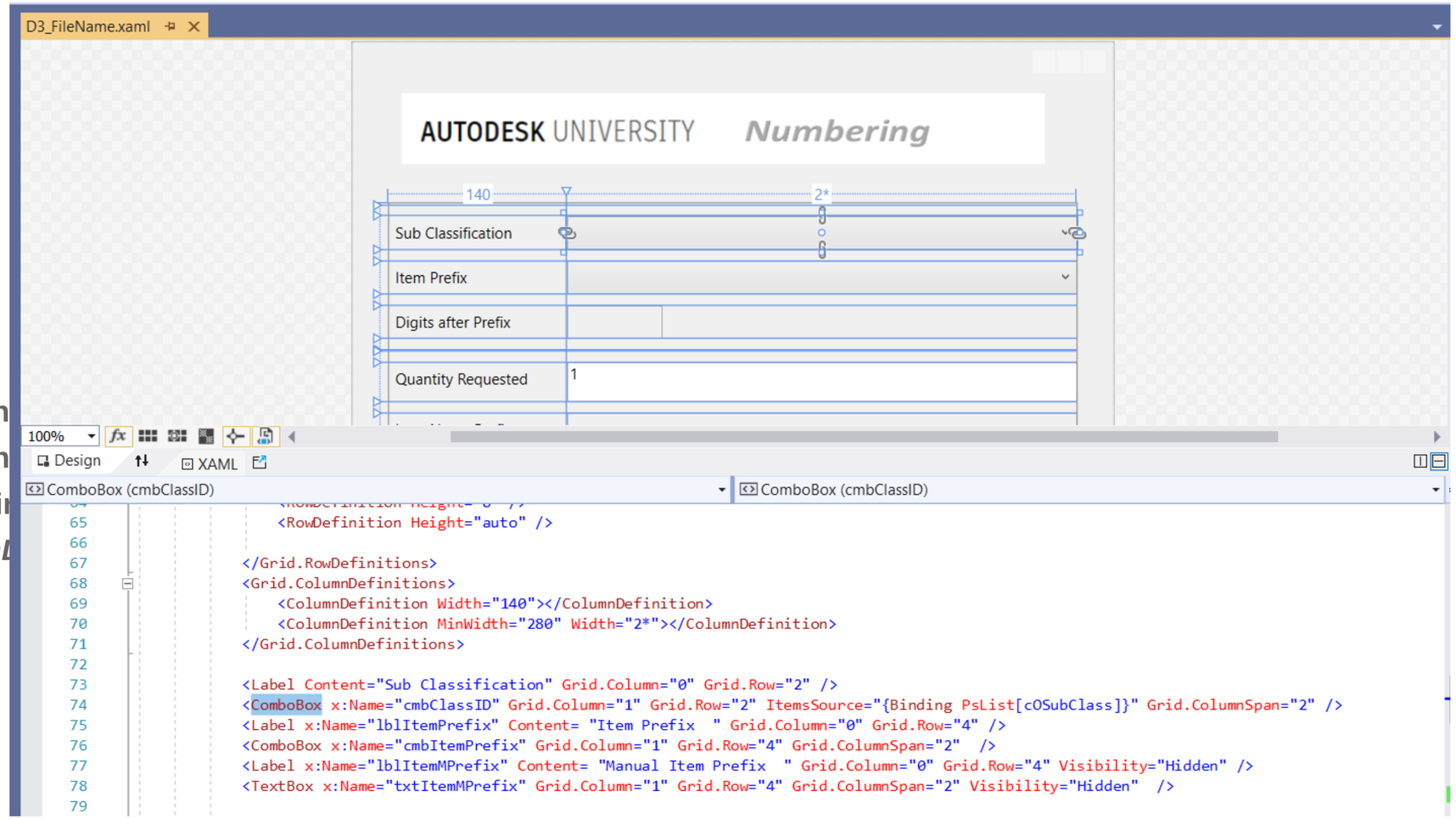
File Edit Format View Help

```xml
<?xml version="1.0" encoding="utf-8"?>

<!--
Label: Label string for the command, must be quoted
Description: Description text for the menucommand. Must be quoted
Hint: Hint string, must be quoted
PSFile: Name of the *ps1 file that contains the powershell script to execute when the menu command is invoked, value must be quoted.
Image: Name of a bitmap file that contains the image for the menu command. Image should be 16x16 or 32x32, value must be quoted.
ToolbarPaintStyle: Controls how the item is drawn on toolbars, values are Text, Glyph, TextAndGlyph
NavigationTypes: comma separated list of entities for which this command should be active. Values are Bom, ChangeOrder, File, FileVersion, Folder, Item, Other
MultiSelectEnabled: If true, command is active for multiple selections, can be true or false
-->
<mymenu>
  <MenuItem>
    <NewTask Label="$UIString[MNU22]" Description="$UIString[MNU17]" Hint="$UIString[MNU18]"
             PSFile="CreateCustomObject.ps1" Image="NewCustomObject.ico" ToolbarPaintStyle="TextAndGlyph"
             NavigationTypes="Task" MultiSelectEnabled="False" />
    <EditTask Label="$UIString[MNU21]" Description="$UIString[MNU19]" Hint="$UIString[MNU20]"
              PSFile="EditCustomObject.ps1" Image="EditCustomObject.ico" ToolbarPaintStyle="TextAndGlyph"
              NavigationTypes="Task" MultiSelectEnabled="False" />
    <NewLegacyECN Label="New Legacy ECN Entry" Description="New Legacy ECN Entry" Hint="New Legacy ECN Entry"
             PSFile="CreateCustomObject.ps1" Image="NewCustomObject.ico" ToolbarPaintStyle="TextAndGlyph"
             NavigationTypes="Task" MultiSelectEnabled="False" />
    <EditLegacyECN Label="Edit Legacy ECN Entry" Description="Edit Legacy ECN Entry" Hint="Edit Legacy ECN Entry"
              PSFile="EditCustomObject.ps1" Image="EditCustomObject.ico" ToolbarPaintStyle="TextAndGlyph"
              NavigationTypes="Task" MultiSelectEnabled="False" />

    <NewFile Label="$UIString[MNU1]" Description="$UIString[MNU11]" Hint="$UIString[MNU7]" PSFile="CreateFile.ps1"
             Image="NewStandardFile.ico" ToolbarPaintStyle="TextAndGlyph" NavigationTypes="File,Folder"
             MultiSelectEnabled="False" />
    <EditFile Label="$UIString[MNU2]" Description="$UIString[MNU12]" Hint="$UIString[MNU8]" PSFile="EditFile.ps1"
              Image="EditFileDatasheet.ico" ToolbarPaintStyle="TextAndGlyph" NavigationTypes="File"
              MultiSelectEnabled="False" />
    <NewFolder Label="$UIString[MNU3]" Description="$UIString[MNU13]" Hint="$UIString[MNU9]" PSFile="CreateFolder.ps1"
               Image="NewStandardFolder.ico" ToolbarPaintStyle="TextAndGlyph" NavigationTypes="Folder"
               MultiSelectEnabled="False" />
    <EditFolder Label="$UIString[MNU4]" Description="$UIString[MNU14]" Hint="$UIString[MNU10]" PSFile="EditFolder.ps1"
```

# Three parts of Vault Data Standards

## DIALOG FILES - XAML

**Dialog files are written**
**for describing a GUI. In**
**use for interacting with**
**XAML, Microsoft is goi**
**Location: "*C:\ProgramD***



# Three parts of Vault Data Standards

Windows PowerShell ISE

File Edit View Tools Debug Add-ons Help

Reservation.ps1 | Functions.ps1 | D3_General_Functions.ps1

```
103  function D3CommitNumbers
104  {
105
106  $vltUser = $VaultConnection.UserName
107  $FinalComment = "Reserved by " + $vltUser
108
109  $F_Prefix =  $dsWindow.FindName("cmbItemPrefix").SelectedValue
110  $F_SubClass =  $dsWindow.FindName("cmbClassID").SelectedValue
111
112  $F_DwgPrefix = $dsWindow.FindName("txtItemPrefix").Text
113  $F_Qty = [int]$dsWindow.FindName("txtQTY").text
114
115  $F_M_Prefix = $dsWindow.FindName("txtItemMPrefix").Text
116  if ($F_M_Prefix.Length -gt 1){
117   $F_Prefix = $F_M_Prefix}
118
119   $D3ItemList = $dsWindow.FindName("cmbFileList").Items
120
121
122
123  if ($D3ItemList.Count -eq 0) {
124       $wshell = New-Object -ComObject Wscript.Shell
125       $wshell.Popup("You Must Generate Item Number List First",0,"Done",0x1)
126       Break
127       }
128
129
130
131
```

POWERSHELL SCRI

**PowerShell is availa**

**Locations vary on v**

**Menu PS1: "*C:\Prog***

**Function PS1: "*C:\F***

# Folder Structure of Data Standards

**C:\ProgramData\Autodesk\Vault 2021\Extensions\DataStandard**

- **\Cad**
- **\Vault**

The \Cad and \Vault folders are the OOB solutions, and should be left alone, if you are to modify anything the files should be copied to the appropriate .Custom Folder

- **\Cad.Custom**
  - **\addins**
  - **\Configuration**

- **\Vault.Custom**
  - **\addinVault**
    - **\Menus**
  - **\Configuration**
    - **\Eco**
    - **\File**
    - **\Folder**
    - **\Item**
    - **\(any custom object you name)**

Any PowerShell files in the \addinVault folder will get pre-loaded when vault opens, so all functions are available

The sub \Menus folder is where are PowerShell files that a Menu command executes are loaded

The \Configuration folder is where all XAML files are stored, if a menu PS1 is calling the XAML file is in the root of this folder, if a new TAB is desired for any of the \ECO \File ...etc data types the XAML file is in its sub folder.

# The Fun Stuff

# Creating the Apps and Utilities for your environment

# Use cases we will explore

## ADDING TABS ON THE DETAILS OF A FILE, ITEM, CHANGE ORDER, OR CUSTOM OBJECT

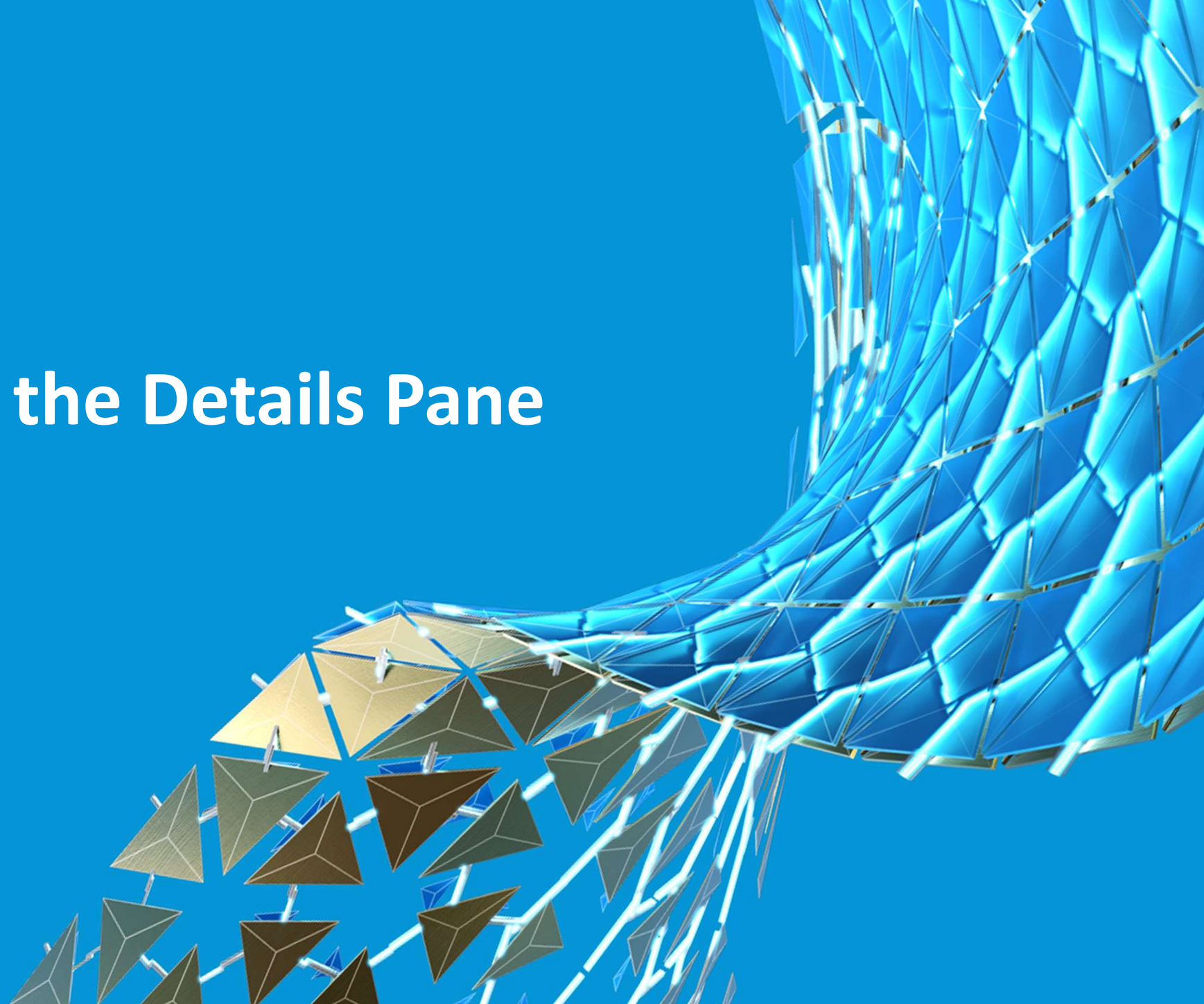
**By creating an XAML file and placing it in the proper location you can add Tabs to the detail section of the Vault interface that are specific to the object type you select. We will explore some use cases.**

## COMPLICATED FILE/ITEM NUMBERING SYSTEMS

**File naming / number systems can be complicated and may not fit the Vault Numbering Scheme workflow. We will explore the use of Custom Objects to define and create Files/Items with meaningful unique names.**

# The Fun Stuff

# Adding a Tab to the Details Pane



# Adding a Tab to the Details Pane

Curious about what jobs are queued for a particular file? Let’s add a tab for that.

**Create an XAML file called Jobs, I usually copy the Datasheet.XAML file, and then edit it, so the headers are all correct.**

```
<DataGrid x:Name="dtGrdJobs" AutoGenerateColumns="False" IsReadOnly="True" HorizontalGridLinesBrush="#FFABADB3" VerticalGridLinesBrush="#FFABADB3" Grid.Row="2" Grid.ColumnSpan="1" M
    <DataGrid.Columns>
        <DataGridTextColumn Binding="{Binding CreateDate}" Width="Auto" MinWidth="140">
            <DataGridTextColumn.HeaderTemplate>
                <DataTemplate>
                    <TextBlock Text="{Binding DataContext.UIString[LBL33], RelativeSource={RelativeSource AncestorType={x:Type DataGrid}}, FallbackValue='Date Created'}"/>
                </DataTemplate>
            </DataGridTextColumn.HeaderTemplate>
        </DataGridTextColumn>
        <DataGridTextColumn Binding="{Binding CreateUserName}" Width="*" MinWidth="200">
            <DataGridTextColumn.HeaderTemplate>
                <DataTemplate>
                    <TextBlock Text="{Binding DataContext.UIString[LBL34], RelativeSource={RelativeSource AncestorType={x:Type DataGrid}},FallbackValue='Created by'}"/>
                </DataTemplate>
            </DataGridTextColumn.HeaderTemplate>
        </DataGridTextColumn>
        <DataGridTextColumn Binding="{Binding Description}" Width="*" MinWidth="200">
            <DataGridTextColumn.HeaderTemplate>
                <DataTemplate>
                    <TextBlock Text="{Binding DataContext.UIString[LBL3], RelativeSource={RelativeSource AncestorType={x:Type DataGrid}},FallbackValue='Description'}"/>
                </DataTemplate>
            </DataGridTextColumn.HeaderTemplate>
```

Full instructions for this example found in Markus Koechl’s AKN Document
https://knowledge.autodesk.com/support/vault-products/getting-started/caas/simplecontent/content/vault-data-standard-job-list-tab.html

# Adding a Tab to the Details Pane

## Write the PowerShell Functions

**In this case there are two parts to the scripting**

**1. You must tell Vault what to do when the Tab contents change this is handled in the Default.PS1 in a function called “OnTabContextChanged “ an if statement is added to include the Jobs.xaml referenece**

```
if ($VaultContext.SelectedObject.TypeId.SelectionContext -eq "FileMaster" -and $xamlFile -eq "Jobs.xaml")
    {
        $fileMasterId = $vaultContext.SelectedObject.Id
        $file = $vault.DocumentService.GetLatestFileByMasterId($fileMasterId)
        $m_JobData = @(mFileJobList($file.Name))
        $dsWindow.FindName("dtGrdJobs").ItemsSource = $m_JobData
        If ($m_JobData) { $dsWindow.FindName("txtJobQueue").Visibility = "Collapsed"}
        Else { $dsWindow.FindName("txtJobQueue").Visibility = "Visible"}
    }
}
```

# Adding a Tab to the Details Pane

## Write the PowerShell Functions

**In this case there are two parts to the scripting**

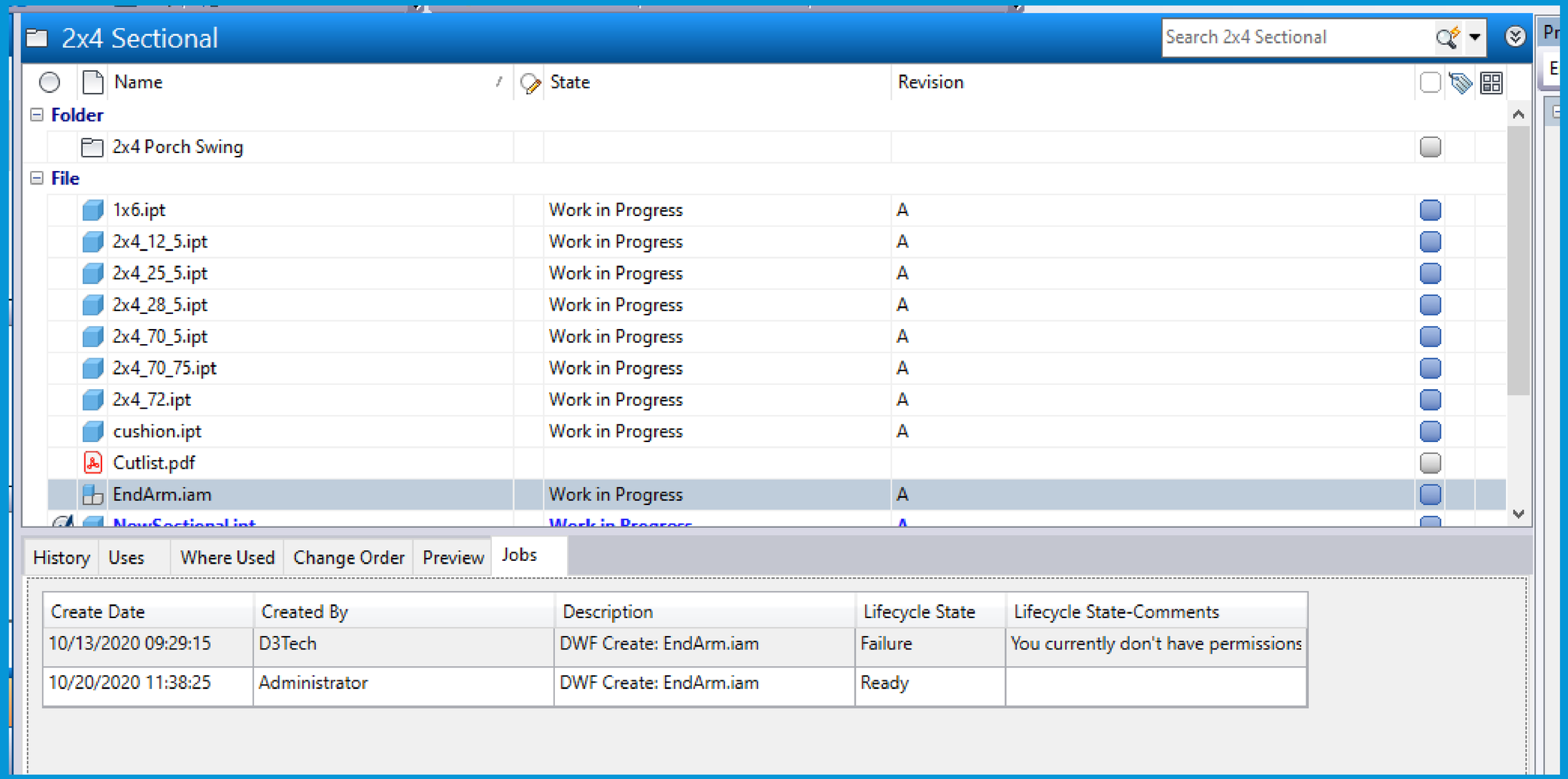
**2. Create the function referenced in the “OnTabContextChanged “ mFileJobList**

**This Function is included in a .PS1 file located in the C:\ProgramData\Autodesk\Vault 2021\Extensions\DataStandard\Vault.Custom\addinVault folder**

```
Add-Type @"
public class mJobData
{
    public string CreateDate {get;set;}
    public string CreateUserName {get;set;}
    public string Description {get;set;}
    public string StatusCode {get;set;}
    public string StatusMsg {get;set;}
}
"@

function mFileJobList([STRING] $mFileName) {
    $mJobMaxCount = 100
    $mJobStartDate = Get-Date -Day 01 -Month 01 -Year 2010
    $mFileName = "*" + $mFileName
    $mJobList = $vault.JobService.GetJobsByDate($mJobMaxCount, $mJobStartDate)

    $_TableData = @()
    foreach($mJob in $mJobList)
    {
        IF ($mJob.Descr -like $mFileName ) {
            $row = New-Object mJobData
            $row.CreateDate = $mJob.CreateDate
            $row.CreateUserName = $mJob.CreateUserName
            $row.Description = $mJob.Descr
            $row.StatusCode = $mJob.StatusCode
            $row.StatusMsg = $mJob.StatusMsg
            $_TableData += $row
        }
    }
    return $_TableData
}
```

# Adding a Tab to the Details Pane

Results



# The Fun Stuff

# Complicated File/Item Naming

# New App

## Scenario

**Item naming convention should be the Class Prefix plus an autogenerated number that must be unique and vary in length depending on the Class properties. While also allowing for a manual entry of a name, with a validation check for an existing Item.**

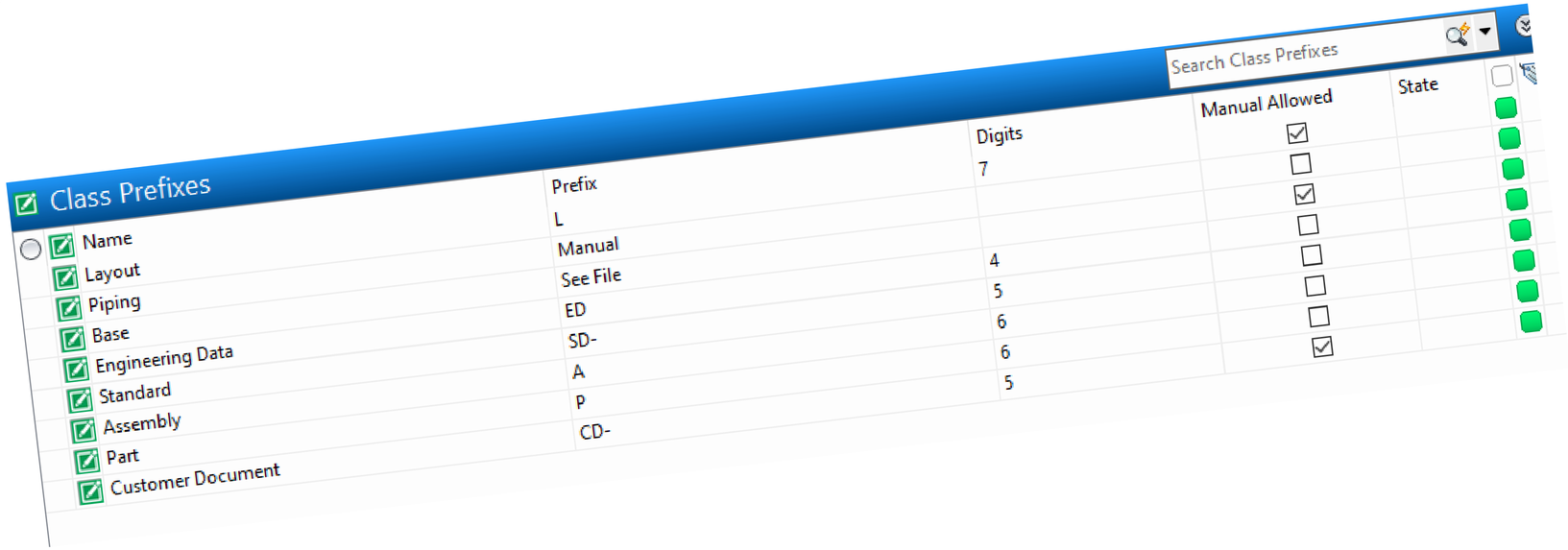
**Where is the information for the Class Prefix and all the relevant information stored?**

**Usually in an Excel document somewhere, or on sticky notes at everyone’s computers.**

# AU Numbering System

## Solution

- **Setup a Custom Object definition for the Class Prefix information (see Part 1)**
- **Create a Data Standards App that uses the information from the Custom Object**
- **Verifies unique names**
- **Generates the Item**



# AU Numbering System

## Create the Data Standards App

- **3 Parts**
  - Create a PS1 file in the Menus folder that calls the XAML, Create the Functions in a PS1 file in the AddinVault folder to execute various aspects of the Program
  - Create the XAML for the User Interface
  - Modify the MenuDefinitions.xml to create a new button on the Standard tool bar

# AU Numbering System

## PS1 to call the XAML

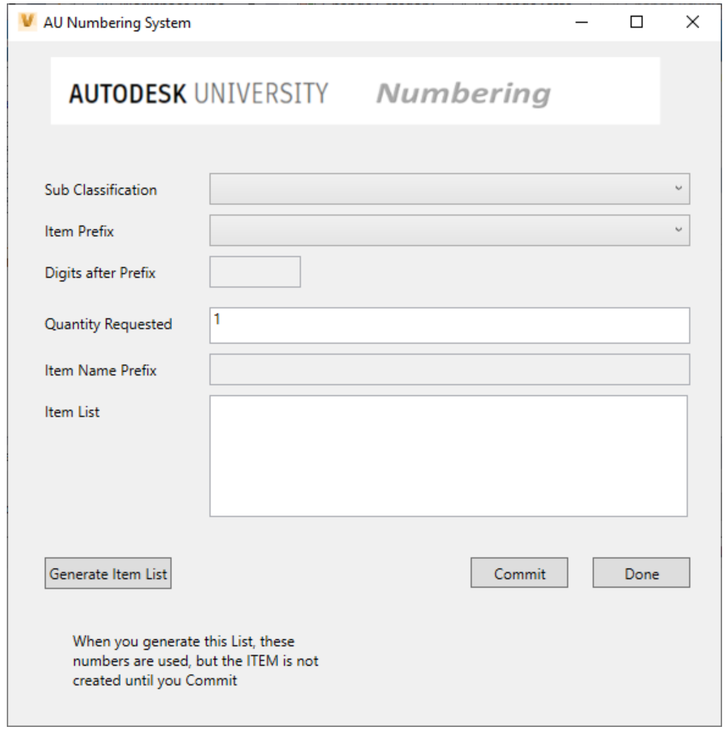
- **PowerShell to display the XAML**
  - File Name Reserve.PS1, located at: C:\ProgramData\Autodesk\Vault 2021\Extensions\DataStandard\Vault.Custom\addinVault\Menus

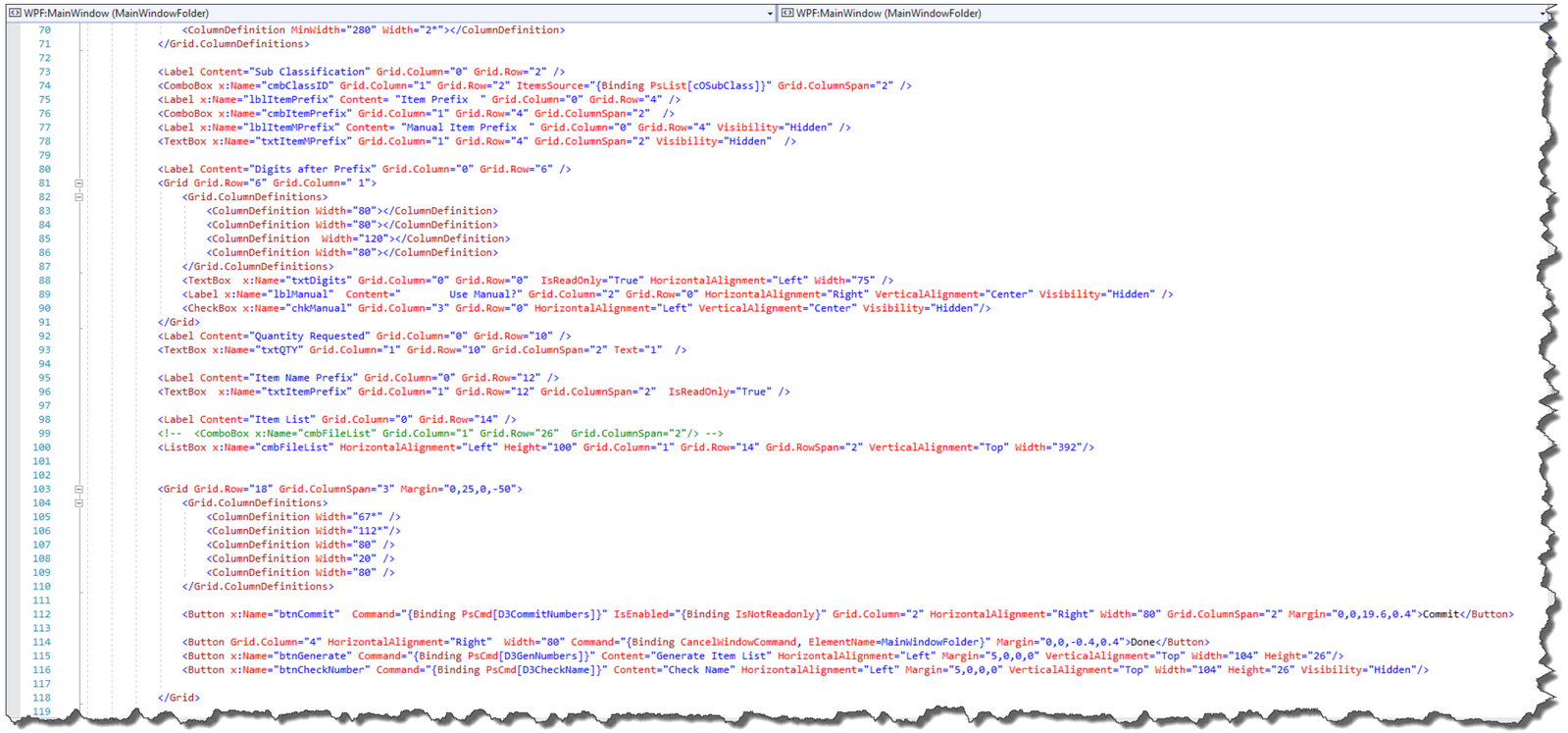
```
Import-Module -Name powerVault
$vaultContext.ForceRefresh = $true
$currentSelected = $vaultContext.CurrentSelectionSet[0]
$folderId = $currentSelected.Id
#if selected object is of type 'FILE' then use $vaultContext.NavSelectionSet[0].Id,
#it will give you back the folder Id where this file is located
if ($currentSelected.TypeId.EntityClassId -eq "FILE")
{
    $folderId = $vaultContext.NavSelectionSet[0].Id
}


$vYear = GetVaultVersion


$xamlFile = New-Object CreateObject.WPF.XamlFile "AU_FileName", "C:\ProgramData\Autodesk\Vault $($vYear)\Extensions\DataStandard\Vault.Custom\Configuration\D3_FileName.xaml"
$dialog = $dsCommands.GetCreateDialog($folderId)
$dialog.XamlFile = $xamlFile

#new file can be found in $dialog.CurrentFile
$result = $dialog.Execute()
$dsDiag.Trace($result)
```

# AU Numbering System

## the XAML

- **Created using Visual Studio 2019**
  - File Name D3_FileName.XAML, located at: C:\ProgramData\Autodesk\Vault 2021\Extensions\DataStandard\Vault.Custom\Configuration




```
70              <ColumnDefinition MinWidth="280" Width="2*"></ColumnDefinition>
71          </Grid.ColumnDefinitions>
72
73          <Label Content="Sub Classification" Grid.Column="0" Grid.Row="2" />
74          <ComboBox x:Name="cmbClassID" Grid.Column="1" Grid.Row="2" ItemsSource="{Binding PsList[cOSubClass]}" Grid.ColumnSpan="2" />
75          <Label x:Name="lblItemPrefix" Content= "Item Prefix  " Grid.Column="0" Grid.Row="4" />
76          <ComboBox x:Name="cmbItemPrefix" Grid.Column="1" Grid.Row="4" Grid.ColumnSpan="2"  />
77          <Label x:Name="lblItemMPrefix" Content= "Manual Item Prefix  " Grid.Column="0" Grid.Row="4" Visibility="Hidden" />
78          <TextBox x:Name="txtItemMPrefix" Grid.Column="1" Grid.Row="4" Grid.ColumnSpan="2" Visibility="Hidden"  />
79
80          <Label Content="Digits after Prefix" Grid.Column="0" Grid.Row="6" />
81          <Grid Grid.Row="6" Grid.Column=" 1">
82              <Grid.ColumnDefinitions>
83                  <ColumnDefinition Width="80"></ColumnDefinition>
84                  <ColumnDefinition Width="80"></ColumnDefinition>
85                  <ColumnDefinition  Width="120"></ColumnDefinition>
86                  <ColumnDefinition Width="80"></ColumnDefinition>
87              </Grid.ColumnDefinitions>
88              <TextBox  x:Name="txtDigits" Grid.Column="0" Grid.Row="0"  IsReadOnly="True" HorizontalAlignment="Left" Width="75" />
89              <Label x:Name="lblManual"  Content="          Use Manual?" Grid.Column="2" Grid.Row="0" HorizontalAlignment="Right" VerticalAlignment="Center" Visibility="Hidden" />
90              <CheckBox x:Name="chkManual" Grid.Column="3" Grid.Row="0" HorizontalAlignment="Left" VerticalAlignment="Center" Visibility="Hidden"/>
91          </Grid>
92          <Label Content="Quantity Requested" Grid.Column="0" Grid.Row="10" />
93          <TextBox x:Name="txtQTY" Grid.Column="1" Grid.Row="10" Grid.ColumnSpan="2" Text="1"  />
94
95          <Label Content="Item Name Prefix" Grid.Column="0" Grid.Row="12" />
96          <TextBox  x:Name="txtItemPrefix" Grid.Column="1" Grid.Row="12" Grid.ColumnSpan="2"  IsReadOnly="True" />
97
98          <Label Content="Item List" Grid.Column="0" Grid.Row="14" />
99          <!--  <ComboBox x:Name="cmbFileList" Grid.Column="1" Grid.Row="26"  Grid.ColumnSpan="2"/> -->
100         <ListBox x:Name="cmbFileList" HorizontalAlignment="Left" Height="100" Grid.Column="1" Grid.Row="14" Grid.RowSpan="2" VerticalAlignment="Top" Width="392"/>
101
102
103         <Grid Grid.Row="18" Grid.ColumnSpan="3" Margin="0,25,0,-50">
104             <Grid.ColumnDefinitions>
105                 <ColumnDefinition Width="67*" />
106                 <ColumnDefinition Width="112*"/>
107                 <ColumnDefinition Width="80" />
108                 <ColumnDefinition Width="20" />
109                 <ColumnDefinition Width="80" />
110             </Grid.ColumnDefinitions>
111
112             <Button x:Name="btnCommit"  Command="{Binding PsCmd[D3CommitNumbers]}" IsEnabled="{Binding IsNotReadonly}" Grid.Column="2" HorizontalAlignment="Right" Width="80" Grid.ColumnSpan="2" Margin="0,0,19.6,0.4">Commit</Button>
113
114             <Button Grid.Column="4" HorizontalAlignment="Right"  Width="80" Command="{Binding CancelWindowCommand, ElementName=MainWindowFolder}" Margin="0,0,-0.4,0.4">Done</Button>
115             <Button x:Name="btnGenerate" Command="{Binding PsCmd[D3GenNumbers]}" Content="Generate Item List" HorizontalAlignment="Left" Margin="5,0,0,0" VerticalAlignment="Top" Width="104" Height="26"/>
116             <Button x:Name="btnCheckNumber" Command="{Binding PsCmd[D3CheckName]}" Content="Check Name" HorizontalAlignment="Left" Margin="5,0,0,0" VerticalAlignment="Top" Width="104" Height="26" Visibility="Hidden"/>
117
118         </Grid>
119
```

# AU Numbering System

## the PowerShell

- **Created using Windows PowerShell ISE**
  - File Names D3_General_Functions.ps1, Functions.ps, Reservations.ps1
  - located at: C:\ProgramData\Autodesk\Vault 2021\Extensions\DataStandard\Vault.Custom\addinVault

- **D3_General_Functions.ps1**
  - Contains general reusable functions collected over a period of time
    - GetNextNumber – calls the Numbering Schemes from Vault
    - D3CreateItem – Creates a new Item with an assigned number and category
    - Various Property gathering functions

# AU Numbering System

## the PowerShell

- **Created using Windows PowerShell ISE**
  - File Names D3_General_Functions.ps1, Functions.ps, Reservations.ps1
  - located at: C:\ProgramData\Autodesk\Vault 2021\Extensions\DataStandard\Vault.Custom\addinVault

  - **Functions.ps1**
    - Contains functions related to the User Interface
      - coSubClass – Pulls out the Custom Object information
      - coItemDigits – Gets the Number of Digits from the selected Class
      - cOgetItemPrefix – Gets the Prefix from the selected Class
      - D3DialogREsset – hides and displays various UI components based on Manual CheckBox
      - d3DialogReaction – makes one pull down populate another using events

# AU Numbering System

## the PowerShell

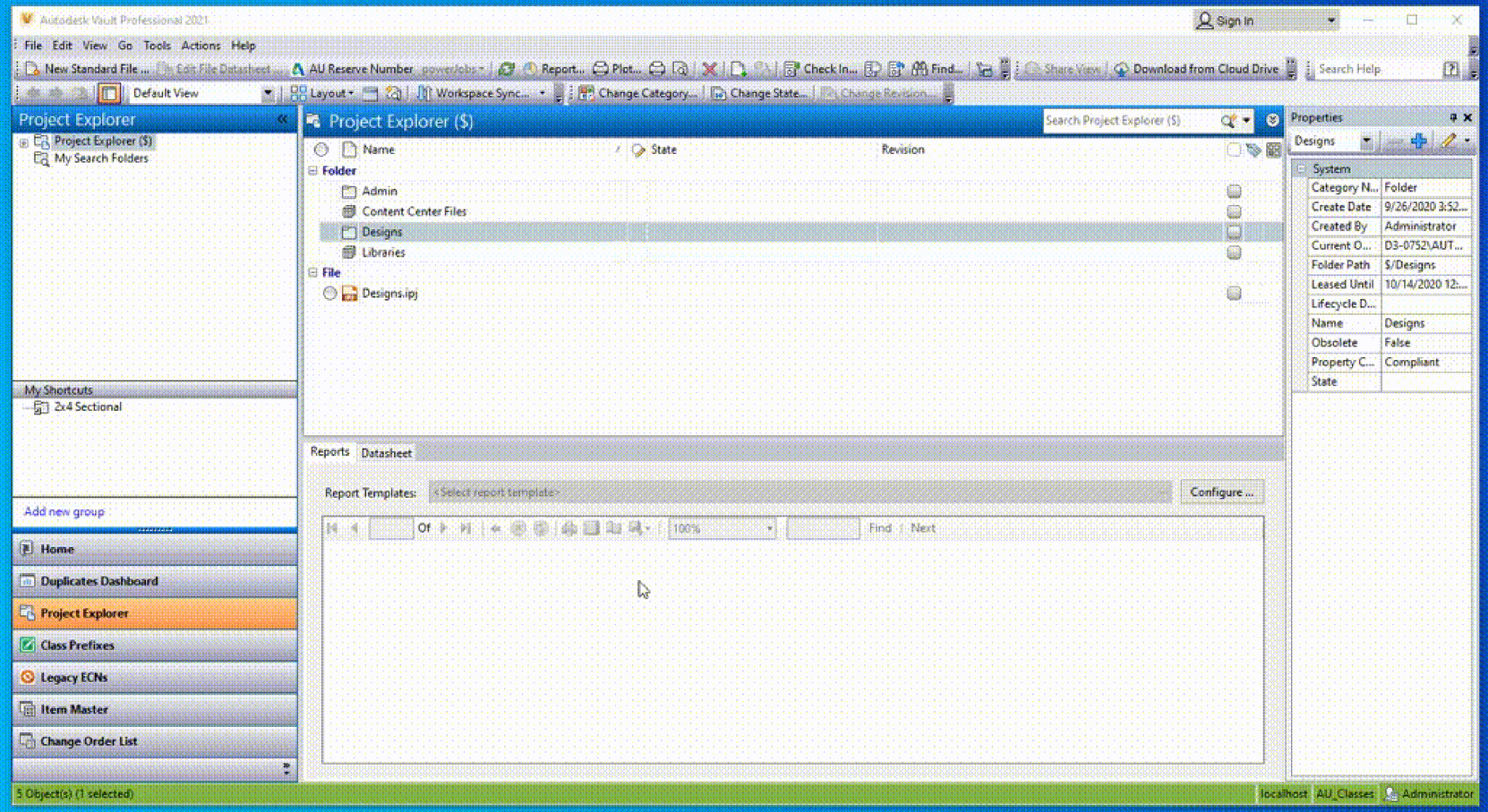
- **Created using Windows PowerShell ISE**
  - File Names D3_General_Functions.ps1, Functions.ps, Reservations.ps1
  - located at: C:\ProgramData\Autodesk\Vault 2021\Extensions\DataStandard\Vault.Custom\addinVault

- **Reservations.ps1**
  - Contains general the executable functions to get a number, and create the Item
    - D3GenNumbers – gathers selections from the UI, and calls the Numbering Scheme
    - D3CheckName – Verifies a unique name if the Manual selection is chosen
    - D3CommitNumbers/D3CommitNumbers_Manual – creates the Items

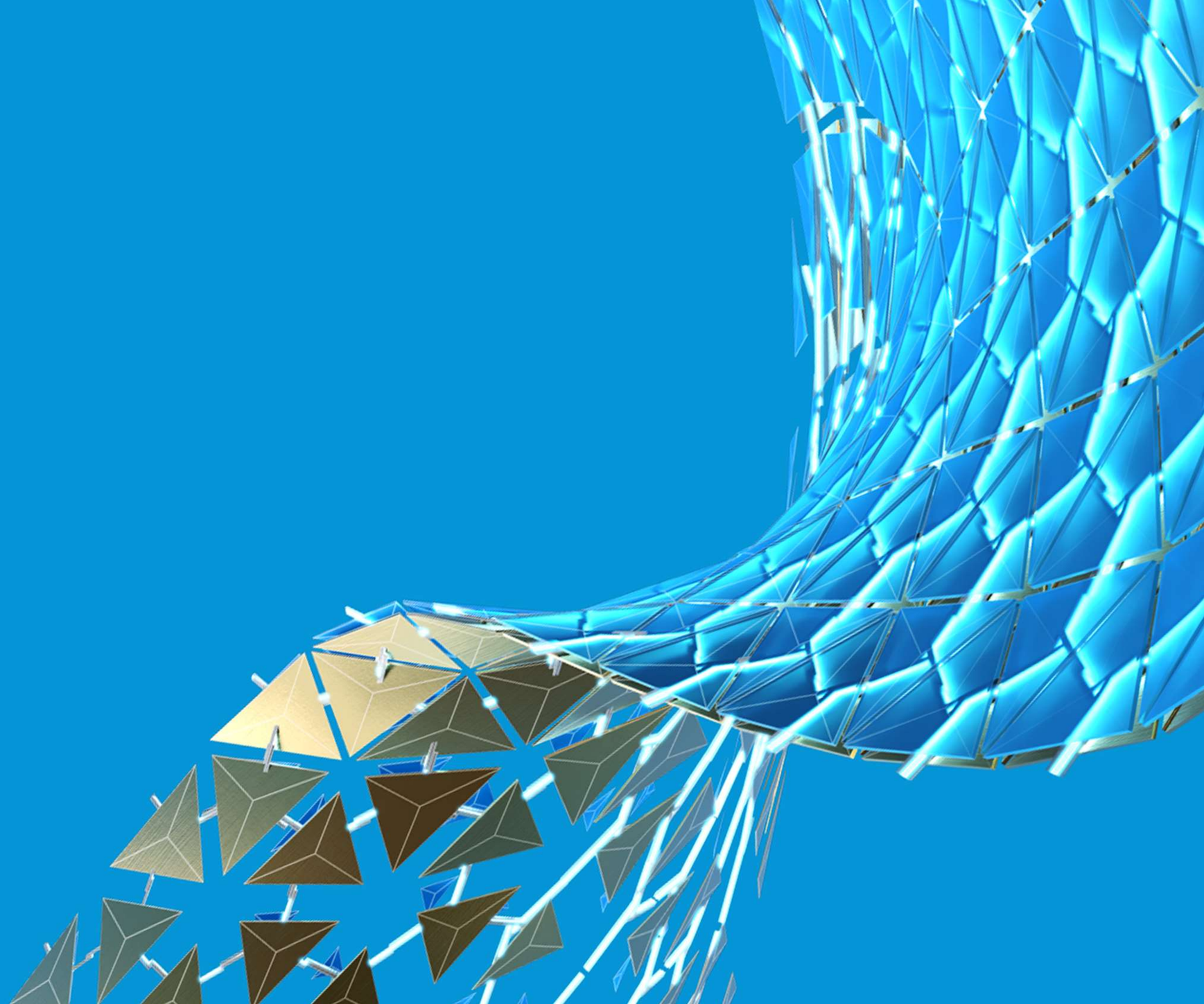
# AU Numbering System

Pulling it all together

Live Demo – Video below for reference



# Conclusion



# Conclusion

## Recap

**Data Standards can be used for any number of applications I hope that I have opened your eyes to a few of the opportunities that are available**

## Resources

- **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://download.coolorange.com/#powerVault**
- **Code snippets from this class will be available on the AU website**