

MFG467346

# [Vault-Configure don't Customize - the Power of Data Standards (Part 2 of 2)

Kimberley Hendrix  
D3 Technologies, LLC

## Learning Objectives

- Discover quality use cases for Data Standard configurations
- Learn how to create effective utilities with Data Standards
- Learn how to solve irritating discrepancies in your data with effective use of Data Standards
- Learn how to easily deploy and maintain Data Standard throughout your organization

## Description

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

## Speaker(s)

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 a solutions consultant, focusing on data management and the customization of applications to streamline access to data across an organization

## What is 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 Use Data Standards

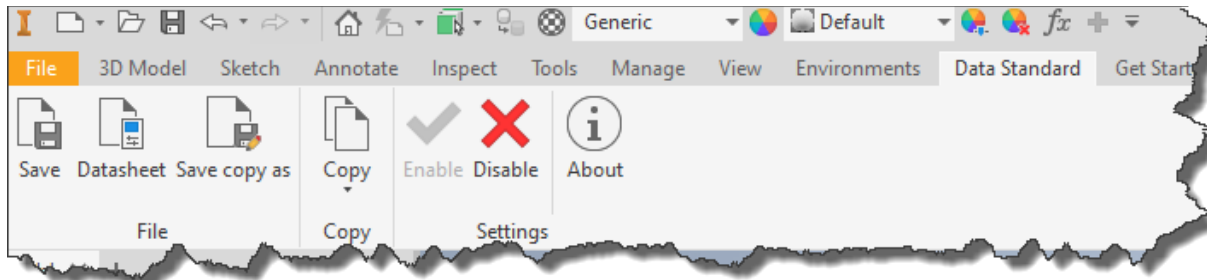
- It is provided with Autodesk Vault Clients both Workgroup and Professional.
- Customizations afforded with Data Standards do not require reworking and compiling year over year.
- The language used to configure Data Standards automation is PowerShell, available with all Windows installations.
- The user interface, or dialog boxes are written in XAML format
- 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 editor.
- The next years upgrade will generally only require copying the files to a new location:
- For example, from C:\ProgramData\Autodesk\Vault 2020\Extensions\DataStandard  
to  
C:\ProgramData\Autodesk\Vault 2021\Extensions\DataStandard

## Learning Objectives

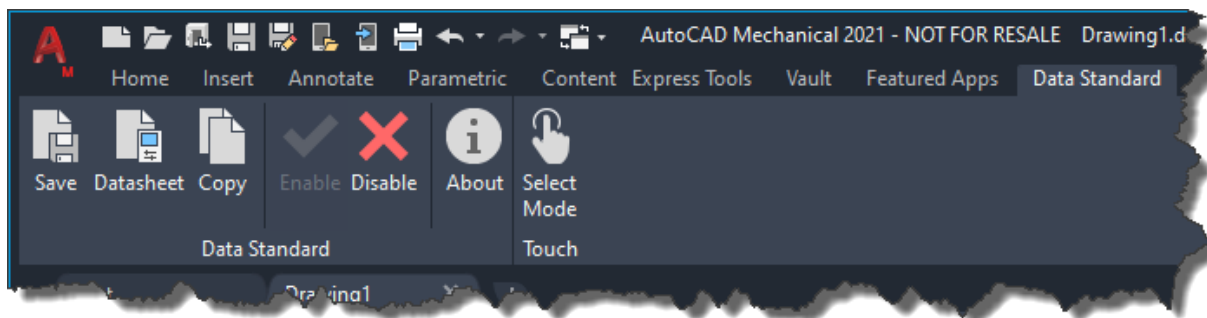
### Discover use cases for Data Standards

Before we get into any configured use cases for Data Standards lets go over what you get from Data Standards Out of the Box (OOB).

### Inventor & AutoCAD



*INVENTOR DATA STANDARD RIBBON*



*AUTOCAD DATA STANDARD RIBBON*

Out of the Box Data Standards without any modifications includes

- Save (custom dialog)
- DataSheet
- Save Copy As (Inventor only)
- Copy
- Disable

The dialogs for each Inventor and AutoCAD are very similar the differences being only around the amount of properties that are available.

### **SAVE**

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

New File - Part2.ipt

Folder  
2x4 Sectional 2x4 Porch Swing

Category Engineering  
Number Scheme  
Number  
File Name New PartName  
Comments

Property Name Property Value  
Description  
Mfg Approved by  
Engr Approved by  
Material Generic  
Cost 0.00  
Stock Number  
LastUpdatedWith 2021 (Build 250183000, 183)  
Designer kim.hendrix  
Engineer  
Checked by  
Author kim.hendrix  
Keywords  
Title  
Subject  
Comments

Path  
C:\\_AUClasses\Designs\2x4 Sectional\2x4 Porch Swing\

OK Cancel

INVENTOR SAVE DIALOG WITH DATA STANDARDS

### 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.

Save copy as - prop.ipt

Folder

2x4 Sectional
2x4 Porch Swing

Category	Engineering	Property Name	Property Value
Save As Type	.stp;*.ste;*.step;*.stpz	Description	
Number Scheme	None	Mfg Approved by	
Number		Engr Approved by	
File Name	prop.stp	Material	Generic
Comments		Cost	0.00
		Stock Number	
		LastUpdatedWith	2021 (Build 250183000, 183)
		Designer	kim.hendrix
		Engineer	
		Checked by	
		Author	kim.hendrix
		Keywords	
		Title	
		Subject	
		Comments	

Path

C:\\_AUClasses\Designs\2x4 Sectional\2x4 Porch Swing\

OK

Cancel

Options

SAVE COPY AS (INVENTOR ONLY)

### Copy

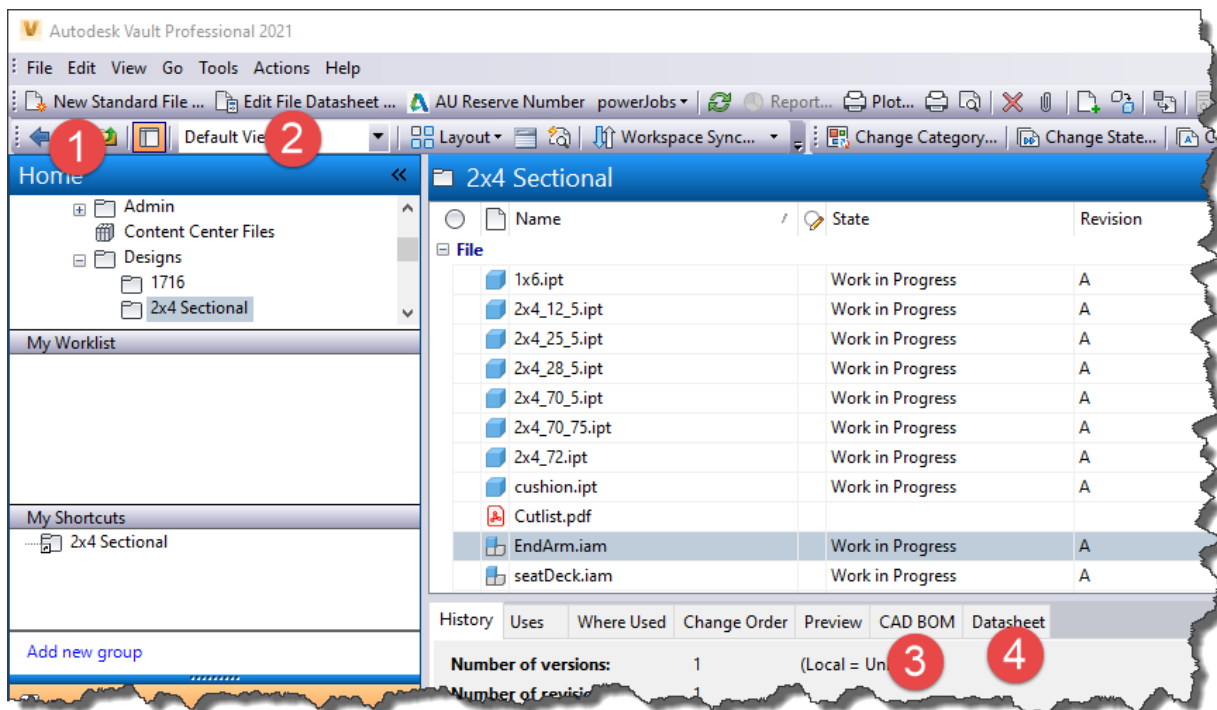
Simply makes a copy of the file in the same format.

### Disable

Allows for Data Standards to be turned off

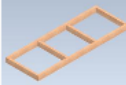

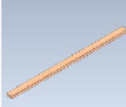

## Vault Client Data Standards (OOB)

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





*VAULT DATA STANDARDS (OUT OF THE BOX)*

## CAD BOM Tab

sectiona sofa.iam		Work in Progress		A			
sectiona sofa.idw		Work in Progress		A			
History	Uses	Where Used	Change Order	Preview	CAD BOM	Datasheet	
Thumbnail	Position	Part Number	Quantity	Component Type	Material	Title	Description
	0	seatDeck	2	Part			
	0	EndArm	3	Part			
	0	2x4-5	1	Part	Wood (Birch)		
	0	2x4-2	2	Part	Wood (Birch)		

*CAD BOM TAB IN VAULT CLIENT*

## Data Sheet Tab

	sectiona sofa.iam	Work in Progress	A
	sectiona sofa.idw	Work in Progress	A

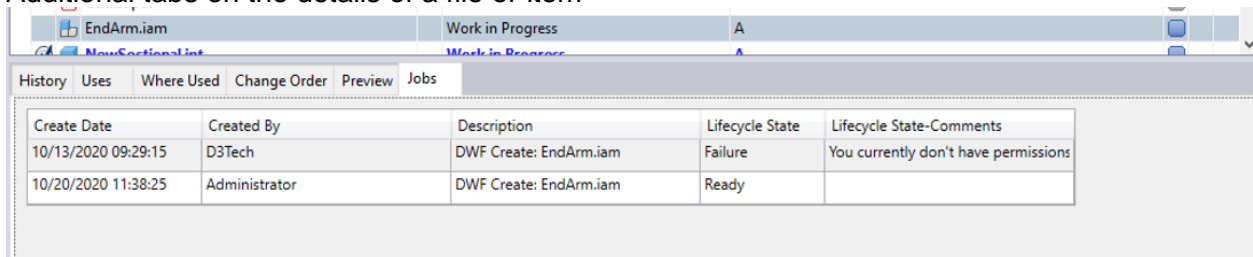
History	Uses	Where Used	Change Order	Preview	CAD BOM	Datasheet	
Category	Engineering			Property Name	Property Value		
Name	sectiona sofa.iam			Author	kim.hendrix		
State	Work in Progress			Mfg Approved By			
Create Date	9/26/2020 3:52:49 PM			Manager			
Created By	Administrator			Engr Approved By			
				Engineer			
				Stock Number			
				Material			
				Description			
				Cost			
				Checked By			

*DATA SHEET TAB VAULT CLIENT*

## Configured Use Cases for Data Standards

We will evaluate a couple of uses cases in this class.

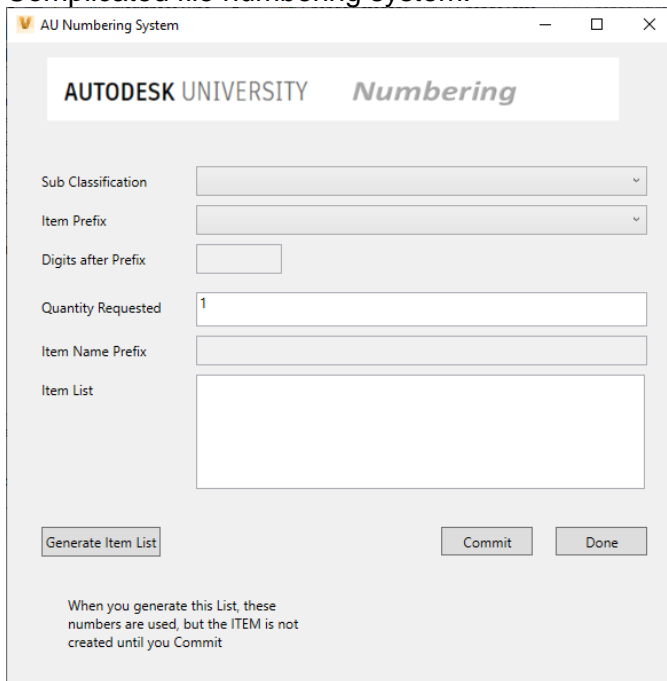
### 1. Additional tabs on the details of a file or item



Create Date	Created By	Description	Lifecycle State	Lifecycle State-Comments
10/13/2020 09:29:15	D3Tech	DWF Create: EndArm.iam	Failure	You currently don't have permissions
10/20/2020 11:38:25	Administrator	DWF Create: EndArm.iam	Ready	

*JOBS TAB ADDED TO VAULT CLIENT*

### 2. Complicated file numbering system.



**AUTODESK UNIVERSITY** *Numbering*

Sub Classification:

Item Prefix:

Digits after Prefix:

Quantity Requested:

Item Name Prefix:

Item List:

When you generate this List, these numbers are used, but the ITEM is not created until you Commit

*ITEM NUMBERING SYSTEM USING CUSTOM OBJECTS*

In order to configure these applications, we need to study the anatomy of Vault Data Standards.

## The Anatomy of Vault Data Standards

There are three main parts to Data Standards

- The Menu
- The User Interface
- The PowerShell

### The Menu

The menus are driven by a file MenuDefinitions.XML there is an OOB one stored in


*"C:\ProgramData\Autodesk\Vault YYYY\Extensions\DataStandard\Vault"*

When you are ready to modify this file to add your configurations to it you should COPY this file into the Vault.Custom folder, all modifications should be made there.

*"C:\ProgramData\Autodesk\Vault YYYY\Extensions\DataStandard\Vault.Custom"*

The <MenuItem> section describes what you are going to do and what it will look like.

```
<mymenu>
  <MenuItem>
    <ReserveNumber Label="AU Reserve Number" Description="Generate Number" Hint="Generate Number "
      PSFile="Reserve.ps1" Image="AU.ico" ToolbarPaintStyle="TextAndGlyph" NavigationTypes="File,Item"
      MultiSelectEnabled="False" />
```

- Label and the Icon file will make the menu button 
- Hint provides the hover text
- PSFile is what pressing the button executes
- Navigation Types are what object you can be on to activate the button
- MultiSelectEnabled allows for shift or ctrl select to effect multiple files

```
<CommandSite>
  <FileContext Label="myMenu" DeployAsPullDown="False" Location="FileContextMenu">
    <Item Name="NewFile"></Item>
    <Item Name="EditFile"></Item>
    <Item Name="ReserveNumber"></Item>
  </FileContext>
  <StandardToolbar Label="myMenu" DeployAsPullDown="False" Location="StandardToolbar">
    <Item Name="NewFile"></Item>
    <Item Name="EditFile"></Item>
    <Item Name="ReserveNumber"></Item>
  </StandardToolbar>
```

The <Command Site> section details where the menu option will be available the excerpt above shows just the FileContext and StandardToolbar in the actual file the options are

- FileContext
- FolderContext
- StandardToolbar

- ToolsMenu
- HlepMenu
- CustomObject (one for each defined Custom Object)

Each place that you want the new Menu to appear you add

`<Item Name = "ReserveNumber"></Item>`

*(ReserveNumber is the name you gave it in MenuSites above)*

The Standard, Tools or Help will show on the toolbars on the top of the Vault Client, the File, Folder or Custom Object will show on a Right Mouse click on that Object.

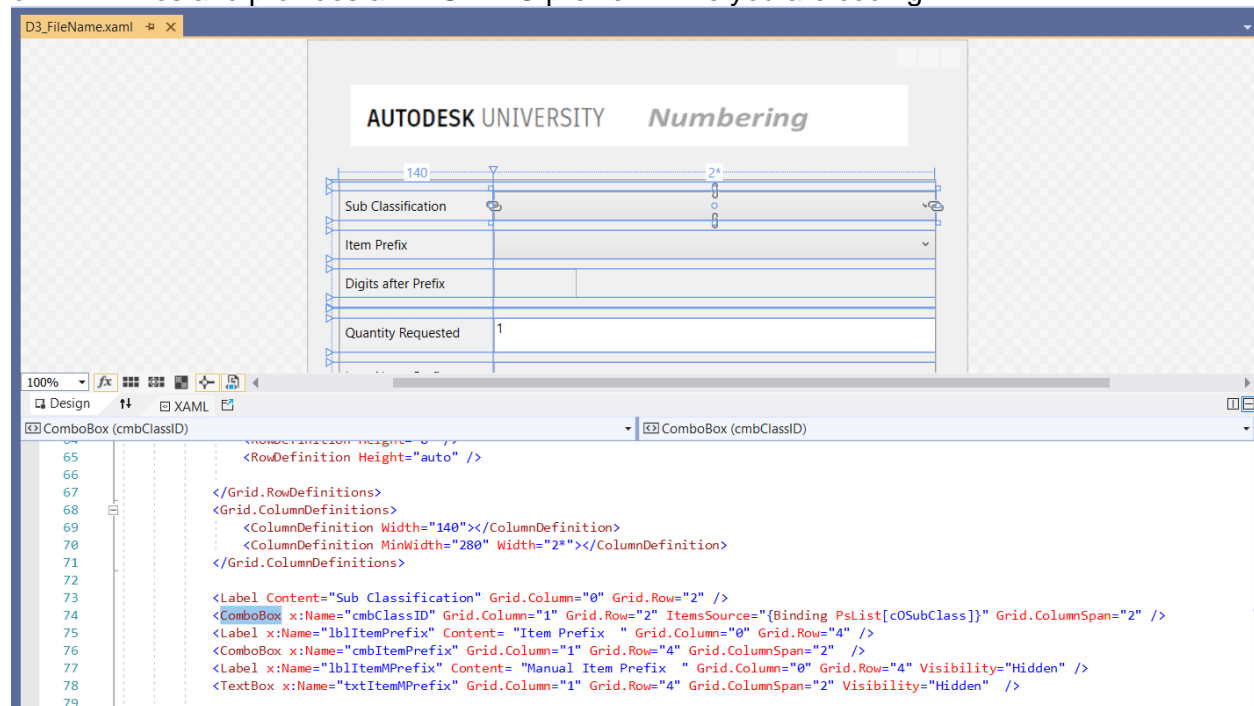
## The User Interface

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 can easily write and edit your GUI in a text editor.

Location:

`"C:\ProgramData\Autodesk\Vault YYYY\Extensions\DataStandard\Vault.Custom\Configuration"`

XAML files break up the dialog into a series of Grids, with as many Rows and Columns as you need to define the dialog. Grids can also be embedded into a Row/Column. In the Vault help there is a great section that defines how to setup Microsoft Visual Studio to help in the creating of XAML files and provides a WYSIWYG preview while you are coding.



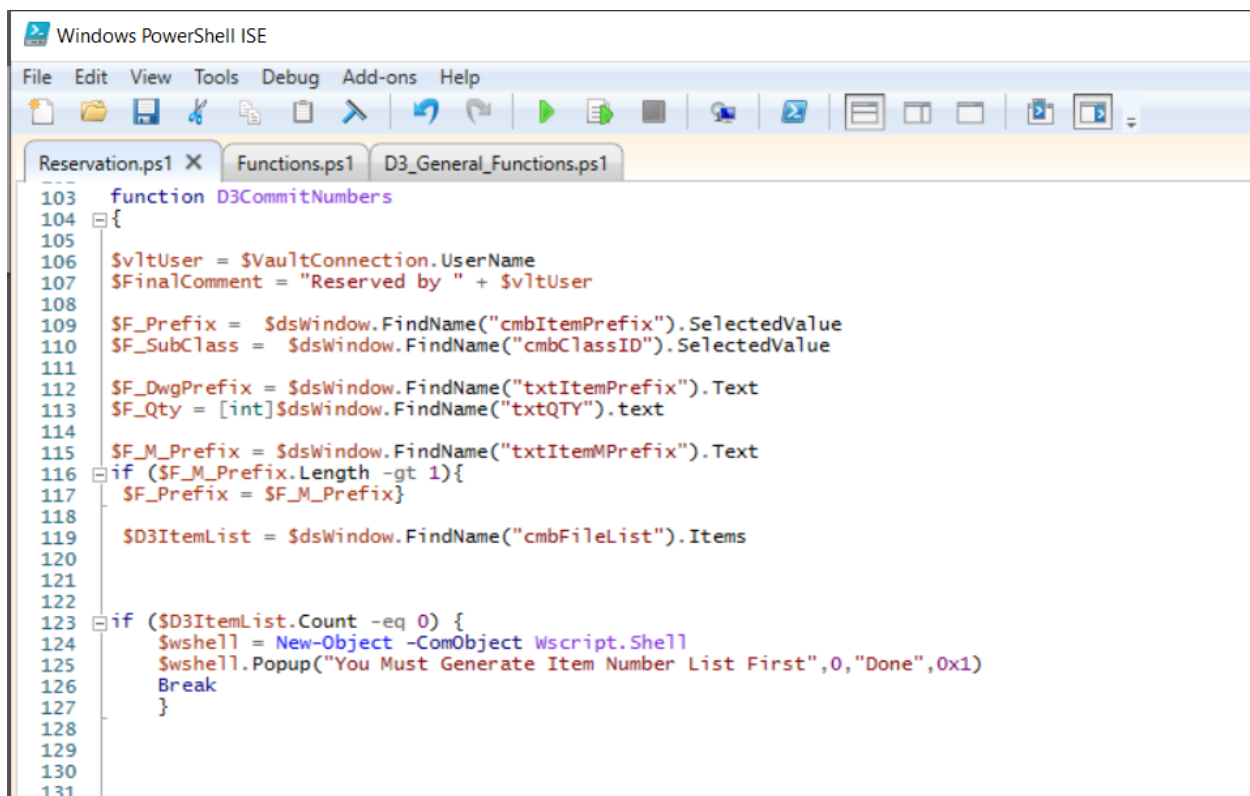
*XAML FILE CODE AND PREVIEW INSIDE VISUAL STUDIO*

## The PowerShell

PowerShell is an automated task framework from Microsoft, with a command line shell and a scripting language integrated into the .NET framework.

For Vault Data Standards typically two PowerShell scripts are required to execute a custom app.

- the PS1 called from the menu
  - C:\ProgramData\Autodesk\Vault YYYY\Extensions\DataStandard\Vault.Custom\addinVault\Menus
  - This is where the PS1 files reside that are identified in the MenuDefinitions.xml file
- the PS1 that contains functions called from the XAML
  - C:\ProgramData\Autodesk\Vault YYYY\Extensions\DataStandard\Vault.Custom\addinVault
  - PS1 file located in this directory are pre-loaded and contain functions that can be called from the XAML file or from each other.



```

103 function D3CommitNumbers
104 {
105
106     $vltUser = $VaultConnection.UserName
107     $FinalComment = "Reserved by " + $vltUser
108
109     $F_Prefix = $sdswindow.FindName("cmbItemPrefix").SelectedValue
110     $F_SubClass = $sdswindow.FindName("cmbClassID").SelectedValue
111
112     $F_DwgPrefix = $sdswindow.FindName("txtItemPrefix").Text
113     $F_Qty = [int]$sdswindow.FindName("txtQTY").text
114
115     $F_M_Prefix = $sdswindow.FindName("txtItemMPrefix").Text
116     if ($F_M_Prefix.Length -gt 1){
117         $F_Prefix = $F_M_Prefix
118     }
119     $D3ItemList = $sdswindow.FindName("cmbFileList").Items
120
121
122
123     if ($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 BEING EDITED IN WINDOWS POWERSHELL ISE EDITOR*

Windows comes with an editor Windows PowerShell ISE that aids in the coding of these files, there are other options as well, Visual Studio Code and others.

## Folder Structure

The Vault Data Standards add-in has a very specific folder structure that once you master makes configuring Data Standards much easier.

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

<b>\Cad</b>	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
<b>\Vault</b>	
<b>\Cad.Custom</b>	
<b>\addins</b>	
<b>\Configuration</b>	Any PowerShell files in the \addinVault folder will get pre-loaded when vault opens, so all functions are available
<b>\Vault.Custom</b>	
<b>\addinVault</b>	The sub \Menus folder is where are PowerShell files that a Menu command executes are loaded
<b>\Menus</b>	
<b>\Configuration</b>	
<b>\Eco</b>	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.
<b>\File</b>	
<b>\Folder</b>	
<b>\Item</b>	
<b>\(any custom object you name)</b>	

## Configure New Tab for Details Screen

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.

I am using an example from:  
Markus Koechl's AKN Document

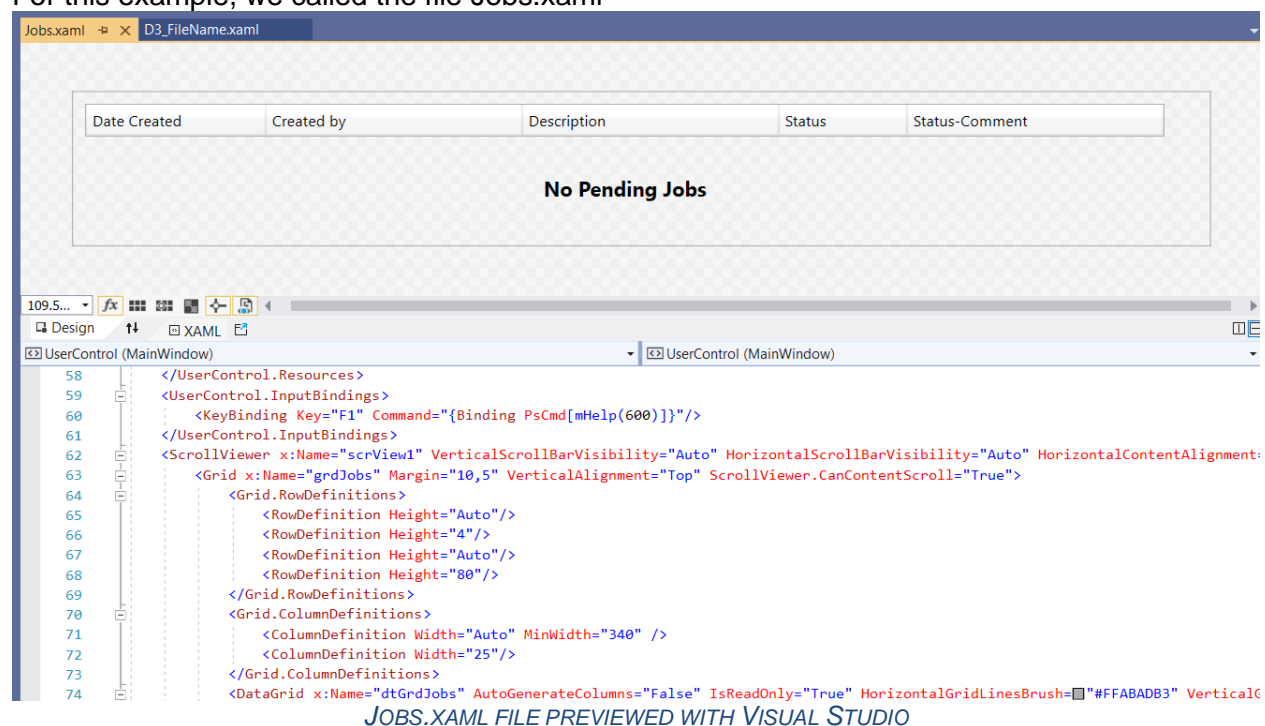
<https://knowledge.autodesk.com/support/vault-products/getting-started/caas/simplecontent/content/vault-data-standard-job-list-tab.html>

Use case I need to work on and modify a file but I am not sure if the job processor has pending or errored jobs on this file.

Create and place an XAML file in

`C:\ProgramData\Autodesk\Vault 2021\Extensions\DataStandard\Vault.Custom\Configuration\File`

For this example, we called the file `Jobs.xaml`



It is a good ideal to start with an XAML file that is provided, like the `DataSheet.xaml` located in the default configuration folder. It has all the headers setup properly.

So how does the vault know what to do when the Tabs change? You are on a file, on the History tab, and you select this new Job tab, the Vault needs to fire something.

In the `Default.PS1` file in the `C:\ProgramData\Autodesk\Vault`

`YYYY\Extensions\DataStandard\Vault\addin\Vault` folder has a function called

`"OnTabContextChanged"` if we subscribe to the only make changes in the `Vault.Custom` folder

than this default.ps1 should be copied to the Vault.Custom folder and then the following added to the function. You can modify right in that folder but beware if you uninstall/reinstall you could lose stuff in the Vault folder.

```
if ($VaultContext.SelectedObject.TypeId.SelectionContext -eq "FileMaster" -and $xmlFile -eq "Jobs.xml")
{
    $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"}
}
}
```

*EXCERPT FROM DEFAULT.PS1*

In excerpt above, in blue this ONTABCHANGE calls another function "mFileJobList" this function needs to be written and put in a PS1 file location in C:\ProgramData\Autodesk\Vault 2021\Extensions\DataStandard\Vault.Custom\addin\Vault In my example I created a Functions.PS1 and it includes the following.

```

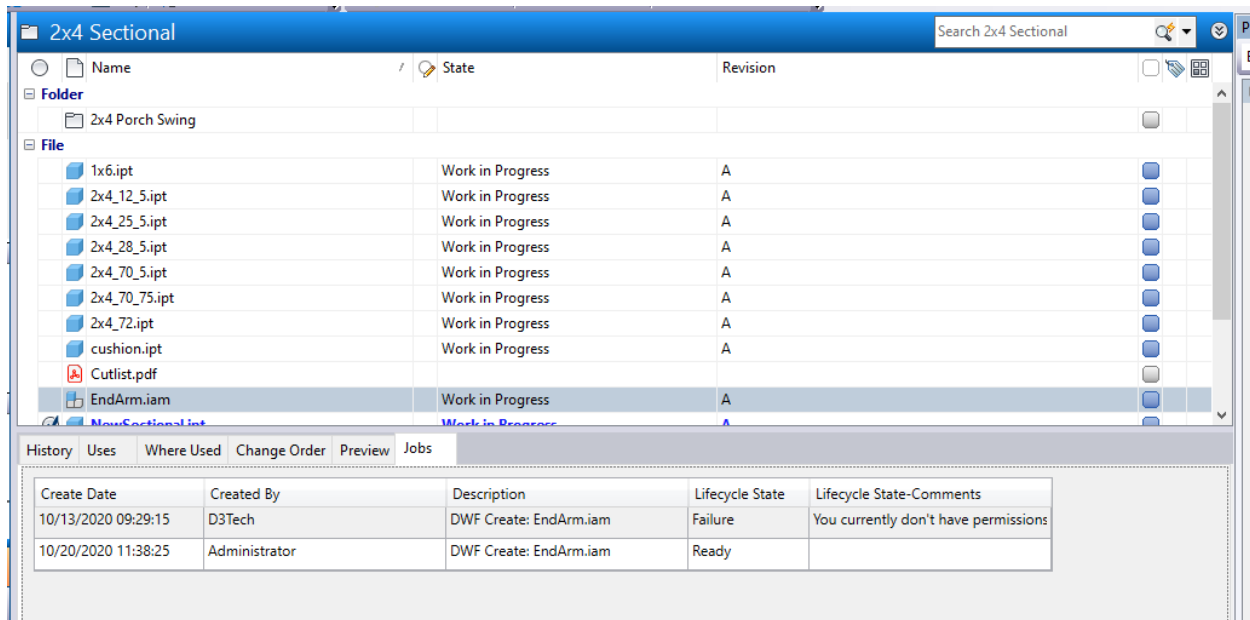
Add-Type @"
public class mJobData
{
    public string CreateDate {get;set;}
    public string CreateUserName {get;set;}
    public string Description {get;set;}
    public string StatusCode {get;set;}
    public string StatusMsg {get;set;}
}
"@

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
}
}
```

*EXCERPT FROM FUNCTIONS.PS1*

Breaking down this function, the first part the public class mJobData creates some public variables to make them available elsewhere. The actual function mFileJobList takes a filename as a parameter, then makes a call to the vault API "getJobsByDate" this will return all jobs in the job queue, then the foreach, will filter through all the jobs, and populate a table of the jobs that reference the filename.



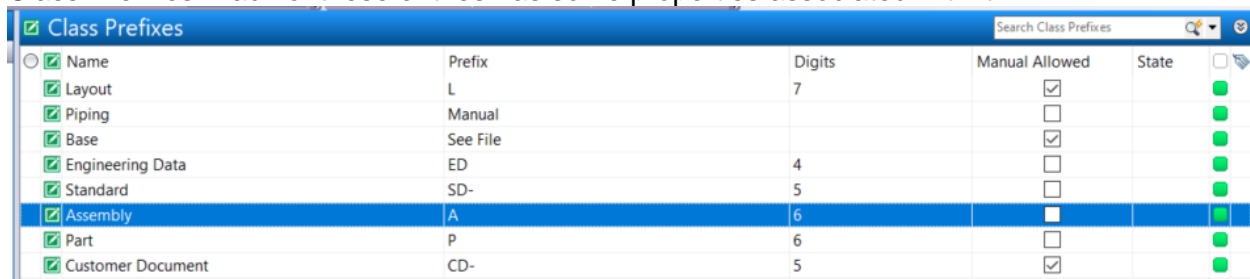
*JOBS TAB ON VAULT CLIENT*

## Complicated File/Item Naming

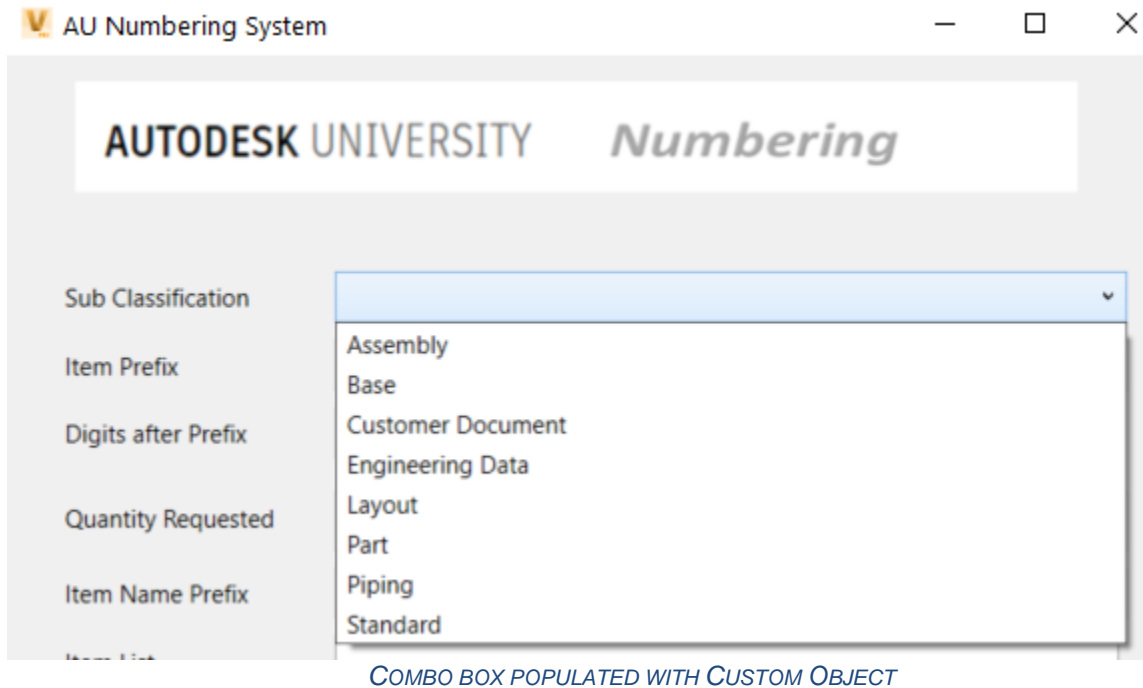
Use case, end user requires a system to Name Items with more than an autogenerated number. The first part of the name must be from a list of Classes, and each of these classes have defined item prefixes, number of digits following the prefix, and an option to override with a manual entry.

Out of the box Vault numbering schemes cannot handle that.

Referring to Part 1 of this class about Custom Objects. We defined a Custom Object called Class Prefixes. Each of these entries has some properties associated with it.



We will use this information to create the item name.



How did we do that? In the XAML file that creates the UI, when the Combo Box for Sub Classification was created it calls a Function: `cOSubClass` (custom object sub class)

```
<Label Content="Sub Classification" Grid.Column="0" Grid.Row="2" />
<ComboBox x:Name="cmbClassID" Grid.Column="1" Grid.Row="2" ItemsSource="{Binding PsList[cOSubClass]}"
    Grid.ColumnSpan="2" />
```

The ItemSource of the combobox calls the function `PsList[cOSubClass]`

In the folder where the PS1's reside, "AddinVault" there is a PS1 file that has this function in it.

1. Gather all the Custom Object definitions available in the vault, filter it for Subclass\_Categories
2. Gather all the Properties defined for Custom Objects
3. Perform a Search for all the Custom Objects in Subclass\_Categories
4. Pull the Names of each out into an array
5. Return the list of Names

```

10 function c0SubClass
11 {
12     $dsdiag.Trace(">> c0SubClass")
13     d3DialogReaction
14     $customObjects = $vault.CustomEntityService.GetAllCustomEntityDefinitions()
15     $classID = $customObjects | Where-Object { $_.displayName -eq "SubClass_Categories" }
16     $allPropDefs = $vault.PropertyService.GetPropertyDefinitionsByEntityClassId("CUSTENT")
17     $propDef = $allPropDefs | Where-Object { $_.SysName -eq "CustomEntitySystemName" }
18
19     $dsdiag.Trace(" custom objects found")
20     $srchConds = New-Object autodesk.Connectivity.WebServices.SrchCond[] 1
21     $srchCond = New-Object autodesk.Connectivity.WebServices.SrchCond
22     $srchCond.PropDefId = $propDef.Id
23     $srchCond.SrchOper = 3
24     $srchCond.SrchTxt = $classID.Name
25     $srchCond.PropType = [Autodesk.Connectivity.WebServices.PropertySearchType]::SingleProperty
26     $srchCond.SrchRule = [Autodesk.Connectivity.WebServices.SearchRuleType]::Must
27     $srchConds[0] = $srchCond
28     $dsdiag.Trace(" search conditions build")
29     $srchSort = New-Object autodesk.Connectivity.WebServices.SrchSort
30     $srchStatus = New-Object autodesk.Connectivity.WebServices.SrchStatus
31     $bookmark = ""
32     $global:ClassIDs=@()
33     do{
34         $global:ClassIDs += $vault.CustomEntityService.FindCustomEntitiesBySearchConditions(@($srchCond),@($srchSort),[ref]$bookmark,[ref]$srchStatus)
35     }while($global:ClassIDs.Count -lt $srchStatus.TotalHits)
36     $dsdiag.Trace(" search performed. "+$results.Count+" elements found")
37     $classIDNames = @()
38     $global:ClassIDs | ForEach-Object { $classIDNames += $_.Name }
39     $classIDNames = $classIDNames | sort
40     $dsdiag.Trace("<< c0SubClass")
41     return $classIDNames
42 }

```

#### FUNCTION COSUBCLASS

When the user selects the class the want, an event is triggered  
`$dswindow.FindName("cmbItemPrefix").add_selectionchanged(`

This routine calls yet another PowerShell function to populate the other dropdown boxes respectively – see the files in the class downloads for the full code.

Sub Classification	Base
Item Prefix	DD-
Digits after Prefix	6
Use Manual? <input type="checkbox"/>	

Summary of files to create this solution.

Files:

- D3\_General\_Functions.ps1
- Functions.ps1
- Reservations.ps1

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

### Functions.ps1

Contains functions related to the User Interface

- coSubClass – Pulls out the Custom Object information
- coltemDigits – Gets the Number of Digits from the selected Class
- cOgetltemPrefix – 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

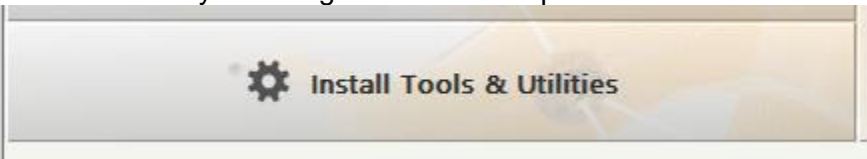
### Reservations.ps1

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

## Deploying Data Standards

This is a bit tricky installing the OOB is simple and is done with the



Section of the installation. That is the easy part.

Referring to the section above for folder structure you can see that all the custom configurations are stored on each PC. In the c:\ProgramData folder, which depending on your IT department you may not have access too.

As an administrator, I drink my own kool-aid and store a controlled set of files from the folder C:\ProgramData\Autodesk\Vault 2021\Extensions\DataStandard

I drop this entire folder into the Vault, typically under a \$/Admin/ folder. From here you can simply instruct your users to “GET” this folder and copy it over their existing DataStandards folder.

If the users don’t have access to that directory – local admin rights, then a utility may have to be written to grab those files with some added security and put them in the correct place.