

# Autodesk Vault 2020 Programming 101

Markus Koechl

Solutions Engineer PDM|PLM, Central Europe

Jeffrey Fishman

Software Engineer, DCP-Inventor





## About the speaker

### Markus Koechl

Markus is a Solution Engineer for Vault Products. He is driven by customer needs, practical workflows and always eager to overcome barriers by extensions or automation. That's the simple reason that he started programming Inventor, Inventor iLogic, and Vault APIs with the background of a Mechanical Engineer.



## About the speaker

### Jeffrey Fishman

Jeffrey Fishman is a Software Engineer for the Autodesk Vault product. He is driven by creative strategy and exploration -- seeking to understand and ultimately improve upon current methodologies and workflows in place today, for a more focused and streamlined tomorrow.

# Our Roles in this Class...

Markus Koechl



Markus – owns this session, being advocate of all attendees in the audience having “non – professional” software engineering education, ensuring that barriers go away.

Jeffrey Fishman



Jeffrey – owns the session [SD323470 - Feel the Power Between Vault and PowerShell](#). Here, he acts as advocate of the coding professionals ensuring that my simplifications are acceptable 😊

# Target Audience | Pre-Requisites

## YOU SHOULD BE...

- Eager to extend Vault automation, usage, tailored capabilities
- Familiar with scripting, e.g.
  - Inventor iLogic or AutoCAD vLisp

## YOU GET MOST OUT OF THIS CLASS IF ...

- You already used Visual Studio for .NET coding.

## ...WE DON'T EXPECT

- Experiences, know-how in Vault API
- Successful completion of Vault SDK samples

# Learning Objectives

## VAULT API – OVERVIEW AND INTRODUCTION

Understand different concepts accessing Vault software's programming interface for customization and automation

## VAULT 2020 SDK – TEMPLATES

Learn how to leverage the new SDK templates to get done your first custom application, job and client extension quickly.

## VAULT INVENTOR SERVER

Get insights on programming custom jobs using VaultInventorServer capabilities.

# Additional Materials

## HAND-OUT

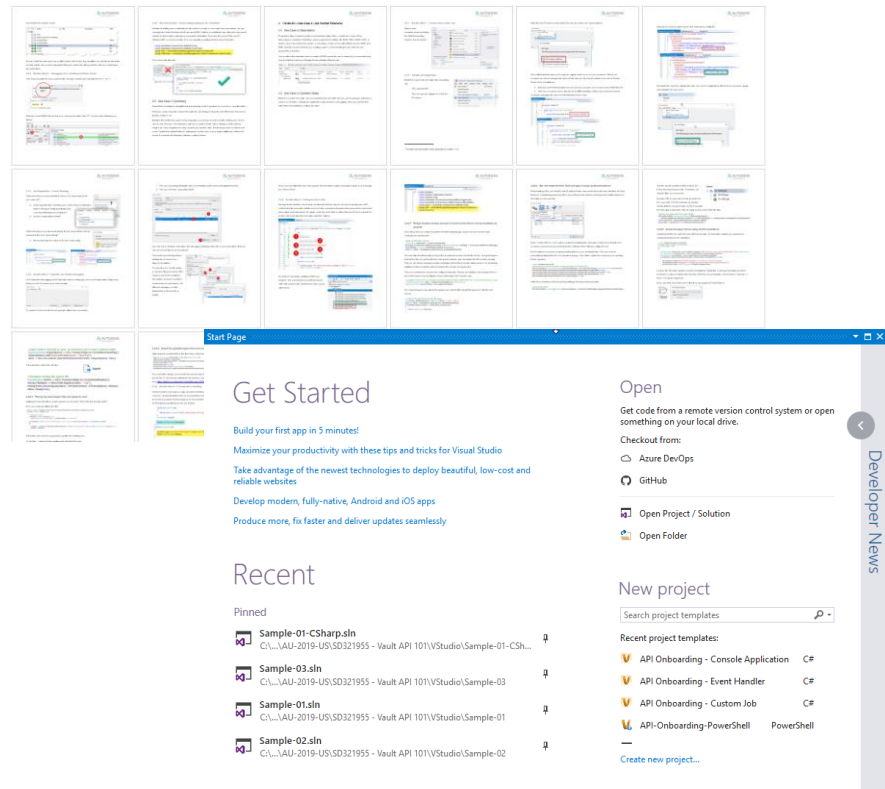
Step-by-step walk-throughs

Alternate solution paths

## VISUAL STUDIO 2017 PROJECTS

Samples 1 to 3 – Code Complete

Alternate solution paths



## Autodesk Vault API – Overview

## Use Case 1 – Automation Sample

## Use Case 2 Job Processor Extension

## Use Case 3 – Event Handler



# Autodesk Vault API – Overview



Server

ADMS

Autodesk Data Management Server

AVFS

Autodesk Vault Filestore Server

Client

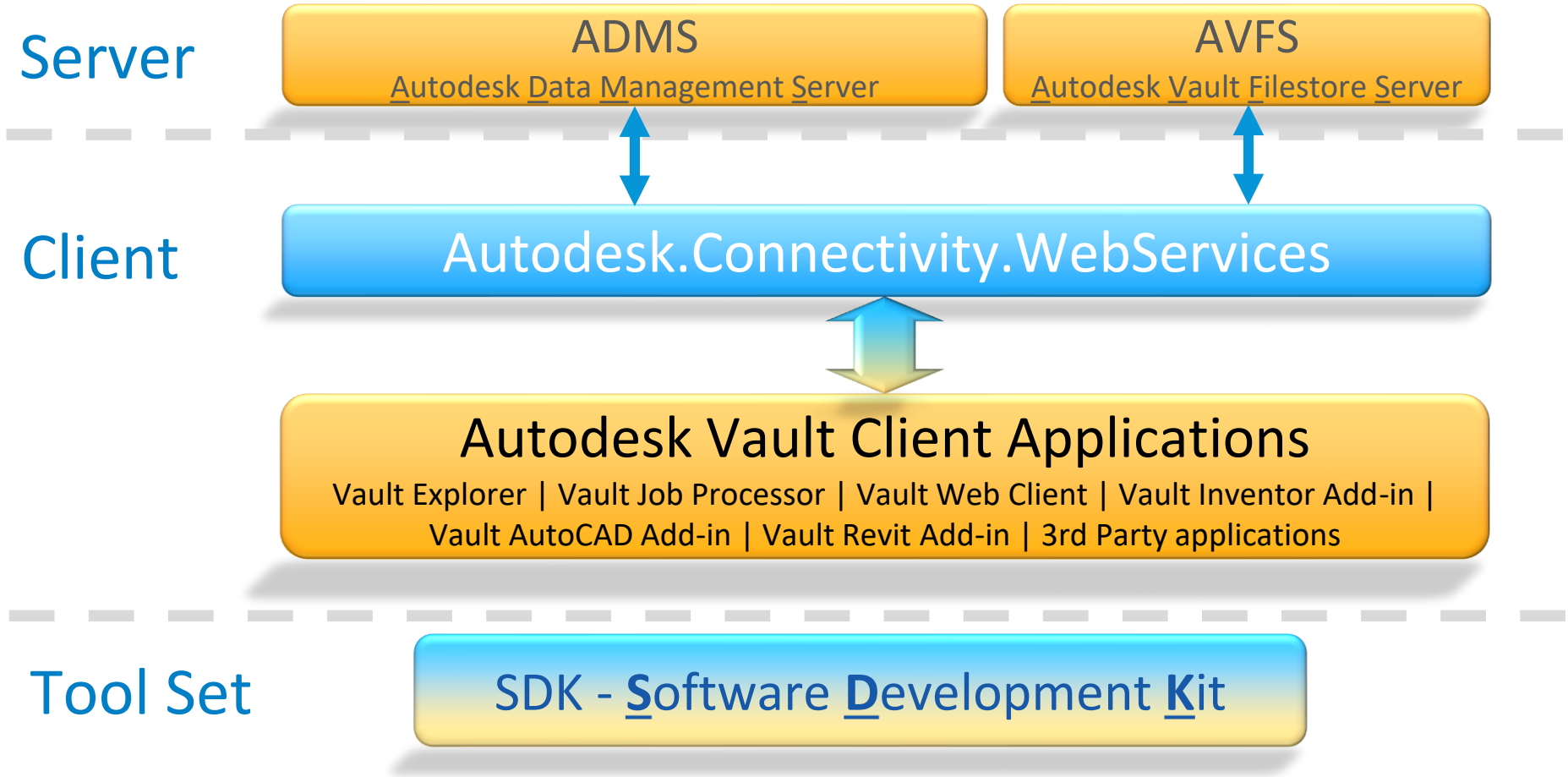
Autodesk.Connectivity.WebServices

Autodesk Vault Client Applications

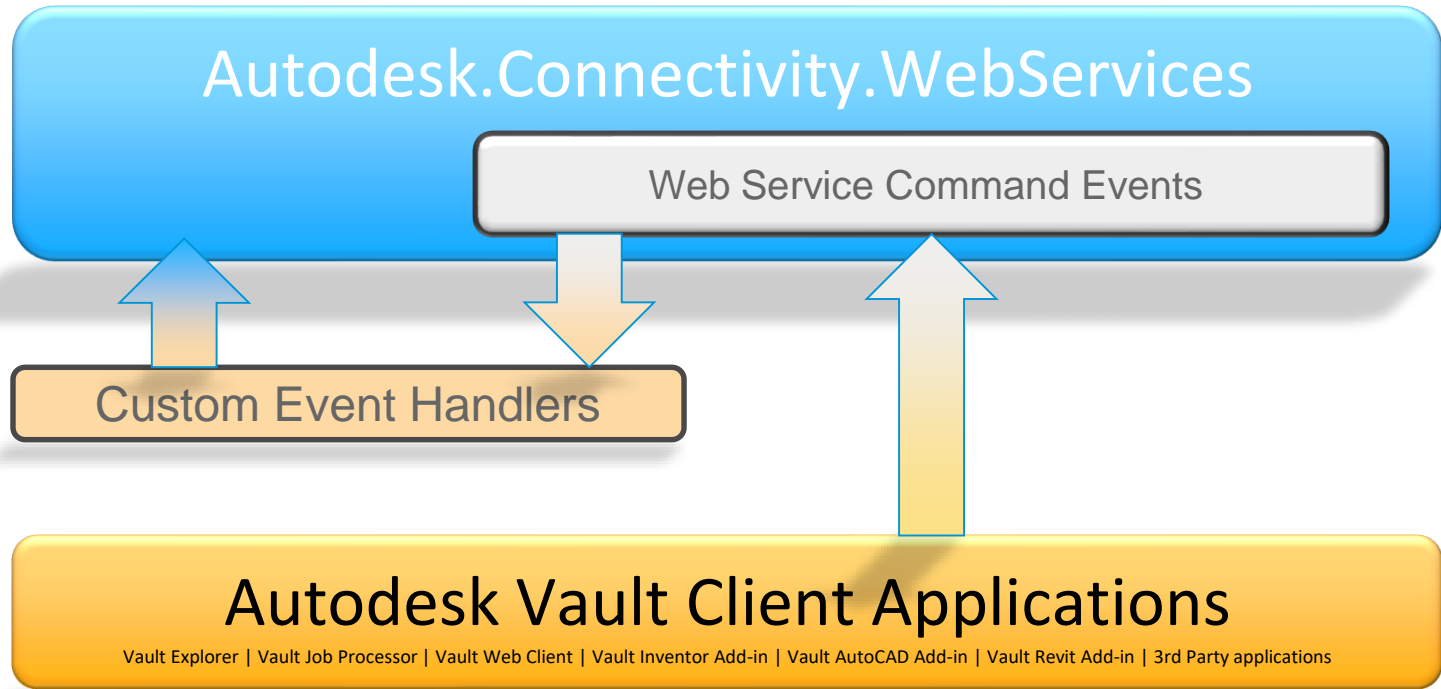
Vault Explorer | Vault Job Processor | Vault Web Client | Vault Inventor Add-in |  
Vault AutoCAD Add-in | Vault Revit Add-in | 3rd Party applications

Tool Set

SDK - Software Development Kit



# Event Handling



# Extensibility

Server

ADMS

Autodesk Data Management Server

AVFS

Autodesk Vault Filestore Server

Client

Autodesk.Connectivity.WebServices

Webservices API

Custom Standalone  
Client | Service

Job Processor

Job Proc. API

Vault Explorer

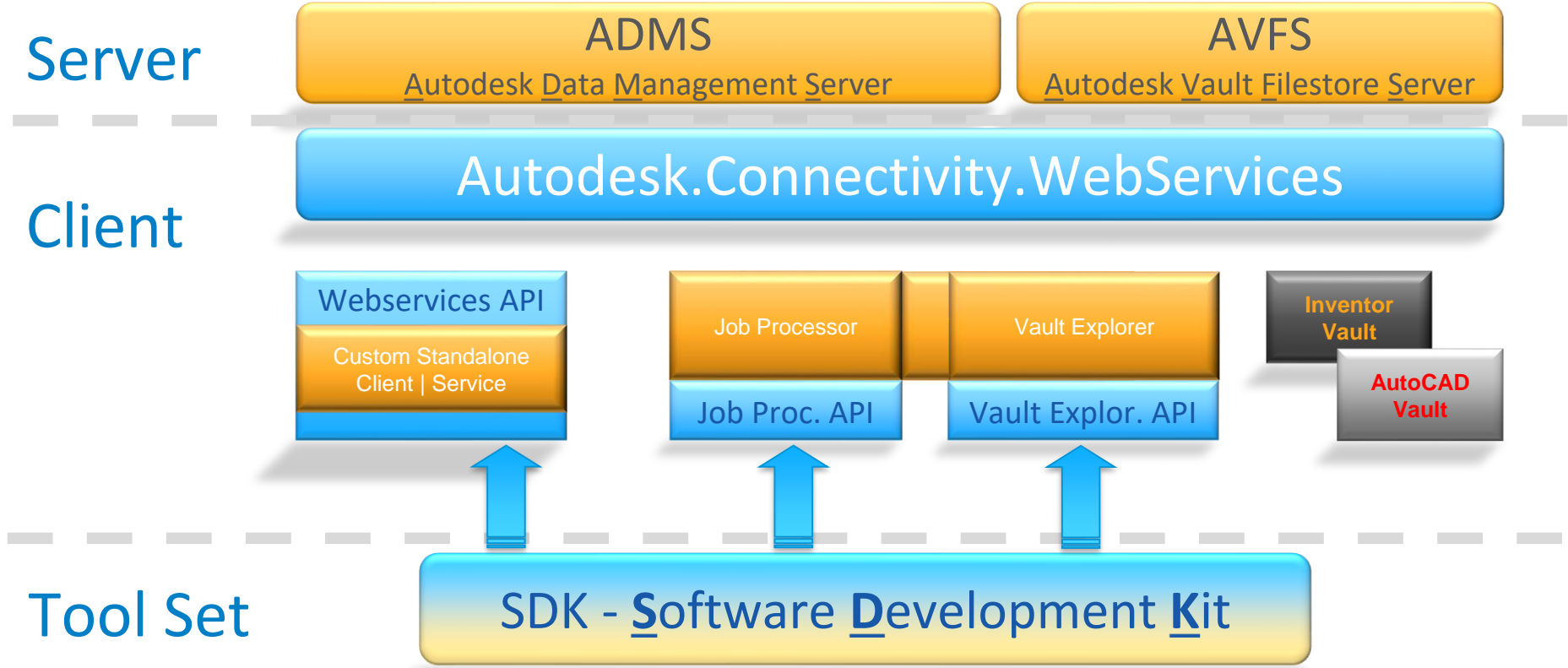
Vault Explor. API

Inventor  
Vault

AutoCAD  
Vault

Tool Set

SDK - Software Development Kit



# Vault Data Standard Extension

Server

ADMS

Autodesk Data Management Server

AVFS

Autodesk Vault Filestore Server

Client

Autodesk.Connectivity.WebServices

Vault Explorer

Inventor Vault

AutoCAD Vault

Vault Explorer

API

Vault Data Standard

Vault Data Standard Configuration

Tool Set

Custom Dialogs/Tabs (XAML)  
Business Logic/Rules (PowerShell)

SDK - Software Development Kit

What you need to start...



# Working Environment

## AUTODESK VAULT SDK

Install the SDK: C:\Program Files\Autodesk\Vault Professional 2020\SD

Copy SDK Templates following ReadMe.txt

Add PowerShell template from class material

## SDK DOCUMENTATION

Open the SDK Documentation:

C:\Program Files\Autodesk\Autodesk Vault 2020 SDK\docs\VaultSDK.chm

## SELECT AN EDITOR/IDE

Sample 01: Any PowerShell Editor – I prefer Visual Studio

PowerShell Tools or Visual Studio Code PowerShell Extension

Sample 02, Sample 03: Visual Studio 2017 (or later), any edition



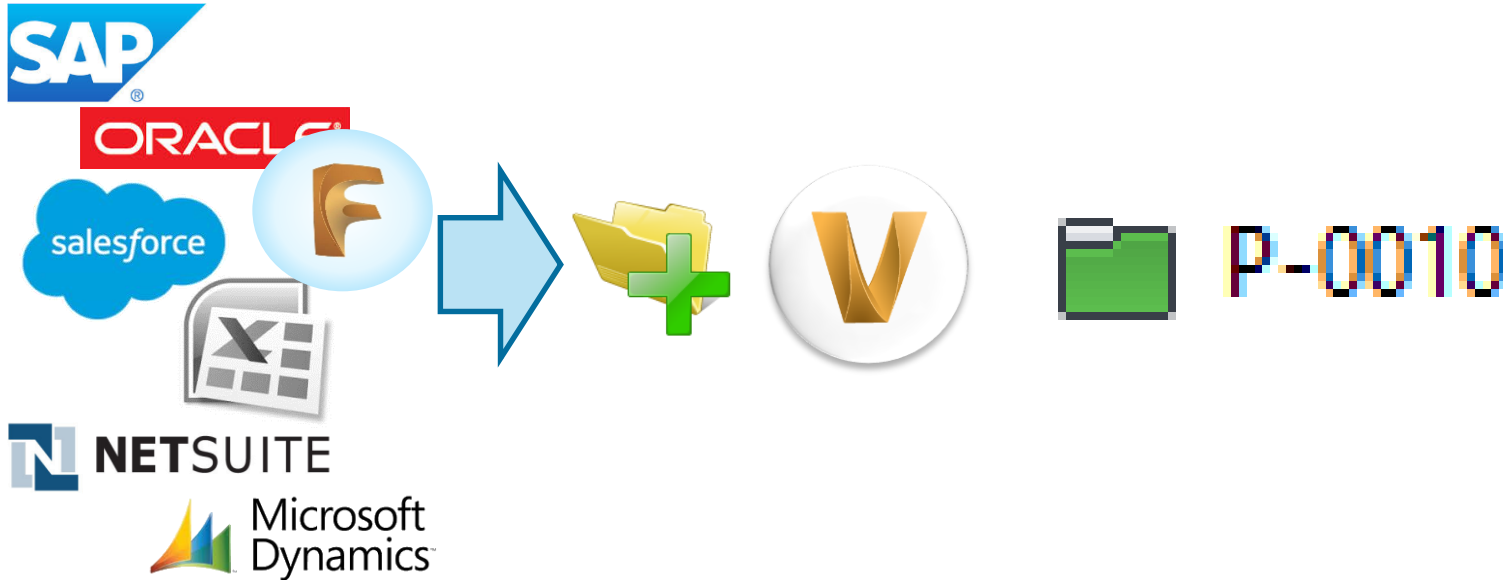
# Use Case 1 – Automation Sample





# Use Case 1 | Create Project Folder by CRM/ERP/..etc.

- New Project starts in CRM, ERP, ...
- CRM “pushes” new project folder creation in Vault via script...



# Use Case 1

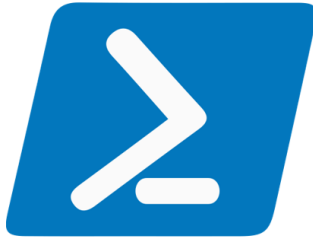
## Learning Objectives

- General steps accessing Vault
- Running Vault API calls from any system/application

# Use Case 1 | Solution Options

## Use Case 1 | PS Solution

1. PowerShell Script Editor
  - New PowerShell Project\*<sup>1</sup> or
  - Open "API-Onboarding-PowerShell.ps1" \*<sup>2</sup>
2. Load the .NET Vault API Assembly
3. Write code to...
  - Connect to Vault
  - Execute "Create Project Folder"
  - Disconnect
4. Debug...if needed



## Use Case 1 | C# Solution

- Visual Studio IDE
  - New API-Onboarding Console Application
- Done - Reference the .NET Vault API Assembly
- Write code to...
  - Connect to Vault
  - Execute "Create Project Folder"
  - Disconnect
- Debug...if needed



# Use Case 1 | PS Solution

## 1. PowerShell Script Editor

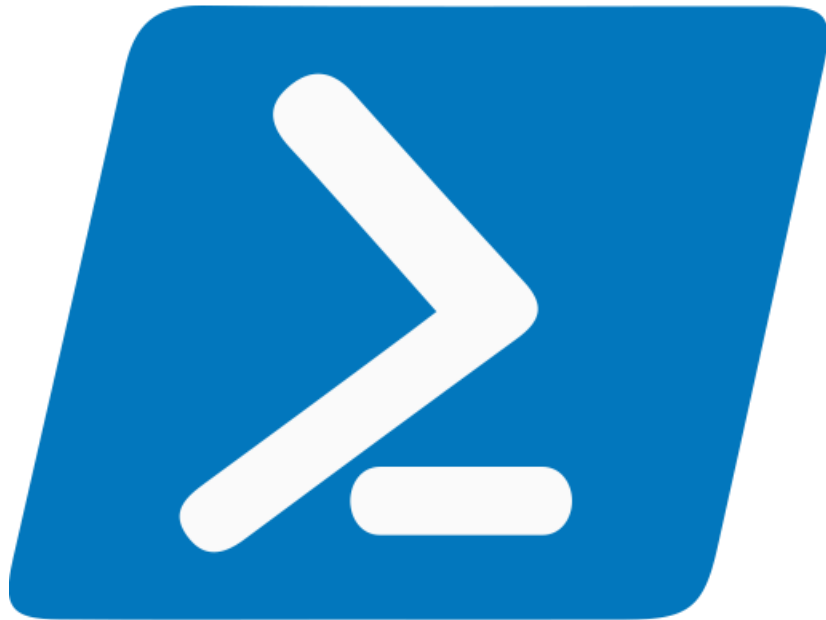
- New PowerShell Project<sup>\*1</sup> or
- Open “API-Onboarding-PowerShell.ps1”<sup>\*2</sup>

## 2. Load the .NET Vault API Assembly

## 3. Write code to...

- Connect to Vault
- Execute “Create Project Folder”
- Disconnect

## 4. Debug...if needed



# Use Case 1 | Solution Steps...

## 1. PowerShell Script Editor

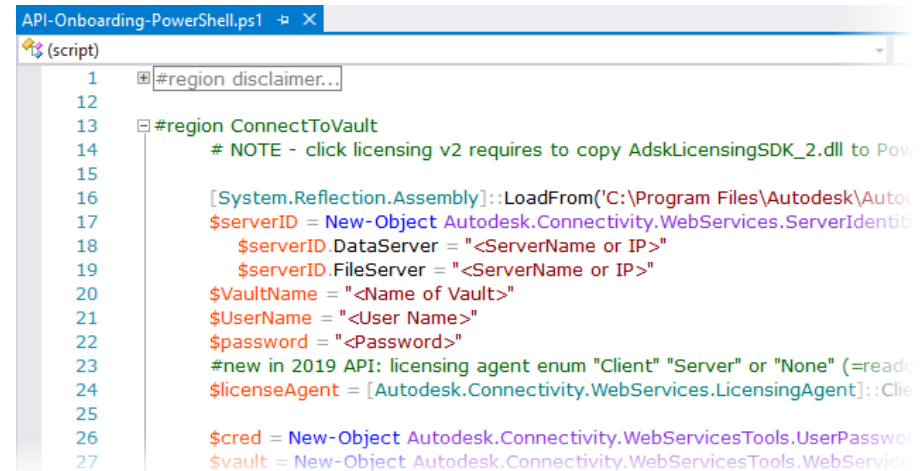
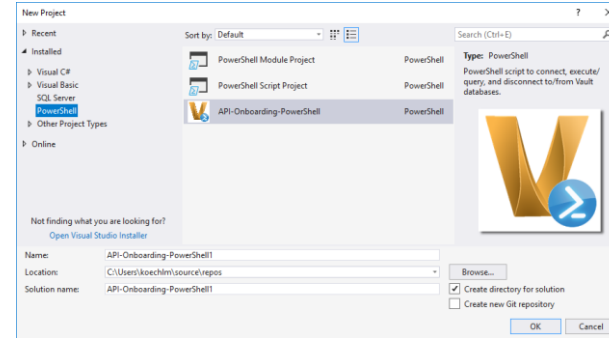
- New PowerShell Project\*<sup>1</sup> or
- Open “API-Onboarding-PowerShell.ps1” \*<sup>2</sup>

## 2. Load the .NET Vault API Assembly

## 3. Write code to...

- Connect to Vault
- Execute “Create Project Folder”
- Disconnect

## 4. Debug...if needed



# Use Case 1 | Solution Steps...

## 1. PowerShell Script Editor

- New PowerShell Project<sup>\*1</sup> or
- Open “API-Onboarding-PowerShell.ps1”<sup>\*2</sup>

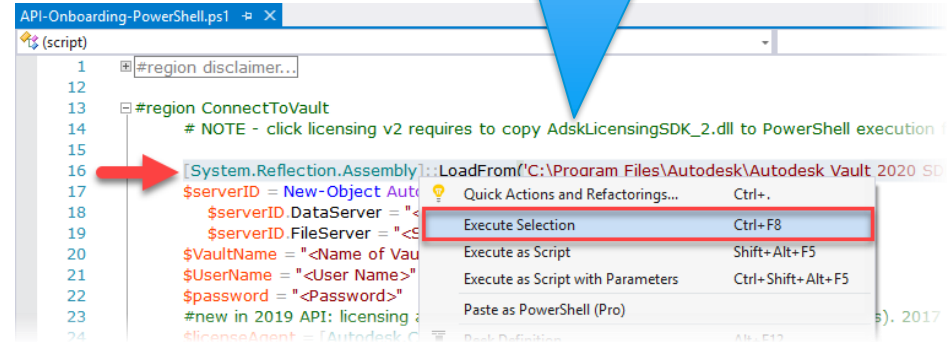
## 2. Load the .NET Vault API Assembly

## 3. Write code to...

- Connect to Vault
- Execute “Create Project Folder”
- Disconnect

## 4. Debug...if needed

Don't miss to copy the  
Licensing library to  
PowerShell!



# Use Case 1 | Solution Steps...

1. PowerShell Script Editor
2. Load the .NET Vault API Assembly
3. Write code to...
  - Connect to Vault using your
    1. Connection Parameters
    2. License Type
  - Execute “Create Project Folder”
  - Disconnect
4. Debug...if needed

The image displays two screenshots of a PowerShell script editor window titled "API-Onboarding-PowerShell.ps1".

The first screenshot shows the following code:

```
17 $serverID = New-Object Autodesk.Connectivity.WebServices.ServerIdentities
18 $serverID.DataServer = "192.168.85.128"
19 $serverID.FileServer = "192.168.85.128"
20 $VaultName = "PDM-Sample"
21 $UserName = "CAD Admin"
22 $password = ""
23 #new in 2019 API: licensing agent enum "Client" "Server" or "None" (=readonly access). 2017 and 2018
24 $licenseAgent = [Autodesk.Connectivity.WebServices.LicensingAgent]::Client
25
26 $cred = New-Object Autodesk.Connectivity.WebServicesTools.UserPass
27 $vault = New-Object Autodesk.Connectivity.WebServicesTools.WebService
28
29 #region ExecuteInVault
30
```

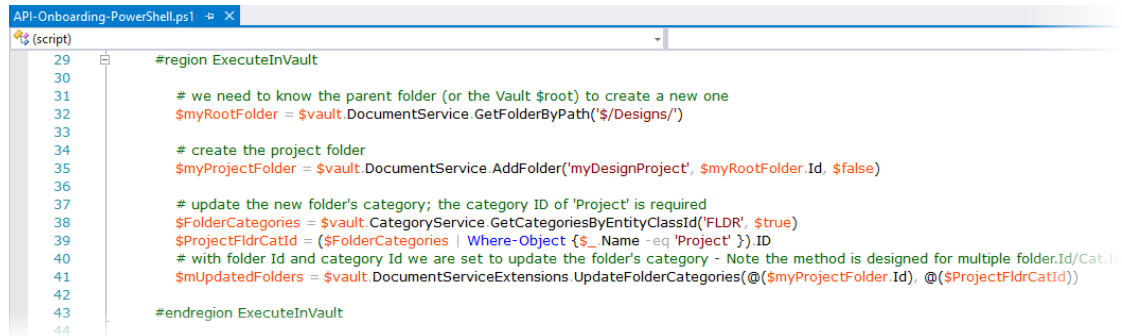
A red circle with the number "1" is placed over the line defining `$licenseAgent`. A dropdown menu is open, showing options: Client (selected), None, Server, Equals, and Format. A red circle with the number "2" is placed over the "Client" option.

The second screenshot shows the following code:

```
43 #endregion ExecuteInVault
44
45 $vault.Dispose() #don't forget to release the connection, to
46
47
48 #endregion ConnectToVault
```

# Use Case 1 | Solution Steps...

1. PowerShell Script Editor
2. Load the .NET Vault API Assembly
3. Write code to...
  - Connect to Vault
  - Execute “Create Project Folder”
  - Disconnect
4. Debug...if needed



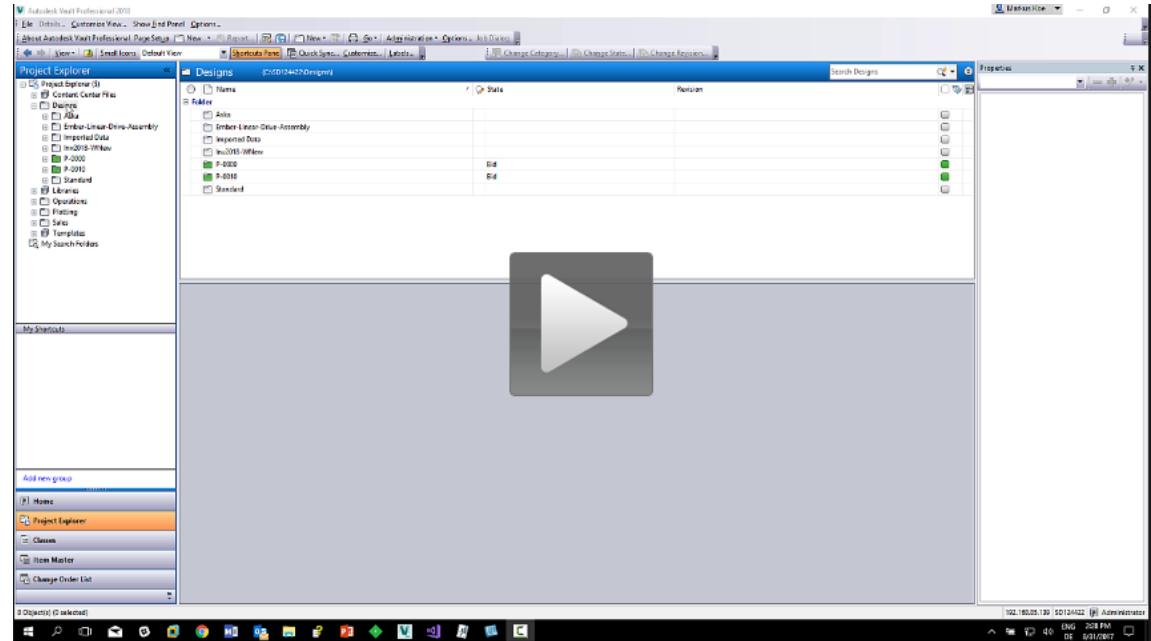
```
API-Onboarding-PowerShell.ps1
(script)

29 #region ExecuteInVault
30
31 # we need to know the parent folder (or the Vault $root) to create a new one
32 $myRootFolder = $vault.DocumentService.GetFolderByPath('$/Designs/')
33
34 # create the project folder
35 $myProjectFolder = $vault.DocumentService.AddFolder('myDesignProject', $myRootFolder.Id, $false)
36
37 # update the new folder's category; the category ID of 'Project' is required
38 $FolderCategories = $vault.CategoryService.GetCategoriesByEntityClassId('FLDR', $true)
39 $ProjectFldrCatId = ($FolderCategories | Where-Object {$_.Name -eq 'Project'}).ID
40 # with folder Id and category Id we are set to update the folder's category - Note the method is designed for multiple folder.Id/CatId
41 $mUpdatedFolders = $vault.DocumentServiceExtensions.UpdateFolderCategories(@($myProjectFolder.Id), @($ProjectFldrCatId))
42
43 #endregion ExecuteInVault
44
```



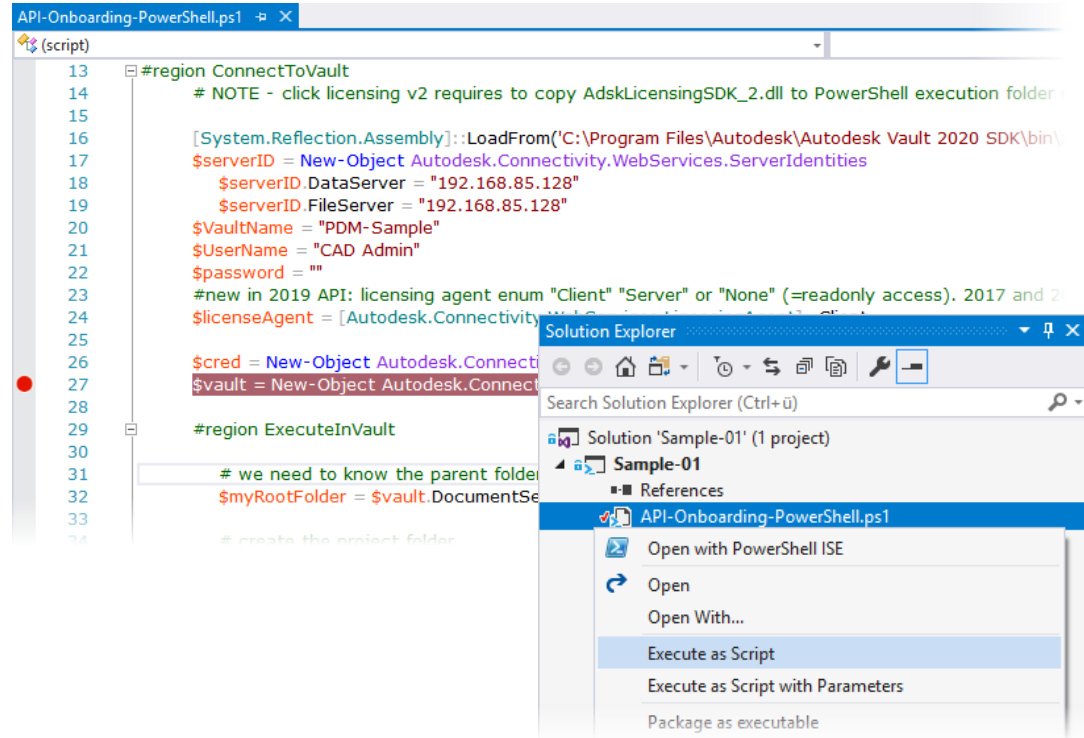
## Excuse | Figuring out how to code...

- Create Folder
- Change Category to “Project”

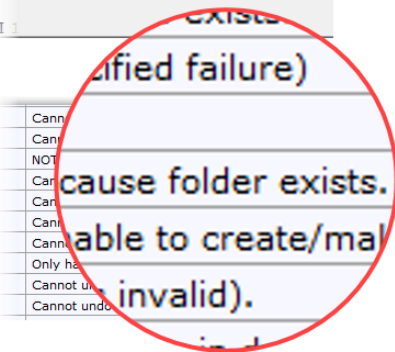


# Use Case 1 | Solution Steps...

1. PowerShell Script Editor
2. Load the .NET Vault API Assembly
3. Write code to...
4. Step through and debug

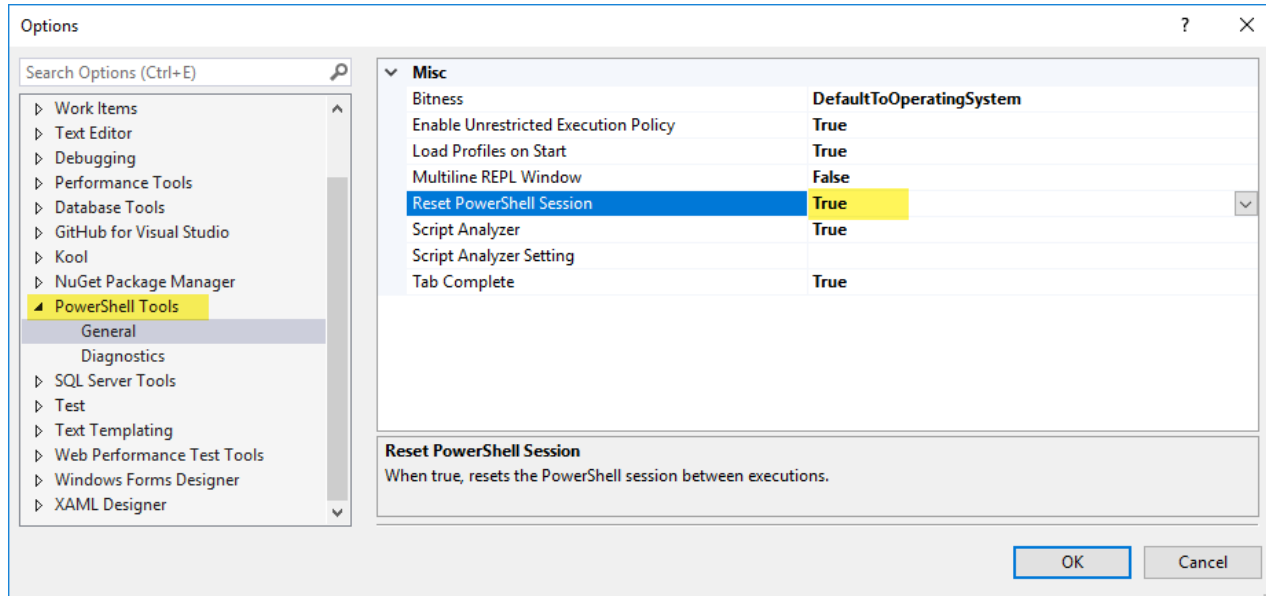


1. PowerShell Script Editor
2. Load the .NET Vault API Assembly
3. Write code to...
4. Step through and debug



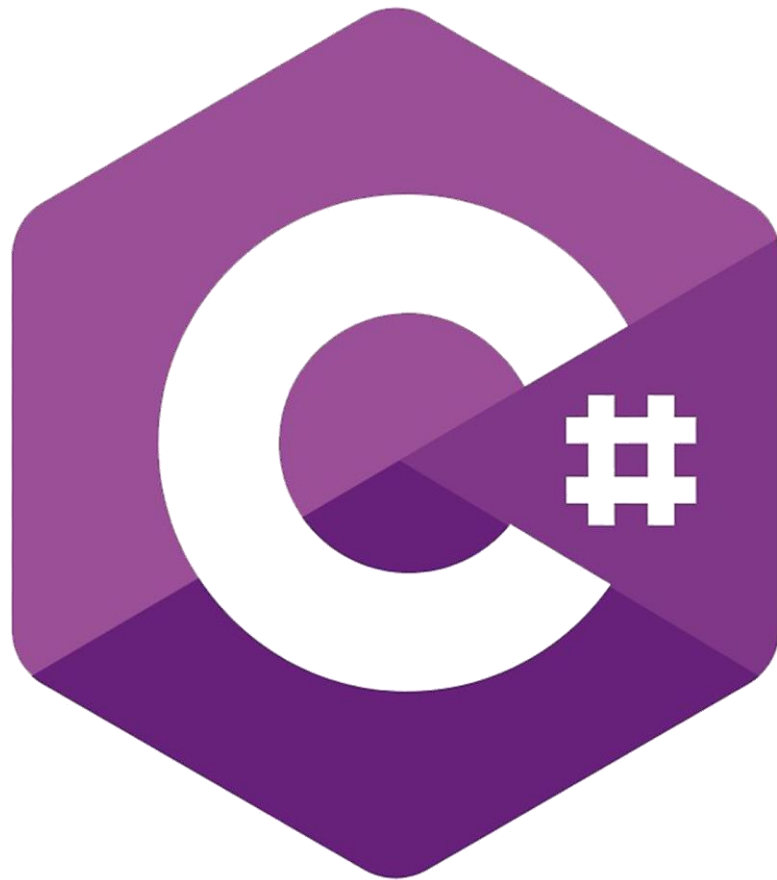
# Step Through and Debug | Tip

- Reset PowerShell session between executions
  - Useful to get “fresh” global variables and constants for the next script run



# Use Case 1 | C# Solution

- Visual Studio IDE
  - New API-Onboarding Console Application
- Done - Reference the .NET Vault API Assembly
- Write code to...
  - Connect to Vault
  - Execute “Create Project Folder”
  - Disconnect
- Debug...if needed



# Use Case 1 | Solution Steps...

## 1. Visual Studio -> New C# Project

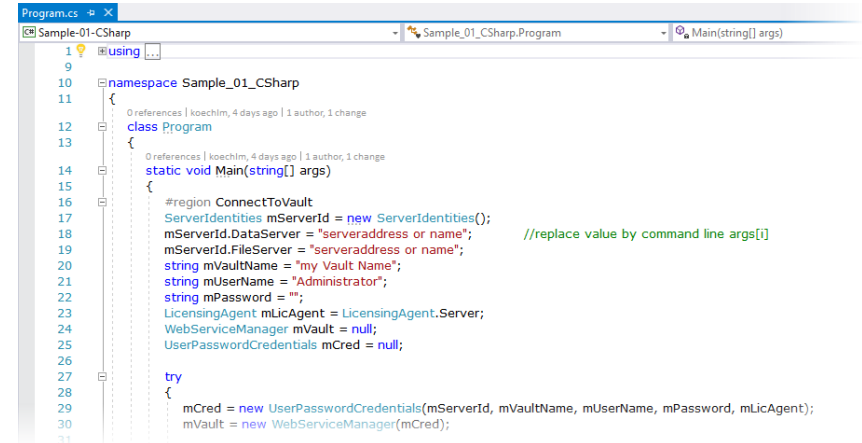
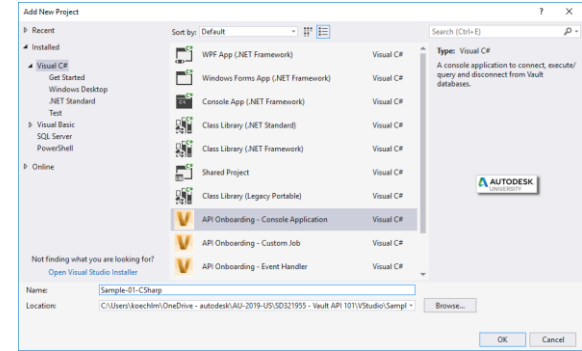
- Select Template API-Onboarding Console Application
- Open code Program.cs

## 2. Review project template

## 3. Write code to...

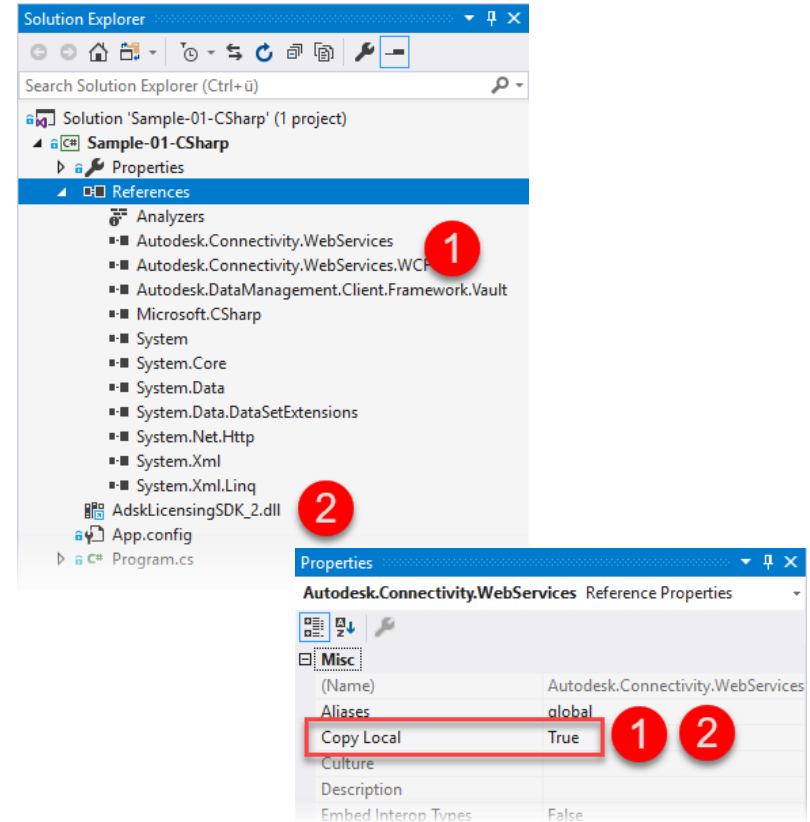
- Connect to Vault
- Execute “Create Project Folder”
- Disconnect

## 4. Debug...if needed



# Use Case 1 | Solution Steps...

1. Visual Studio -> New C# Project
2. Review project template
  - References
  - Licensing “AdskLicensingSDK\_2.dll
3. Write code to...
  - Connect to Vault
  - Execute “Create Project Folder”
  - Disconnect
4. Debug...if needed



# Use Case 1 | Solution Steps...

1. Visual Studio -> New C# Project

2. Review project template

3. Write code to...

- Connect to Vault using your

1. Connection Parameters

2. License Type

- Execute “Create Project Folder”

- Disconnect

4. Debug...if needed

```
1 using ...
9
10 namespace Sample_01_CSharp
11 {
12     0 references | koechlm, 4 days ago | 1 author, 1 change
13     class Program
14     {
15         0 references | koechlm, 4 days ago | 1 author, 1 change
16         static void Main(string[] args)
17         {
18             #region ConnectToVault
19             ServerIdentities mServerId = new ServerIdentities();
20             mServerId.DataServer = "192.168.85.128"; //replace value by command
21             mServerId.FileServer = "192.168.85.128";
22             string mVaultName = "PDMC-Sample";
23             string mUserName = "CAD Admin";
24             string mPassword = "";
25             LicensingAgent mLicAgent = LicensingAgent.;
26             WebServiceManager mVault = null;
27             UserPasswordCredentials mCred = null;
28             try
29             {
30                 throw ex;
31             }
32             //never forget to release the license, especially if pulled from Server
33             mVault.Dispose();
34         }
35     }
36     catch (Exception ex)
```



# Use Case 1 | Solution Steps...

1. Visual Studio -> New C# Project

2. Review project template

3. Write code to...

- Connect to Vault

- Execute “Create Project Folder”

- Disconnect

4. Debug...if needed

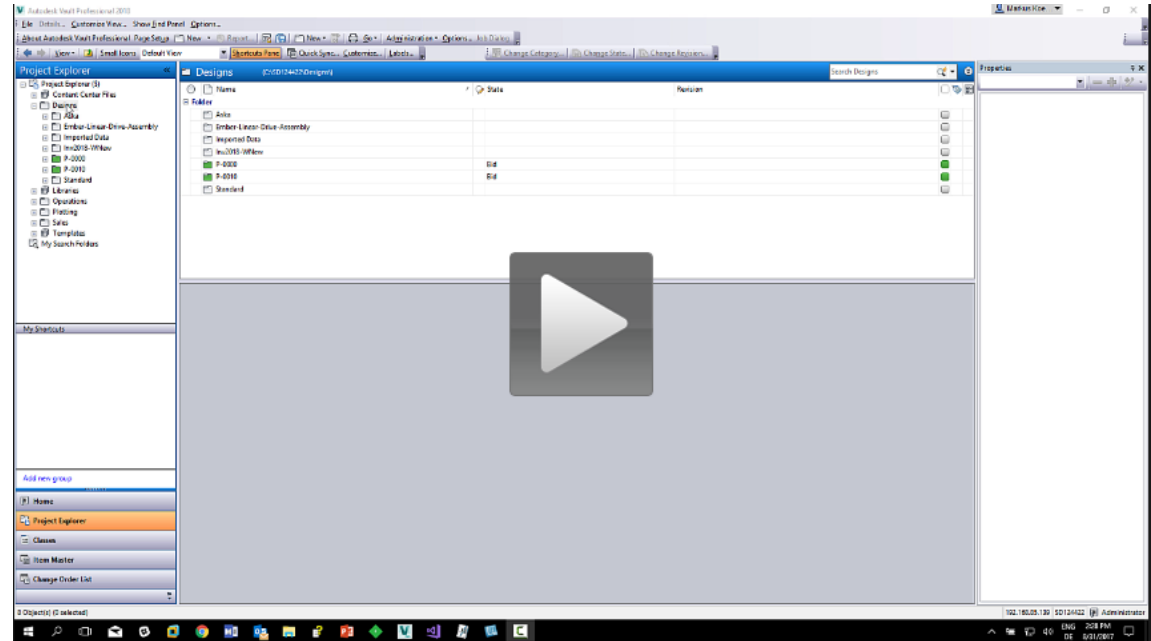
```
//we need to know the parent folder (or the Vault $root) to create a new one
Folder mRootFolder = mVault.DocumentService.GetFolderByPath("$/Designs");
string mFldrName = "myDesignProject";
```

```
//to create a folder with category; the category ID of 'Project' is required
Cat[] mFolderCategories = mVault.CategoryService.GetCategoriesByEntityClassId("FLDR", true);
long mProjectFldrCatId = mFolderCategories.Where(n => n.Name == "Project").FirstOrDefault().Id;
```

```
//with parent folder Id and category Id we are set create folder with category in one call
Folder mProjectFolder = mVault.DocumentServiceExtensions.AddFolderWithCategory(mFldrName, mRootFolder.Id, mProjectFldrCatId);
```

## Excuse | Figuring out how to code...

- Create Folder
- Change Category to “Project”



# Use Case 1 | Solution Steps...

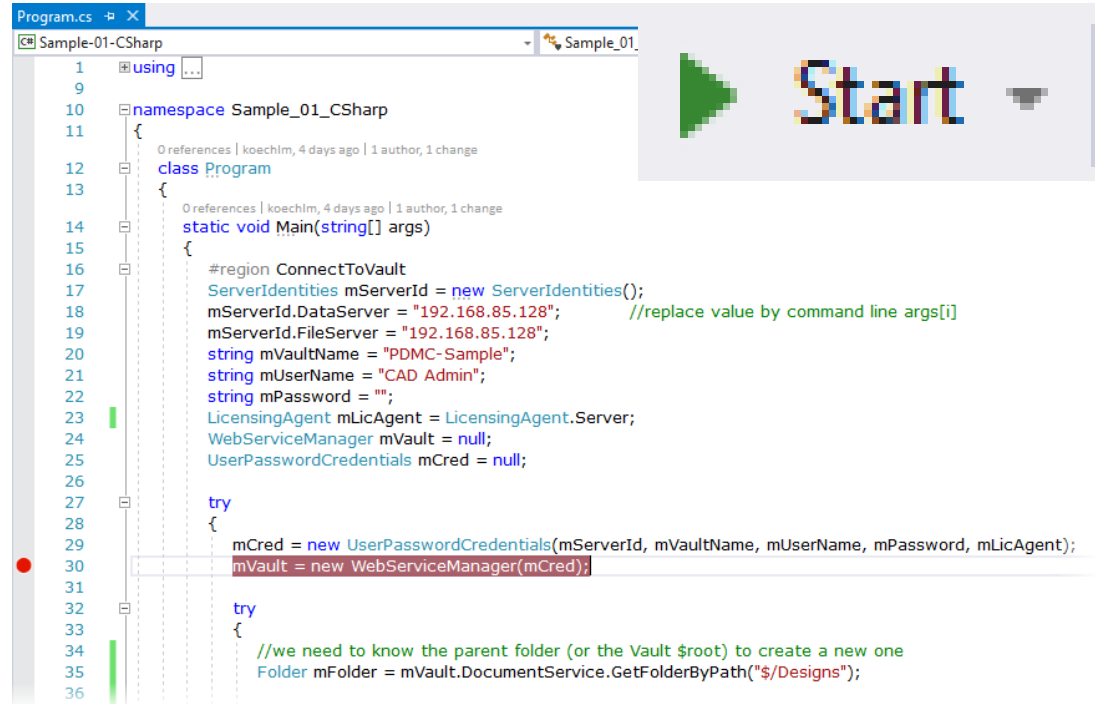
1. Visual Studio -> New C# Project

2. Review project template

3. Write code to...

- Connect to Vault
- Execute "Create Project Folder"
- Disconnect

4. Step through and debug



```
Program.cs - X
Sample-01-CSHarp
1  using ...
9
10 namespace Sample_01_CSharp
11 {
12     0 references | koechlm, 4 days ago | 1 author, 1 change
13     class Program
14     {
15         0 references | koechlm, 4 days ago | 1 author, 1 change
16         static void Main(string[] args)
17         {
18             #region ConnectToVault
19             ServerIdentities mServerId = new ServerIdentities();
20             mServerId.DataServer = "192.168.85.128"; //replace value by command line args[i]
21             mServerId.FileServer = "192.168.85.128";
22             string mVaultName = "PDMC-Sample";
23             string mUserName = "CAD Admin";
24             string mPassword = "";
25             LicensingAgent mLicAgent = LicensingAgent.Server;
26             WebServiceManager mVault = null;
27             UserPasswordCredentials mCred = null;
28
29             try
30             {
31                 mCred = new UserPasswordCredentials(mServerId, mVaultName, mUserName, mPassword, mLicAgent);
32                 mVault = new WebServiceManager(mCred);
33
34                 try
35                 {
36                     //we need to know the parent folder (or the Vault $root) to create a new one
37                     Folder mFolder = mVault.DocumentService.GetFolderByPath("$/Designs");
```

# Use Case 1 | Solution Steps...

1. Visual Studio -> New C# Project

2. Review project template

3. Write code to...

- Connect to Vault
- Execute "Create Project Folder"
- Disconnect

4. Step through and debug

The screenshot shows a Visual Studio IDE with a C# code file. The code is part of a region labeled 'connect to Vault'. It contains a try-catch block. The catch block is for 'Autodesk.Vault.WebServices.VaultServiceErrorException' with error code '1011'. The code throws an exception when the error code is '1011'. A red circle highlights the 'Autodesk' namespace and the '1011' error code. Below the code, the 'Vault SDK' documentation is open, showing a table of 'Server Error Codes'. A red circle highlights the '1011' error code in the table, which corresponds to 'AddFolderExists'.

Title	Location	Rank
Server Error Codes	Vault SDK	1
Restriction Codes	Vault SDK	2

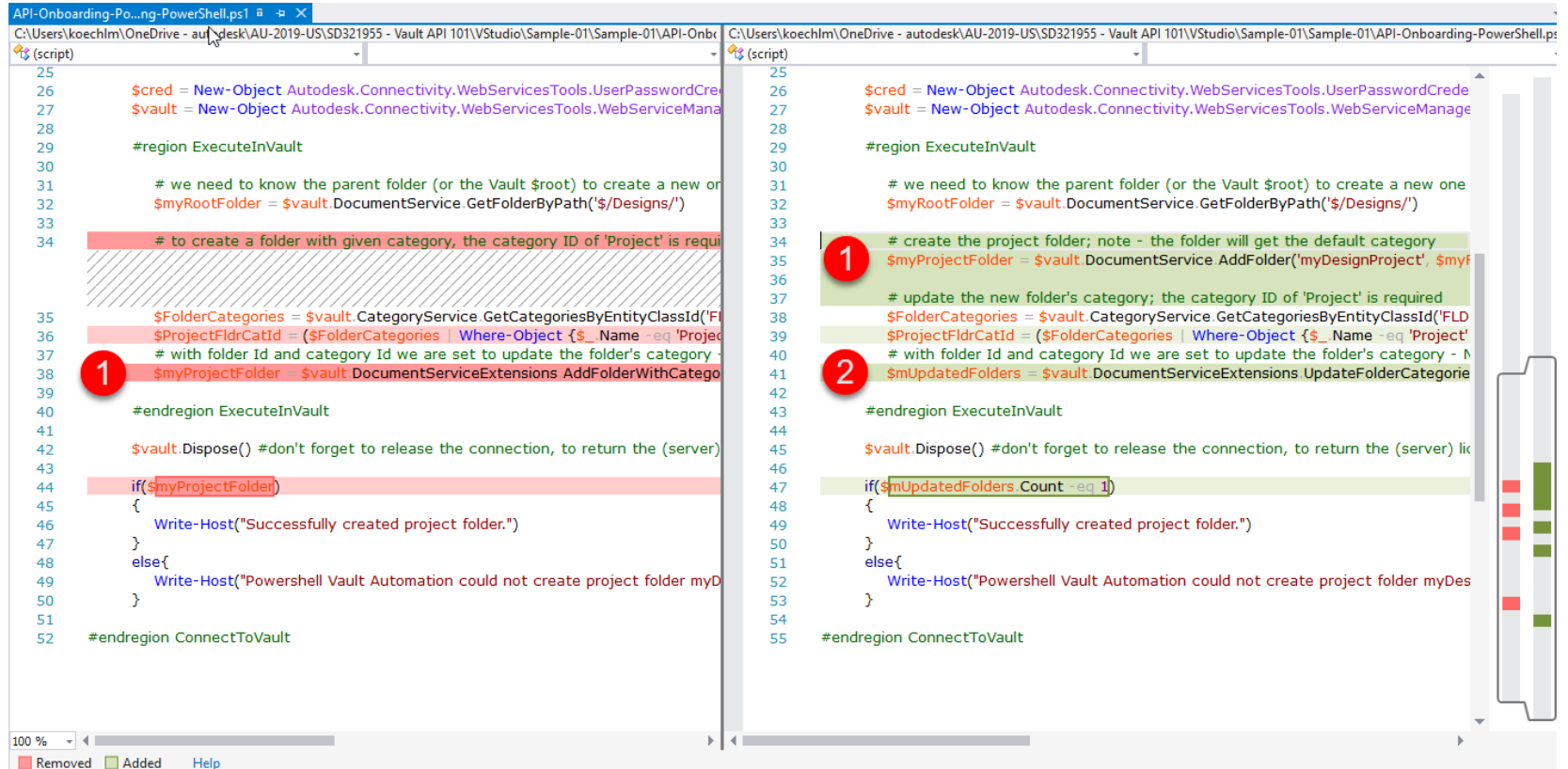
  

Code	Description
1008	AddFileExists
1009	AddFileFailed
1010	NOT USED ANYMORE.
1011	AddFolderExists
1012	AddFailedCreateFolder
1013	GetFileFailed
1014	MakeVersionFailed
1015	DeleteFileWithDependencies
1016	UndoCheckoutWrongUser
1017	UndoCheckoutWrongFolder

## Use Case 1 | Takeaways 1/2

- Connect to Vault using WebServiceManager is straightforward
- Try to minimize the number of server calls during execution
  - E.g., we created the folder and updated the category in a separate call
  - ⇒ Always search for alternate functions
  - ⇒ Avoid loops calling the API

# Sample-01 | Compare Solutions



```
API-Onboarding-Po...ng-PowerShell.ps1
C:\Users\koechlm\OneDrive - Autodesk\AU-2019-US\SD321955 - Vault API 101\VStudio\Sample-01\Sample-01\API-Onboarding-PowerShell.ps1
(script)
25
26 $cred = New-Object Autodesk.Connectivity.WebServicesTools.UserPasswordCred
27 $vault = New-Object Autodesk.Connectivity.WebServicesTools.WebServiceMana
28
29 #region ExecuteInVault
30
31 # we need to know the parent folder (or the Vault $root) to create a new or
32 $myRootFolder = $vault.DocumentService.GetFolderByPath('$/Designs/')
33
34 # to create a folder with given category, the category ID of 'Project' is requi
35
36 $FolderCategories = $vault.CategoryService.GetCategoriesByEntityClassId('FLD
37 $ProjectFldrCatId = ($FolderCategories | Where-Object {$_.Name -eq 'Project'
38 $myProjectFolder = $vault.DocumentServiceExtensions.AddFolderWithCatego
39
40 #endregion ExecuteInVault
41
42 $vault.Dispose() #don't forget to release the connection, to return the (server)
43
44 if($myProjectFolder)
45 {
46     Write-Host("Successfully created project folder.")
47 }
48 else{
49     Write-Host("Powershell Vault Automation could not create project folder myD
50 }
51
52 #endregion ConnectToVault

C:\Users\koechlm\OneDrive - Autodesk\AU-2019-US\SD321955 - Vault API 101\VStudio\Sample-01\Sample-01\API-Onboarding-PowerShell.ps1
(script)
25
26 $cred = New-Object Autodesk.Connectivity.WebServicesTools.UserPasswordCred
27 $vault = New-Object Autodesk.Connectivity.WebServicesTools.WebServiceMana
28
29 #region ExecuteInVault
30
31 # we need to know the parent folder (or the Vault $root) to create a new one
32 $myRootFolder = $vault.DocumentService.GetFolderByPath('$/Designs/')
33
34 # create the project folder; note - the folder will get the default category
35 $myProjectFolder = $vault.DocumentService.AddFolder('myDesignProject', $myF
36
37 # update the new folder's category; the category ID of 'Project' is required
38 $FolderCategories = $vault.CategoryService.GetCategoriesByEntityClassId('FLD
39 $ProjectFldrCatId = ($FolderCategories | Where-Object {$_.Name -eq 'Project'
40 # with folder Id and category Id we are set to update the folder's category - N
41 $mUpdatedFolders = $vault.DocumentServiceExtensions.UpdateFolderCategorie
42
43 #endregion ExecuteInVault
44
45 $vault.Dispose() #don't forget to release the connection, to return the (server) li
46
47 if($mUpdatedFolders.Count -eq 1)
48 {
49     Write-Host("Successfully created project folder.")
50 }
51 else{
52     Write-Host("Powershell Vault Automation could not create project folder myDes
53 }
54
55 #endregion ConnectToVault
```

100 %

Removed Added Help

# Use Case 1 | Takeaways 1/2

- Do you prefer using a GUI to enter user/password at first time using?

⇒ Great 😊! Check-out the alternate solution using a one time login-in dialog

```
#region ConnectToVault
ServerIdentities mServerId = new ServerId
mServerId.DataServer = "192.168.1.128";
mServerId.FileServer = "192.168.1.128";
string mVaultName = "PDMC-";
string mUserName = "CAD";
string mPassword = "";
LicensingAgent mLicAgent = LicensingAgen
WebServiceManager mVault = null;
UserPasswordCredentials mCred = null;
```

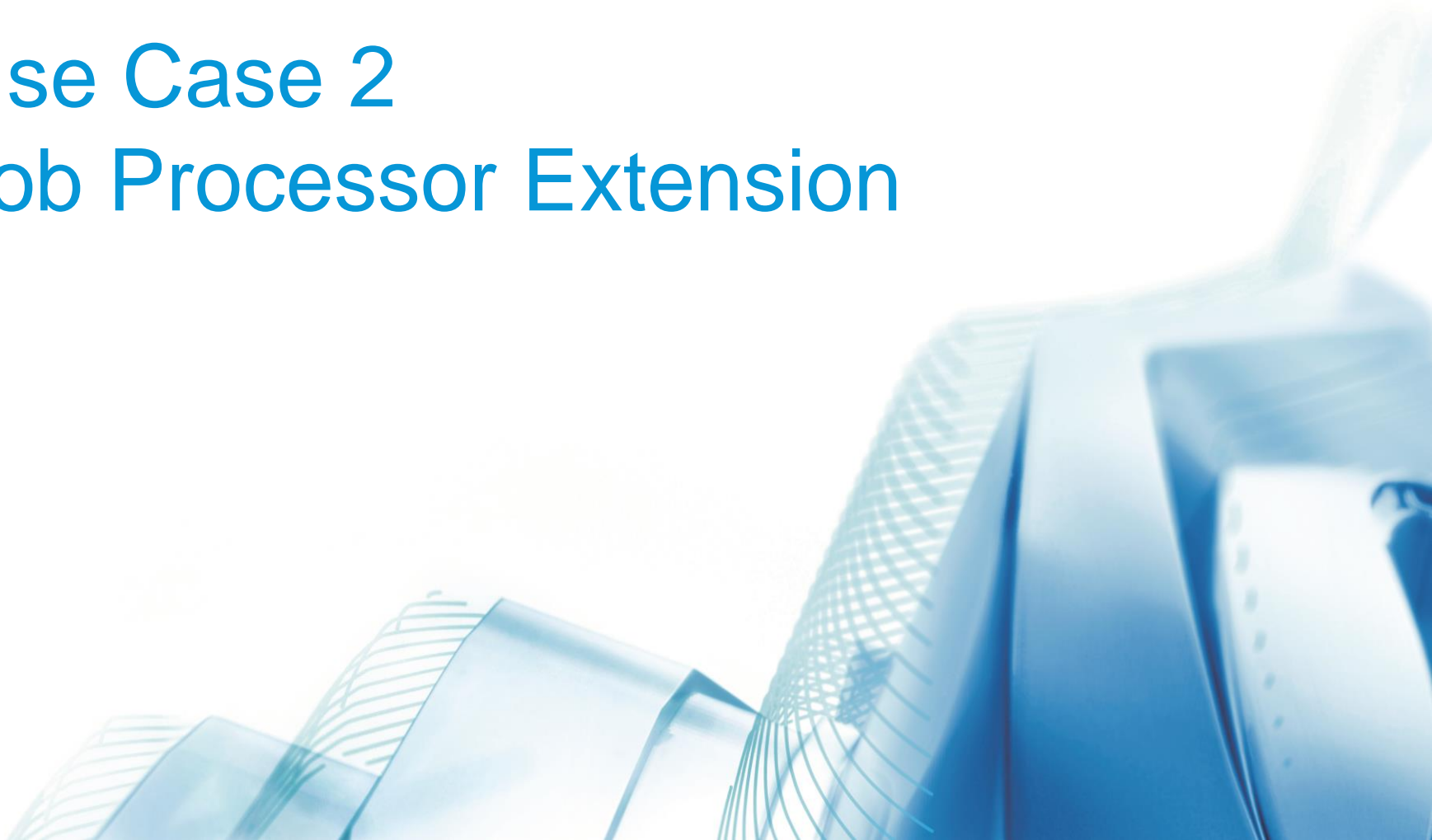
```
try
{
    mCred = new UserPasswordCredentials(
    mVault = new WebServiceManager(mCred
```

```
#region ConnectToVault
VdfForms.Settings.LoginSettings mloginSettings = new VdfForms.Settings.LoginSettings();
//first time log-in will ask for user credentials, set to Autologin = Enabled to avoid future credential requests
mloginSettings.AutoLoginMode = VdfForms.Settings.LoginSettings.AutoLoginModeValues.RestoreAndExecute;
VDF.Vault.Currency.Connections.Connection mConn = null;
WebServiceManager mVault = null;
try
{
    mConn = VdfForms.Library.Login(mloginSettings);
    if (mConn == null)
    {
        Debug.Print("Login failed or canceled");
        return;
    }
    else
    {
        mVault = mConn.WebServiceManager;
    }
}
```



# Use Case 2

## Job Processor Extension





## Use Case 2 | Hand over files to CRM/ERP/...

- Frequently other business systems expect information, files or links from Vault
- If the information is related to CAD files, neutral format files like DWF, PDF, STEP, DXF, etc. are expected
- ✓ Vault comes with default jobs for DWF and PDF
- This sample will share a job that easily adopts all Inventor export formats.

History Uses Where Used Change Order Preview CAD BOM Derivation Tree Datasheet File Structure Assigned Item									
Latest Released									
File Name			Title	Part Number	Description	Revision	State (Historical)	Material	Created By
001434.ipt			SM Sample for Rivet Conne...	001434	SM Part for BOM Setu...	B	Released	1.0330 DC01	JobProcessor
Attachments									
001434.stp			SM Sample for Rivet Conne...	001434	SM Part for BOM Setu...	B	Released	1.0330 DC01	JobProcessor

Design Representation

# Use Case 2

## Learning Objectives

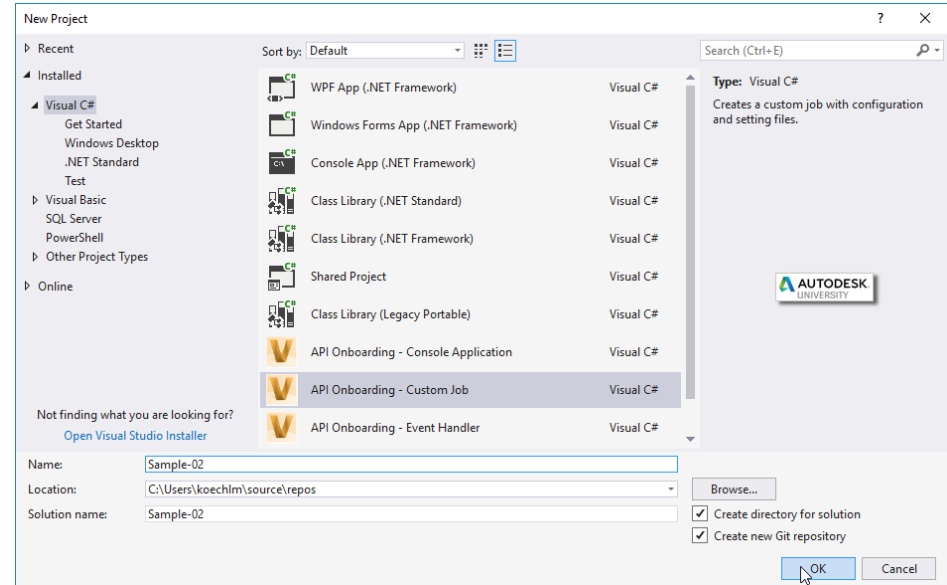
- General steps creating custom jobs
- Best Practice validating job registration
- How to debug Job Processor extensions
- How to consume VaultInventorServer

# Job Processor Extension | Solution Steps

1. Create New Custom Job Project
2. Build & Run from Template
3. Establish Job Debugging
4. Write Code – Job Execution
  - Download source file
  - Process export
  - Upload & attach export-file
5. Complete Error Handling

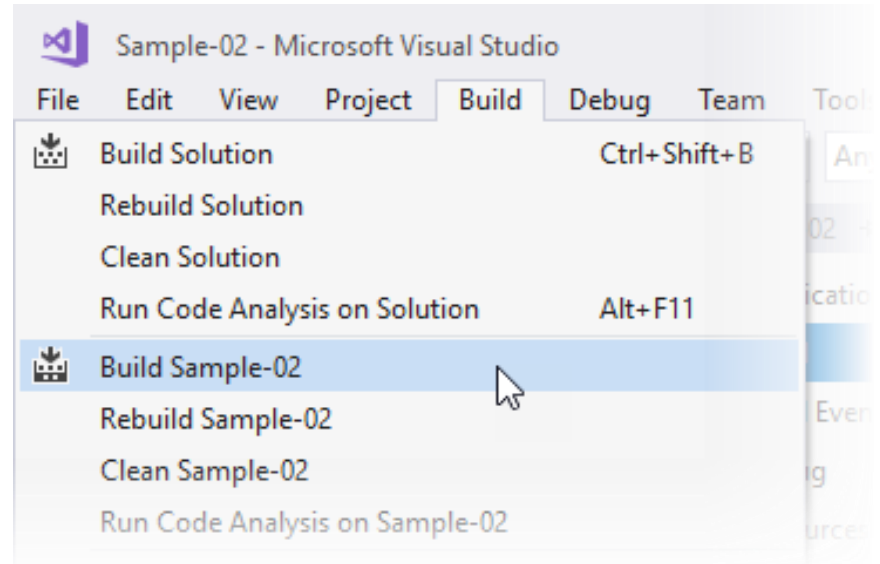
# Job Processor Extension | Solution Steps

1. Create New Custom Job Project
2. Build & Run from Template
3. Establish Job Debugging
4. Write Code – Job Execution
5. Complete Error Handling



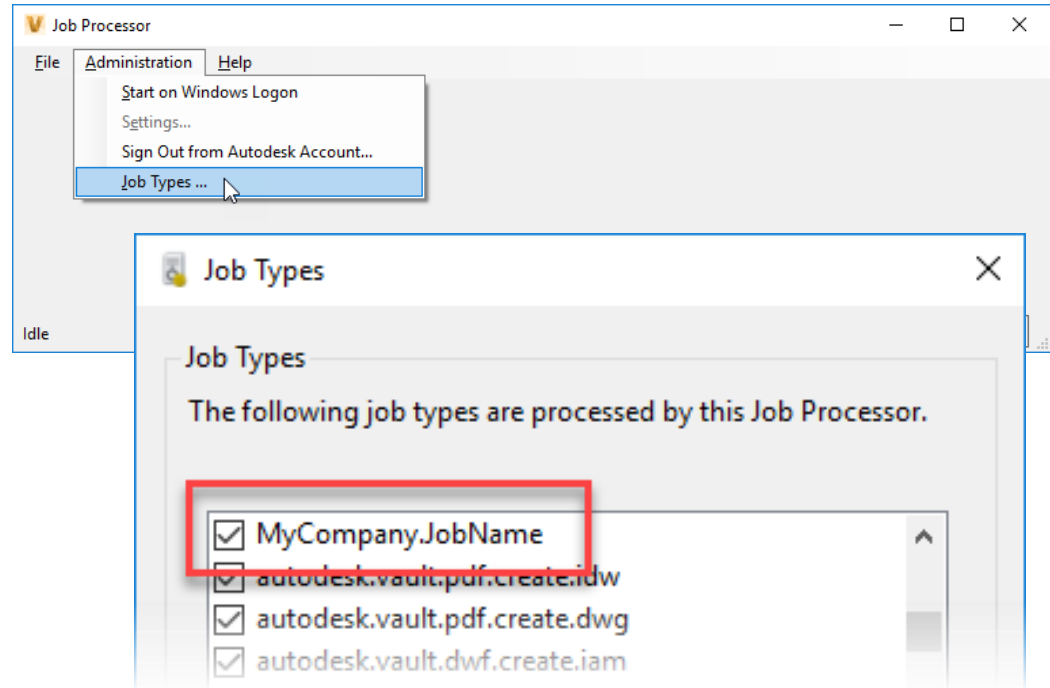
# Job Processor Extension | Solution Steps

1. Create New Custom Job Project
2. Build & Run from Template
3. Establish Job Debugging
4. Write Code – Job Execution
5. Complete Error Handling



# Job Processor Extension | Solution Steps

1. Create New Custom Job Project
2. Build & Run from Template
  - ✓ Validate Job registration
3. Establish Job Debugging
4. Write Code – Job Execution
5. Complete Error Handling



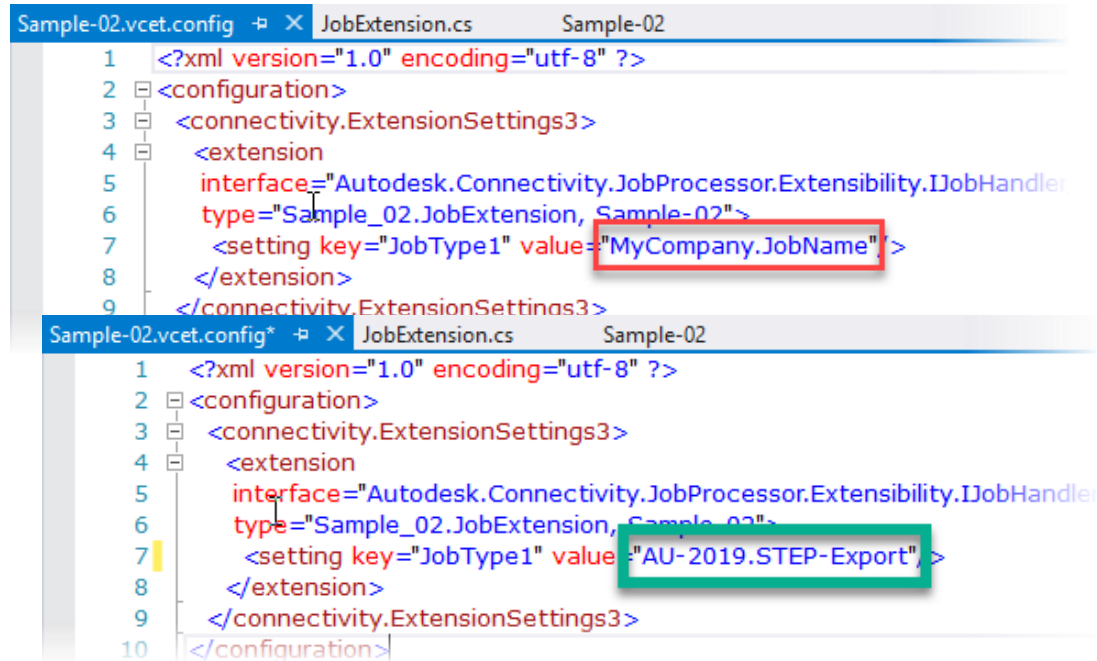
# Job Processor Extension | Solution Steps

1. Create New Custom Job Project
2. Build & Run from Template
  - ☐ Rename the new job in
    - ☐ JobExtension
    - ☐ Config
3. Establish Job Debugging
4. Write Code – Job Execution
5. Complete Error Handling

```
JobExtension.cs | Sample-02
C# Sample-02
15
16 namespace Sample_02
17 {
18     0 references | koechl, 34 minutes ago | 1 author, 1 change
19     public class JobExtension : IJobHandler
20     {
21         private static string JOB_TYPE = "MyCompany.JobName";
22     }
23     #region IJobHandler Implementation
24     0 references | koechl, 34 minutes ago | 1 author, 1 change
25     public bool CanProcess(string jobType)
26     {
27         return jobType == JOB_TYPE;
28     }
29 }
```

# Job Processor Extension | Solution Steps

1. Create New Custom Job Project
2. Build & Run from Template
  - ☐ Rename the new job in
    - ☐ JobExtension
    - ☐ Config
3. Establish Job Debugging
4. Write Code – Job Execution
5. Complete Error Handling



The image shows two screenshots of XML configuration files. The top screenshot shows the 'Sample-02.vcet.config' file with the following content:

```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <configuration>
3   <connectivity.ExtensionSettings3>
4     <extension
5       interface="Autodesk.Connectivity.JobProcessor.Extensibility.IJobHandler
6       type="Sample_02.JobExtension, Sample-02">
7       <setting key="JobType1" value="MyCompany.JobName">
8     </extension>
9   </connectivity.ExtensionSettings3>
```

The bottom screenshot shows the 'Sample-02.vcet.config\*' file with the following content:

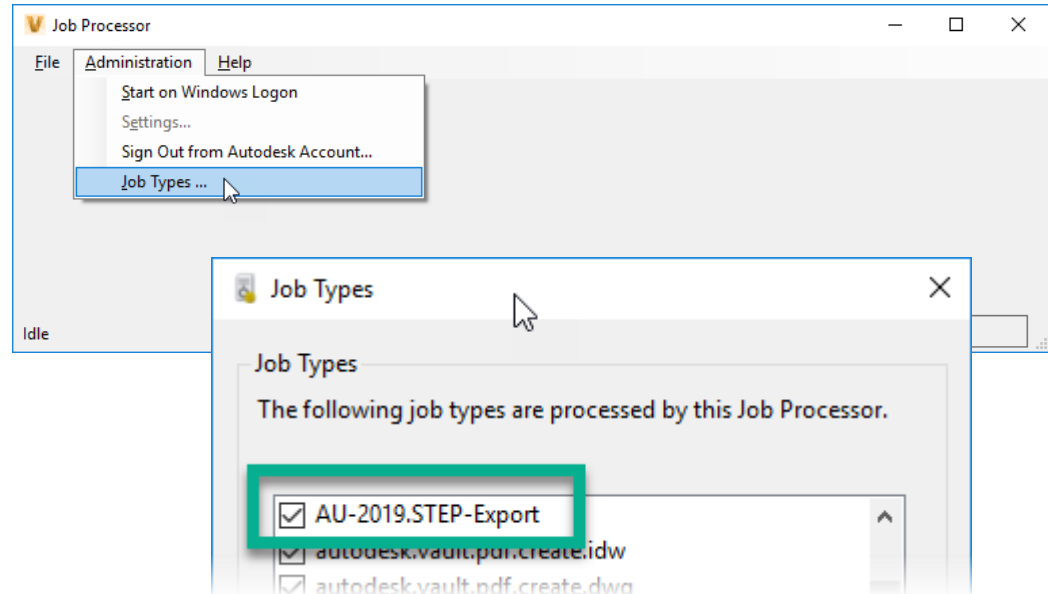
```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <configuration>
3   <connectivity.ExtensionSettings3>
4     <extension
5       interface="Autodesk.Connectivity.JobProcessor.Extensibility.IJobHandler
6       type="Sample_02.JobExtension, Sample_02">
7       <setting key="JobType1" value="AU-2019.STEP-Export">
8     </extension>
9   </connectivity.ExtensionSettings3>
10 </configuration>
```

In both screenshots, the 'value' attribute of the 'JobType1' setting is highlighted with a red box in the top image and a green box in the bottom image.



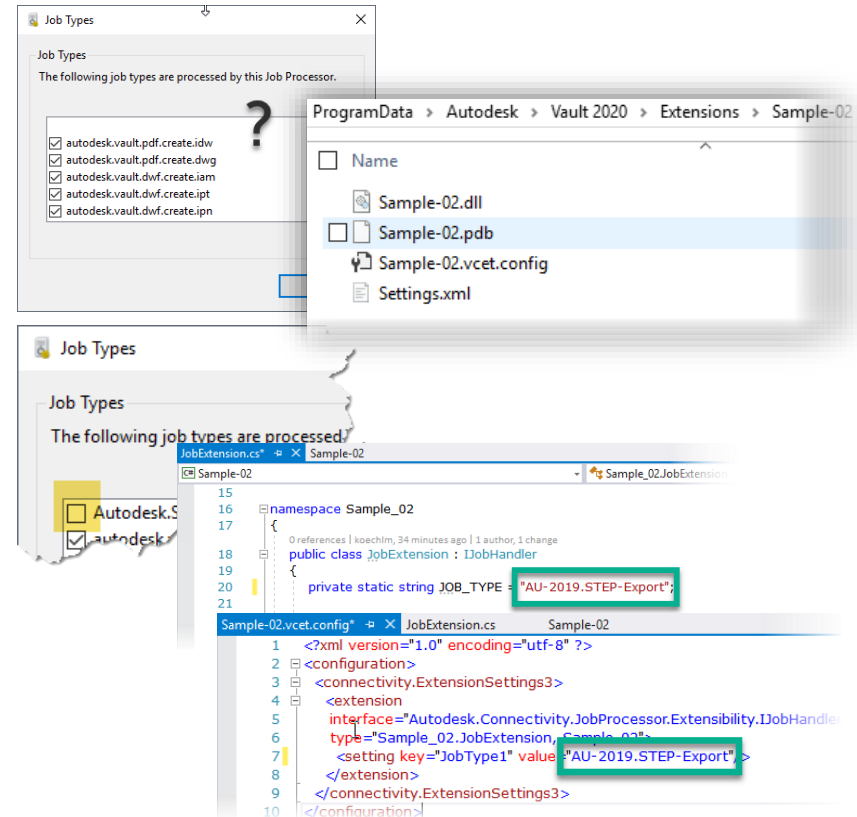
# Job Processor Extension | Solution Steps

1. Create New Custom Job Project
2. Build & Run from Template
  - ✓ Restart Job Processor
  - ✓ Validate Job registration
3. Establish Job Debugging
4. Write Code – Job Execution
5. Complete Error Handling



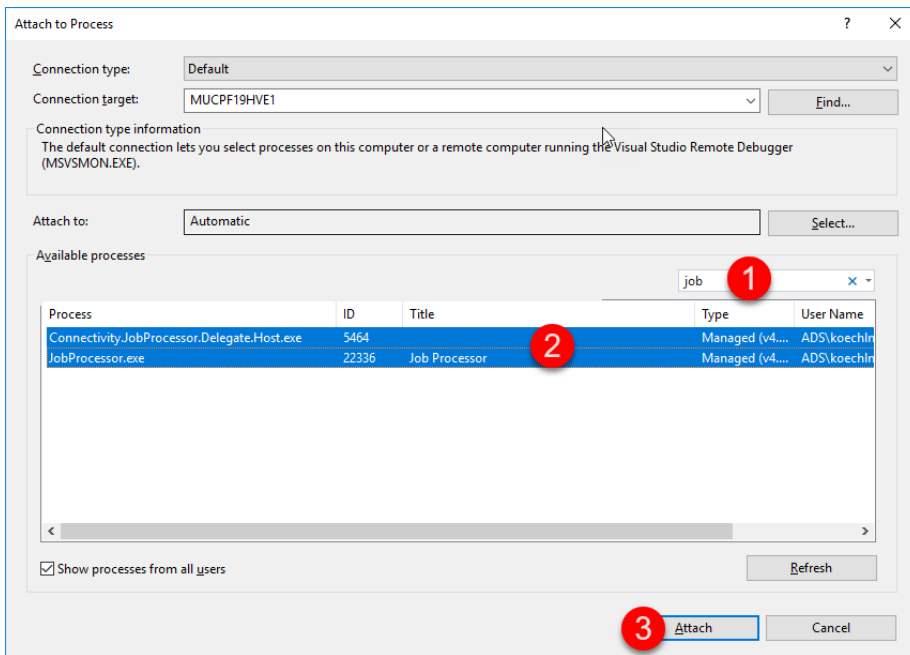
# Excuse | Job Registration – Trouble Shooting

- What failed if..
  - Your new job isn't listed?
  - Check the output folder:
    - .\SubfolderOfExtensions\
    - \*.dll, \*.pdb, \*.vcet.config exist?
- Your new job is listed, but disabled?
  - Check the names match in dll and .vcet.config



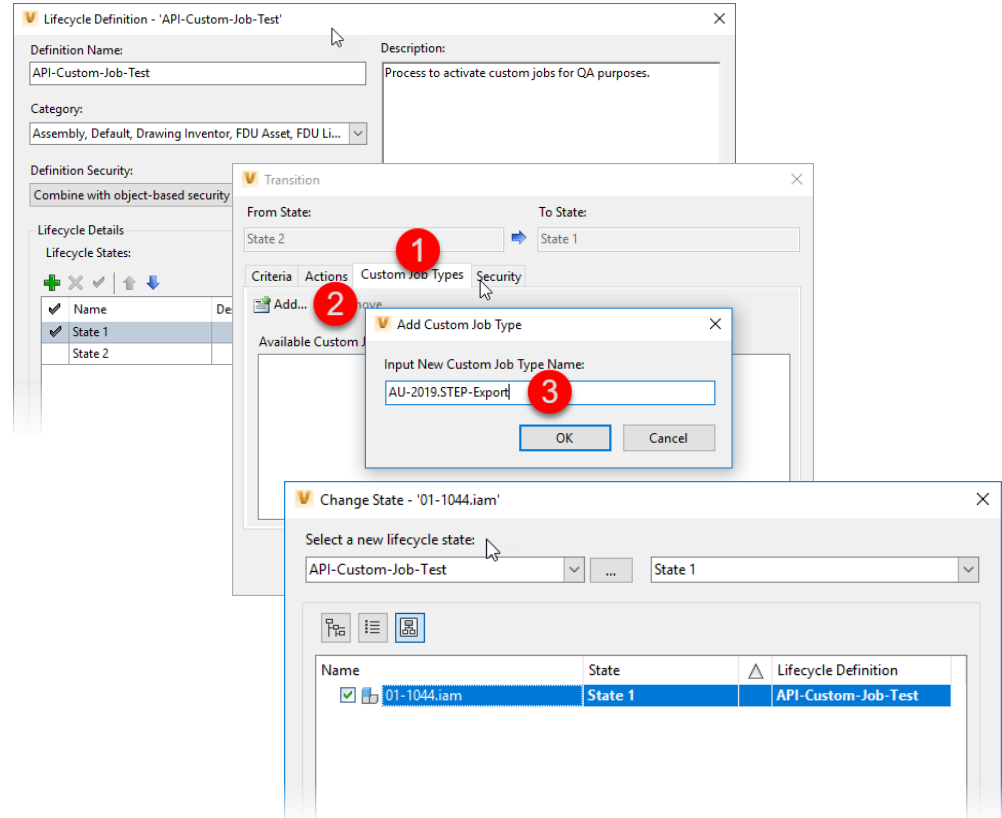
# Job Processor Extension | Solution Steps

1. Create New Custom Job Project
2. Build & Run from Template
3. Establish Job Debugging
  - Attach processes
  - Submit a job
4. Write Code – Job Execution
5. Complete Error Handling



# Job Processor Extension | Solution Steps

1. Create New Custom Job Project
2. Build & Run from Template
3. Establish Job Debugging
  - ✓ Attach processes
  - Submit a job
4. Write Code – Job Execution
5. Complete Error Handling

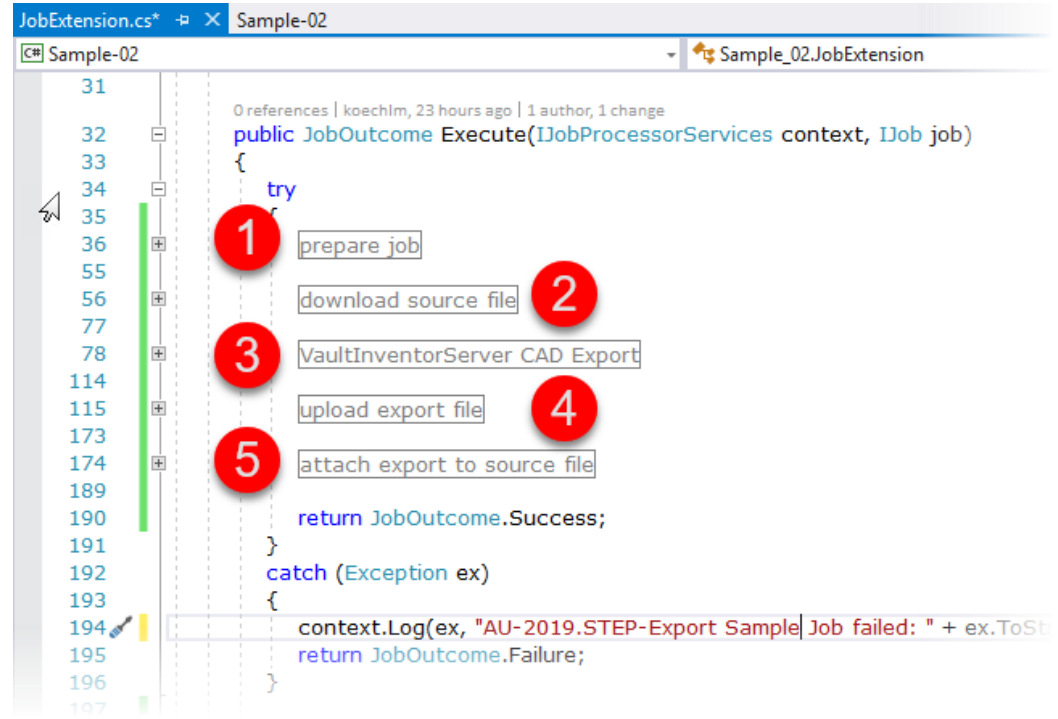


# Demo

- Video
  - Create new custom job project
  - Build and validate
  - Adopt individual job naming
  - Rebuild, re-validate
  - Submit job and debug execution

# Job Processor Extension | Solution Steps

1. Create New Custom Job Project
2. Build & Run from Template
3. Establish Job Debugging
4. Write Code – Job Execution
5. Complete Error Handling



```
JobExtension.cs* X Sample-02
Sample-02
Sample_02.JobExtension

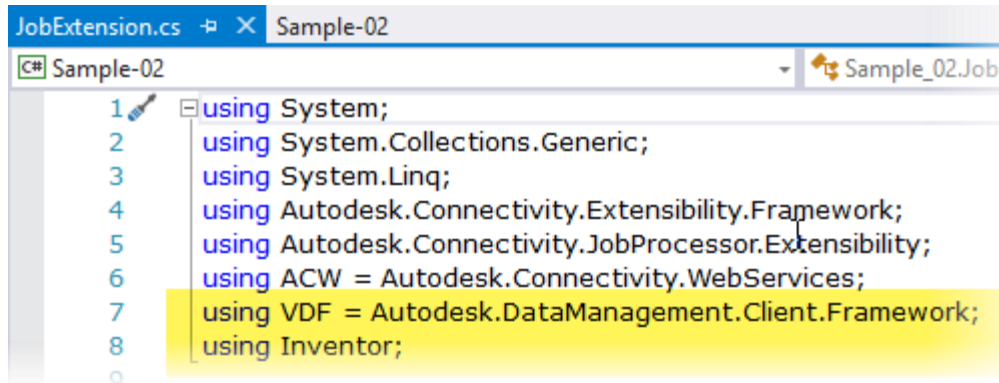
0 references | koehlm, 23 hours ago | 1 author, 1 change
public JobOutcome Execute(IJobProcessorServices context, IJob job)
{
    try
    {
        1 prepare job
        2 download source file
        3 VaultInventorServer CAD Export
        4 upload export file
        5 attach export to source file

        return JobOutcome.Success;
    }
    catch (Exception ex)
    {
        context.Log(ex, "AU-2019.STEP-Export Sample| Job failed: " + ex.ToString());
        return JobOutcome.Failure;
    }
}
```

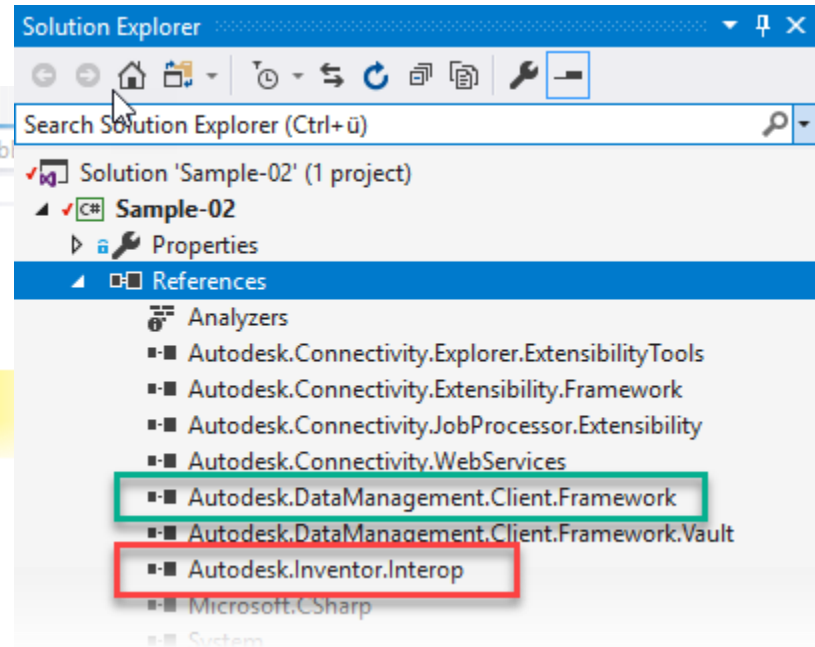
# Job Execution | Sub Task

1 2

- Additional references to be added



```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using Autodesk.Connectivity.Extensibility.Framework;
5 using Autodesk.Connectivity.JobProcessor.Extensibility;
6 using ACW = Autodesk.Connectivity.WebServices;
7 using VDF = Autodesk.DataManagement.Client.Framework;
8 using Inventor;
```



# Job Execution | Sub Task

1

- ❑ Instantiate WebServiceManager – Required to submit API calls
- ❑ Grab the processed file's entity Id

//pick up this job's context

```
Connection connection = context.Connection;
```

```
Autodesk.Connectivity.WebServicesTools.WebServiceManager mWsMgr = connection.WebServiceManager;
```

```
long mEntId = Convert.ToInt64(job.Params["EntityId"]);
```

```
ACW.File mFile = mWsMgr.DocumentService.GetFileById(mEntId);
```

Note – We are getting the particular file version submitted to the job queue



# Job Execution | Sub Task

1



- ❑ Instantiate InventorServer – required to submit Inventor API calls
- ❑ Get the available Inventor translator add-ins

```
//establish InventorServer environment including translator addins; differentiate her in case full Inventor.exe is used
Inventor.InventorServer mInv = context.InventorObject as InventorServer;
ApplicationAddIns mInvSrvAddIns = mInv.ApplicationAddIns;
```

- ❑ Add as many filter's as appropriate

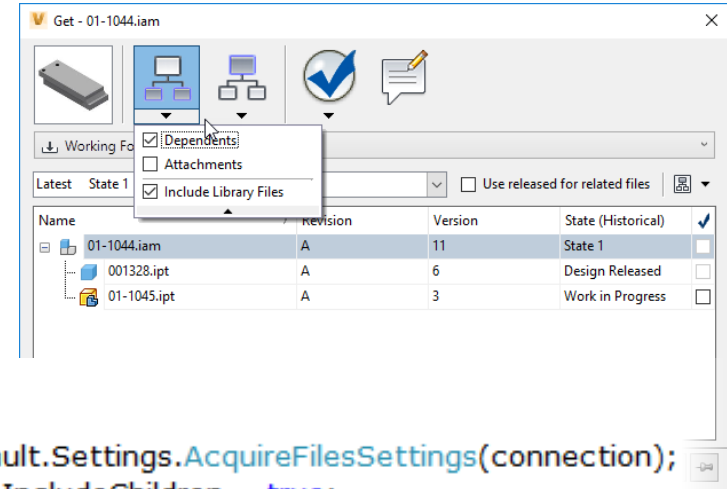
```
// only run the job for ipt and iam file types,
List<string> mFileExtensions = new List<string> { ".ipt", ".iam" };
ACW.File mFile = mWsMgr.DocumentService.GetFileById(mEntId);
if (!mFileExtensions.Any(n => mFile.Name.Contains(n)))
{
    return JobOutcome.Success;
}
```

# Job Execution | Sub Task

2



- ❑ Download the source file(s)
- ❑ Set download options



#region download source file

//download the source file iteration, enforcing overwrite if local files exist

```
VDF.Vault.Settings.AcquireFilesSettings mDownloadSettings = new VDF.Vault.Settings.AcquireFilesSettings(connection);  
mDownloadSettings.OptionsRelationshipGathering.FileRelationshipSettings.IncludeChildren = true;  
mDownloadSettings.OptionsRelationshipGathering.FileRelationshipSettings.IncludeLibraryContents = true;  
mDownloadSettings.OptionsRelationshipGathering.FileRelationshipSettings.ReleaseBiased = true;  
mDownloadSettings.OrganizeFilesRelativeToCommonVaultRoot = true;
```

//execute download

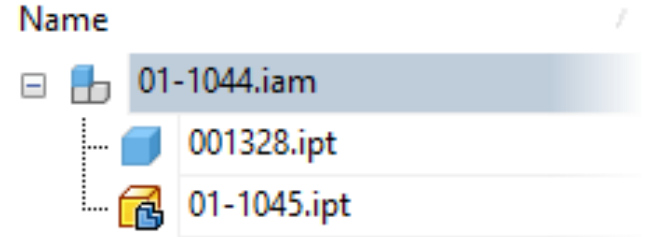
```
VDF.Vault.Results.AcquireFilesResults mDownloadResult = connection.FileManager.AcquireFiles(mDownloadSettings);
```

# Job Execution | Sub Task

2



- ❑ Grab the parent file's local path
  - ❑ Get the file extension for later replacement



//pickup the primary file from result details

```
VDF.Vault.Results.FileAcquisitionResult fileAcquisitionResult =  
    mDownloadResult.FileResults.Where(n => n.File.EntityName == mFileIteration.EntityName).FirstOrDefault();  
string mDocPath = fileAcquisitionResult.LocalPath.FullPath;  
string mExt = System.IO.Path.GetExtension(mDocPath);
```

# Job Execution | Sub Task

3



- ❑ Get the Translator Objects\*<sup>1</sup>
- ❑ Set the specific translator options\*<sup>2</sup>

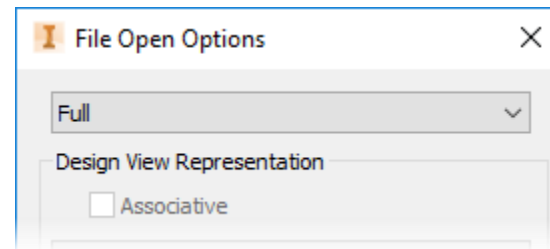
```
#region VaultInventorServer CAD Export
//activate STEP Translator environment,
Document mDoc = null;
try
{
    TranslatorAddIn mStepTrans = mInvSrvAddIns.ItemById["{90AF7F40-0C01-11D5-8E83-0010B541CD80}"] as TranslatorAddIn;
    TranslationContext mTransContext = mInv.TransientObjects.CreateTranslationContext();
    NameValueMap mTransOptions = mInv.TransientObjects.CreateNameValueMap();
    if (mStepTrans.HasSaveCopyAsOptions[mDoc, mTransContext, mTransOptions] == true)
    {
        mTransOptions.Value["ApplicationProtocolType"] = 3; //AP 2014, Automotive Design
        mTransOptions.Value["Description"] = "Sample-Job Step Translator using VaultInventorServer";
        mTransContext.Type = IOMechanismEnum.kFileBrowseIOMechanism;
    }
}
```

## Job Execution | Sub Task

3



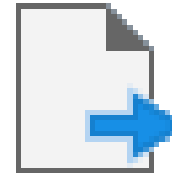
- ❑ Open source file using option “Full Mode”



```
//open (enforce full load on open, as translators don't work in express mode)  
NameValueMap mOpenOptions = mInv.TransientObjects.CreateNameValueMap();  
mOpenOptions.Add("ExpressModeBehavior", "OpenFull");  
mDoc = mInv.Documents.OpenWithOptions(mDocPath, mOpenOptions, false);
```



- ❑ Save translator file



Export

//translate writing the export file

```
DataMedium mData = mInv.TransientObjects.CreateDataMedium();  
mData.FileName = mDocPath.Replace(mExt, ".stp");  
mStepTrans.SaveCopyAs(mDoc, mTransContext, mTransOptions, mData);  
mDoc.Close(true);
```

# Job Execution | Sub Task

4



## ❑ Upload export file to Vault

✓ First time – add file

```
ACW.File wsFile = mWsMgr.DocumentService.FindLatestFilesByPaths(vaultFilePath.ToSingleArray()).First();
if (wsFile == null || wsFile.Id < 0)
{
    // upload file to Vault
    if (mFolder == null || mFolder.Id == -1)
        throw new Exception("Vault folder " + mFolder.FullName + " not found");

    var folderEntity = new Autodesk.DataManagement.Client.Framework.Vault.Currency.Entities.Folder(connection, mFolder);
    try
    {
        addedFile = connection.FileManager.AddFile(folderEntity, "Created by Job Processor", null, null, ACW.FileClassification.DesignRepresentation,
        mExpFile = addedFile;
    }
    catch (Exception ex)
```

# Job Execution | Sub Task

4



- ☐ Upload export file to Vault
  - ✓ Repeated time
    - ☐ check-out existing
    - ☐ Check-in new file

```
// checkin new file version
VDF.Vault.Settings.AcquireFilesSettings aqSettings = new VDF.Vault.Settings.AcquireFilesSettings(connection)
{
    DefaultAcquisitionOption = VDF.Vault.Settings.AcquireFilesSettings.AcquisitionOption.Checkout
};
mResOpt.OverwriteOption = VDF.Vault.Settings.AcquireFilesSettings.AcquireFileResolutionOptions.OverwriteOptions.NoOverwrite;
vdfFile = new VDF.Vault.Currency.Entities.FileIteration(connection, wsFile);
aqSettings.AddEntityToAcquire(vdfFile);
var results = connection.FileManager.AcquireFiles(aqSettings);
try
{
    mUploadedFile = connection.FileManager.CheckinFile(results.FileResults.First().File, "Created by Job Processor", false, null, null, false, null, ACW.FileClassification.DesignRepresentation);
    mExpFile = mUploadedFile;
}
catch (Exception ex)
{
}
```



## Job Execution | Sub Task 5



- ❑ Attach export file to source – using type “DesignRepresentationFileAttachment”

```
ACW.FileAssocParam mAssocParam = new ACW.FileAssocParam();  
mAssocParam.CldFileId = mExpFile.Id;  
mAssocParam.ExpectedVaultPath = mWsMgr.DocumentService.FindFoldersByIds(new long[] { mFile.FolderId }).First().FullName;  
mAssocParam.RefId = null;  
mAssocParam.Source = null;  
mAssocParam.Type = ACW.AssociationType.Attachment;  
mWsMgr.DocumentService.AddDesignRepresentationFileAttachment(mFile.Id, mAssocParam);
```

# Job Processor Extension | Solution Steps

1. Create New Custom Job Project
2. Build & Run from Template
3. Establish Job Debugging
4. Write Code – Job Execution

## 5. Complete Error Handling

Minimum = Feedback Success/Failure

```
catch (Exception ex)
{
    throw new Exception("Job could attach the export result.", ex);
}
#endregion attach

return JobOutcome.Success;
}
catch (Exception ex)
{
    context.Log("AU-2019.STEP-Export Sample Job failed: " + ex.ToString(), MessageType.eError);
    return JobOutcome.Failure;
}
```

## Use Case 2 | Takeaways

- Always validate your custom job's framework before writing code
  - Job name registration | Job submission | Debugging
- Download settings are essential for successful job execution
  - Plan these using UI command Get... first
- The sample solution relies on enforced Vault settings for Inventor project and working folder
  - Varying working folders require download and activation of Inventor project file(s)
- The sample solution does not synchronize file status and properties
  - Get a fully functional sample solution from [my GitHub](#) repository.

# Use Case 3 – Event Handler



# Use Case 3 | CRM/ERP/... Link to Vault Thin Client

- Frequently other business systems expect information, files or links from Vault
- Instead of pushing files – in our sample we create Thin Client links for sharing

The screenshot displays the Autodesk Vault Thin Client interface. On the left, a navigation pane shows modules like Purchasing, Inventory, Production, and HRP. The main window is titled 'Bill of Materials (Resource List) (Extended by vanatec BX)' and shows a BOM for 'Red Sports Bike'. A 'BOM Operation Details' dialog is open, showing operation codes and times. A yellow circle highlights the 'min' button in the dialog. On the right, a 'File Details' panel shows a list of 3D component files, including 'SMC-002-001.iam', 'SMC-200-022.ipt', 'SMC-200-017.ipt', and 'SMC-200-018.ipt'. A yellow arrow points from the 'min' button in the dialog to the 'Documentation' button in the 'File Details' panel.

Autodesk Vault

10.49.83.79/AutodeskTC/10.49.83.79/MFG-2017-PRO-EN#/Entity/Details?id=mRkIMTpMQVRfU1Q6MzUxODA&itemtype... Administrator (Read Only) - Help

MFG-2017-PRO-EN Project Explorer (\$) Designs Asks Special Purchased Parts

File Details

SMC-002-001.iam

Category Name: 3D component  
Revision: -  
State: Released  
Comment: Property Edit

System

Attachments  
Category Glyph  
Category Glyph (Historical)  
Category Name

3D component

SMC-200-022.ipt  
Revision: -  
State (Historical): Released  
Created By: JobProcessor1  
Category Name: 3D component  
Comment: Property Edit

SMC-200-017.ipt  
Revision: -  
State (Historical): Released  
Created By: JobProcessor1  
Category Name: 3D component  
Comment: Property Edit

SMC-200-018.ipt  
Revision: -  
State (Historical): Released  
Created By: JobProcessor1  
Category Name: 3D component  
Comment: Property Edit

## Use Case 3 | Solution Approach

- Subscribe to the “New Folder Event” to create Thin Client link



# Use Case 3

## Learning Objectives

- General steps creating a web service event handler
- Benefits of webservice events
- Benefits of event restrictions
- Limits of event handling

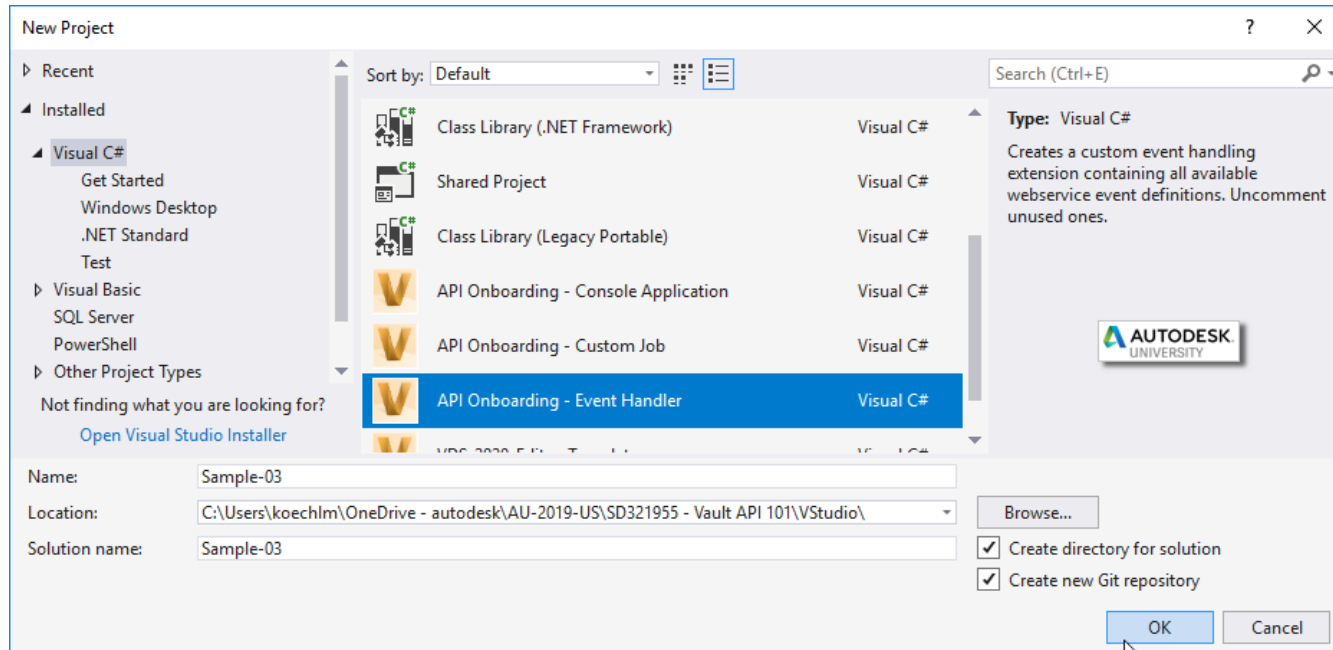
# Use Case 3 | Solution Steps

1. Create new Event Handler Extension
  - Prepare for debugging
2. Coding - TC link
  - Create
  - Save to UDP
3. Coding - restriction on folder move



# Event Handler Extension | Solution Steps

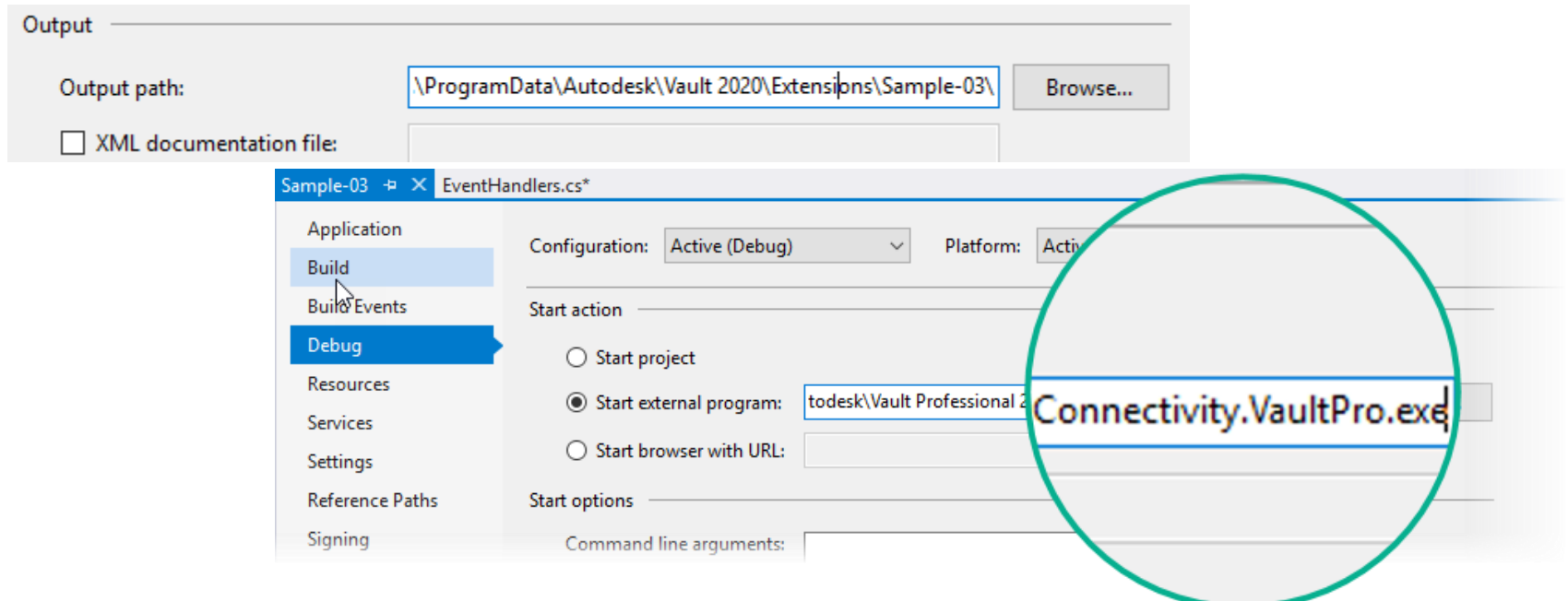
## 1. Create new Event Handler Extension



# Event Handler Extension | Solution Steps

## 1. Create new Event Handler Extension

- Prepare for debugging



# Event Handler Extension | Solution Steps

## 1. Create new Event Handler Extension

- Prepare for debugging

## 2. Coding - TC link

```
// Folder Events
//DocumentService.AddFolderEvents.GetRestrictions += new EventHandler<AddFolderComm
//DocumentService.AddFolderEvents.Pre += new EventHandler<AddFolderComm
DocumentService.AddFolderEvents.Post += new EventHandler<AddFolderComm
//DocumentService.DeleteFolderEvents.GetRestrictions += new EventHandler<D
//DocumentService.DeleteFolderEvents.Pre += new EventHandler<DeleteFolderC
//DocumentService.DeleteFolderEvents.Post += new EventHandler<DeleteFolderC
DocumentService.MoveFolderEvents.GetRestrictions += new EventHandler<Mov
//DocumentService.MoveFolderEvents.Pre += new EventHandler<MoveFolderCor
//DocumentService.MoveFolderEvents.Post += new EventHandler<MoveFolderC
```

# AddFolderEvents\_Post | Solution Steps

## 1. Create new Event Handler Extension

- Prepare for debugging

## 2. Coding - TC link

```
private void AddFolderEvents_Post(object sender, AddFolderCommandEventArgs e)
{
    //several functions require the entity class name of Folder
    string mEntCls = "FLDR";

    //get the new folder's object
    Autodesk.Connectivity.WebServices.Folder mFolder = e.ReturnValue;
    if (mFolder.Cat.CatName != "Project")
    {
        return; //we want TC link for projects only
    }

    create TC Link

    Add property/value (TC Link)
}
```

1

2

# AddFolderEvents\_Post | Sub Task

1

- ❑ Get the WebServiceManager to access all Services
- ❑ Call KnowledgeVaultService.GetPersistentIds()

```
//the sender is DocumentServiceExtension ; cast the object
DocumentServiceExtensions mDocSrcvExtensions = (DocumentServiceExtensions)sender;
//get the WebServiceManager object; we need later to use other services than the DocumentService
Autodesk.Connectivity.WebServicesTools.WebServiceManager mWsMgr = mDocSrcvExtensions.WebServiceManager;

//use KnowledgeVaultService to retrieve the persistent id; folders don't have history, the option Latest is not relevant here
string[] mPersIDs = mWsMgr.KnowledgeVaultService.GetPersistentIds(mEntCls, new long[] { mFolder.Id }, EntPersistOpt.Latest);
string mPersID = mPersIDs[0];
mPersID = mPersID.TrimEnd(mPersID[mPersID.Length - 1]);
```

# AddFolderEvents\_Post | Sub Task

1


## ❑ Construct the Thin Client link

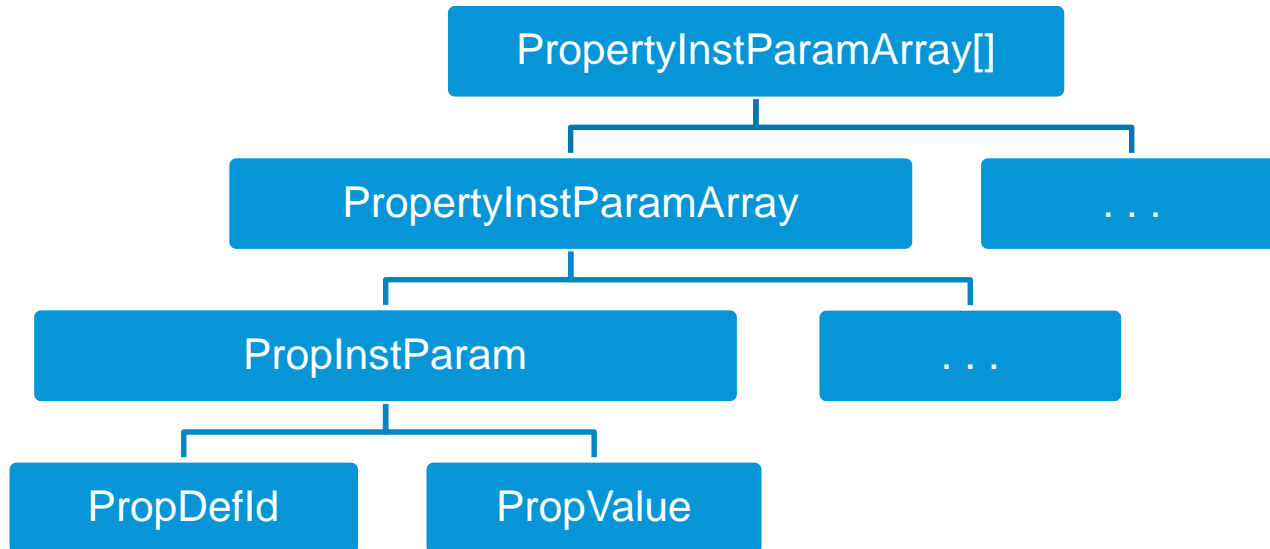
'<ThinClient Service URL>/AutodeskTC/<ServerName>/<VaultName>/#/Entity/Entities?folder=<PersistentID>&start=0

```
//put together the Thin Client Link;  
string mTCLink = null;  
Uri mServerUri = new Uri(mDocService.Url);  
string mVaultName = mWsmMgr.WebServiceCredentials.VaultName;
```

## AddFolderEvents\_Post | Sub Task

2

- ❑ Call the method `DocumentServiceExtensions.UpdateFolderProperties()`
- ❑ Don't be afraid of creating the property-instance-array array 



## AddFolderEvents\_Post | Sub Task

2

- Call the method `DocumentServiceExtensions.UpdateFolderProperties()`

```
List<long> mFldrIds = new List<long>{...};
```

```
//get the folder property definition Id using the displayname
```

```
PropDef[] mPropDefs = mWsMgr.PropertyService.GetPropertyDefinitionsByEntityClassId(mEntCls);
```

```
PropDef mPropDef = mPropDefs.SingleOrDefault(n => n.DispName == "ThinClient Link");
```

```
PropInstParam mPropInstParam = new PropInstParam();
```

```
PropInstParamArray mPropInstParamArray = new PropInstParamArray();
```

```
List<PropInstParam> mPropInstParamList = new List<PropInstParam>();
```

```
List<PropInstParamArray> mPropInstParamArrayList = new List<PropInstParamArray>();
```

```
if (mPropDef != null)//create the nested property instance array(s) and run the update  
{
```

```
    mPropInstParam.PropDefId = mPropDef.Id;
```

```
    mPropInstParam.Val = mTCLink;
```

```
    mPropInstParamList.Add(mPropInstParam);
```

```
    mPropInstParamArray.Items = mPropInstParamList.ToArray();
```

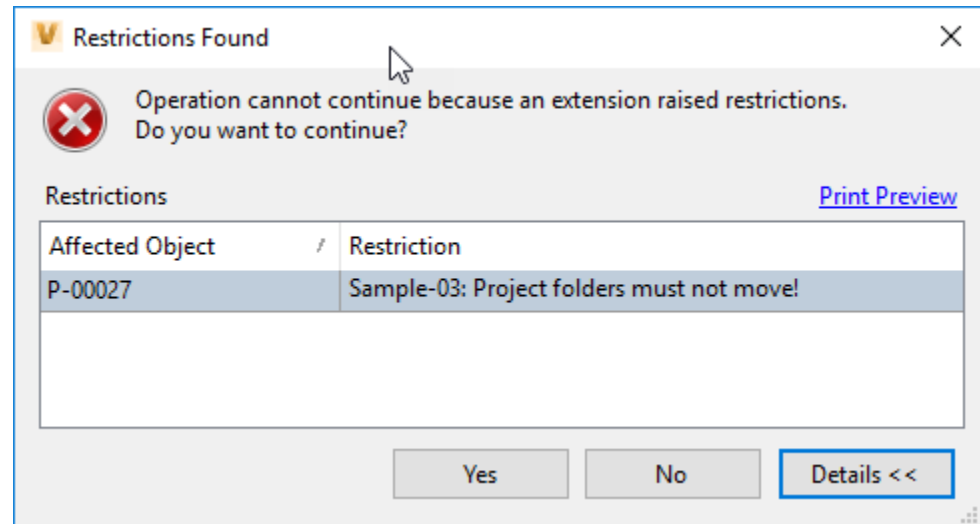
```
    mPropInstParamArrayList.Add(mPropInstParamArray);
```

```
    mWsMgr.DocumentServiceExtensions.UpdateFolderProperties(mFldrIds.ToArray(), mPropInstParamArrayList.ToArray());
```



# Use Case 3 | Solution Steps

1. Create new Event Handler Extension
2. Coding - TC link
3. Coding - restriction on folder move



## MoveFolderEvent\_GetRestrictions

```
//DocumentService.DeleteFolderEvents.Pre += new EventHandler<DeleteFolderEventArgs> (sender, e)
//DocumentService.DeleteFolderEvents.Post += new EventHandler<DeleteFolderEventArgs> (sender, e)
DocumentService.MoveFolderEvents.GetRestrictions += new EventHandler<MoveFolderCommandEventArgs> (sender, e)
//DocumentService.MoveFolderEvents.Pre += new EventHandler<MoveFolderCommandEventArgs> (sender, e)
//DocumentService.MoveFolderEvents.Post += new EventHandler<MoveFolderCommandEventArgs> (sender, e)
```

```
private void MoveFolderEvent_GetRestrictions(object sender, MoveFolderCommandEventArgs e)
{
    //the sender is DocumentService; cast the object
    DocumentService mDocService = (DocumentService)sender;

    //get the new folder's object
    Autodesk.Connectivity.WebServices.Folder mFolder = mDocService.GetFolderById(e.FolderId);

    if (mFolder.Cat.CatName == "Project")
    {
        e.AddRestriction(new ExtensionRestriction(mFolder.Name, "Project folders must not move!"));
    }
}
```

# Demo

## Use Case 3 | Takeaways

- Benefit of events
  - Execute what can be done better now, than later
  - Apply to any “sender”
  - May not allow any action – combine with restrictions
- Limits of events
  - The user waits for event’s pre and post actions – consider jobs for larger tasks
    - A post action could submit an job to the queue for large task too
  - Circular iterations
  - The sender’s connected user shares roles and permissions

# Summary | Decision Criteria

	Standalone	Job Proc. Extension	Event Handler
Vault client installation required?	Yes – Computer license No – Server licensing	Yes	Yes
Task size	Small . . large	Small . . large	Small
Execution – point of time	Individual	<ul style="list-style-type: none"><li>• One time queue submission + priority</li><li>• Time interval queue submission + priority</li></ul>	Immediate

Where to go next?



# Valuable Links & Downloads

- Actual | Up-to-date
  - My personal GitHub repository <https://github.com/koechlm?tab=repositories>
  - Autodesk Manufacturing DevBlog <http://adndevblog.typepad.com/manufacturing/>
  - coolOrange Blog <https://blog.coolorange.com/>
- Legacy | Valid for ever...
  - Link Collection <http://justonesandzeros.typepad.com/blog/2013/08/various-articles-from-around-the-web.html>
  - Webcast-Training Archive <http://adndevblog.typepad.com/manufacturing/2013/05/api-webcast-archive.html>

# Where to find peers? Where to go for answers?

- Autodesk Community Vault API & Vault Data Standard → [Customization Forum](#)



## VAULT PRODUCTS

[LEARN](#)[DOWNLOADS](#)[TROUBLESHOOTING](#)[FORUMS](#)

This board



Search



## Vault Customization

[POST TO FORUMS](#)[All Posts](#)[FAQs](#)[Accepted Solutions](#)[Unanswered](#)

### Forums Links

- [➔ Back to Vault Category](#)
- [➔ All Forums](#)
- [➔ All Ideas](#)
- [➔ Help](#)

### Trending Topics



# Other Classes You May Be Interested In

## SD323470 - FEEL THE POWER BETWEEN VAULT AND POWERSHELL

- Tuesday, 4:30 PM - 05:30 PM

The Vault client in itself is a powerful data utility, but just imagine there's so much more that can be done with the Vault software development kit (SDK) software. Learn how to use the versatility and flexibility of Microsoft PowerShell, and bring your logic to new levels of automation. Learn how to instantiate Vault objects, call Vault Web Service methods, and manipulate the data stored within Vault in a lightweight scripted .NET language.

## Please Fill Out Your Surveys

Make sure your voice is heard by completing your surveys!

Please take the time to complete your survey for this and every class you attend at Autodesk University.

Autodesk uses this information to know what classes to offer in the future.





# AUTODESK®

## Make anything™

Autodesk and the Autodesk logo are registered trademarks or trademarks of Autodesk, Inc., and/or its subsidiaries and/or affiliates in the USA and/or other countries. All other brand names, product names, or trademarks belong to their respective holders. Autodesk reserves the right to alter product and services offerings, and specifications and pricing at any time without notice, and is not responsible for typographical or graphical errors that may appear in this document.

© 2019 Autodesk. All rights reserved.

