IM468712

iLogic & Vault | Vault & iLogic

Markus Koechl

Autodesk

Learning Objectives

- Identify the benefits of managing iLogic rules and settings in Vault
- List benefits of connecting iLogic rules to Vault
- Create iLogic rules that query Vault, retrieve files, or automate new file generation using familiar iLogic rule syntax
- Solve automation tasks by running iLogic rules as custom Jobs in Vault Job Processor
- Explain the differences of writing iLogic rules for Inventor and Vault Job Processor

Description

iLogic is the by far most used application within Inventor. Autodesk Vault is the most used PDM system for Inventor. This class elaborates on how both can achieve synergy by connecting. Attending this class, you will answer these four questions:

- What can you achieve for iLogic rule execution if your rule connects to Vault?
- How can you achieve the iLogic-Vault connection without knowing the Vault API?
- What can you achieve for Vault Job Processor if your job connects to iLogic?
- How can you leverage iLogic rules while running custom jobs?

Speaker

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

TABLE OF CONTENT

iLogic & Vault

Benefits of Managing iLogic Rules and Settings in Autodesk Vault

A Vault administrator can manage all users' access to external iLogic rules and set files based on lifecycles or object-based permissions. As a best practice, all files consolidate in a single folder structure. For this class sample implementation, we extended the Inventor iLogic default location managing External Forms as part of the Design Data library (Figure 1).

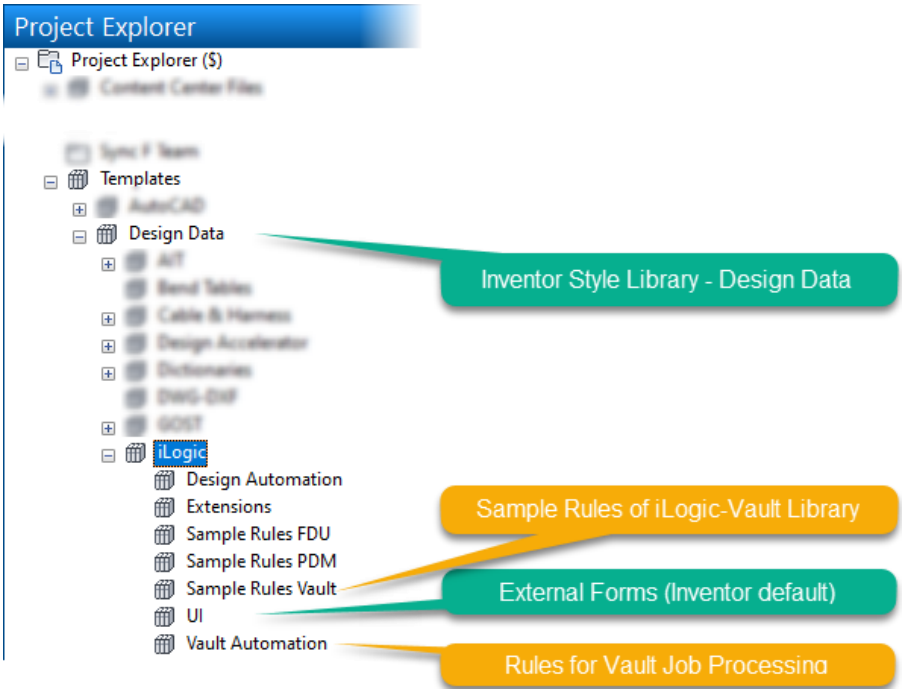


Figure 1

Benefits of managing rules by a Vault Lifecycle:

- Read/Write access for Rule Editors to rules in the state "Work in Progress."
- Optional – Download restriction for non-editors for rules in the state "Work in Progress."
- Read access for all iLogic rule consumers to rules in the state "Released."
- History of rule changes and revisions (Figure 2)

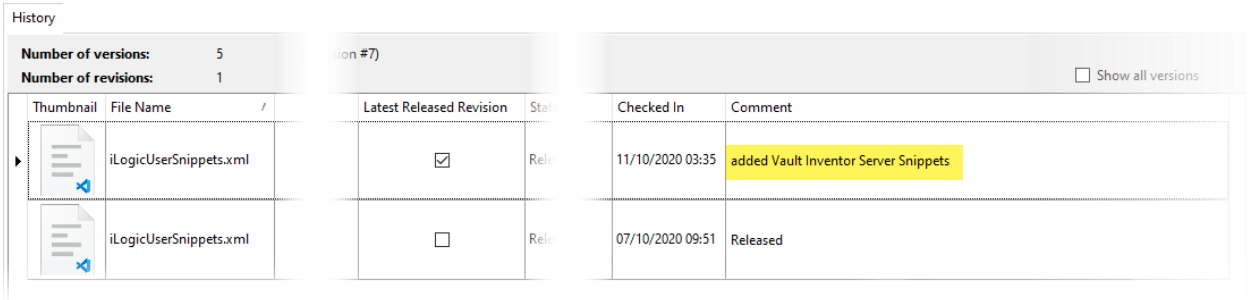


Figure 2

Benefits of managing iLogic settings by a Vault Lifecycle

- Restrict the change of settings to CAD Administrators or iLogic Rule Authors (Figure 4)
- Restrict to Edit or Add External Forms by managing the file iLogicBrowserUIFormSpecification.xml (Figure 3)

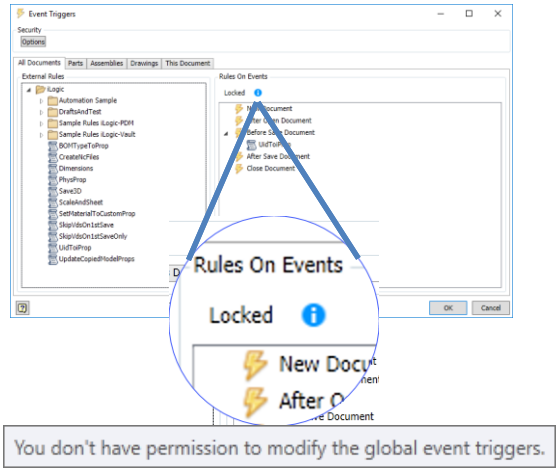


Figure 4

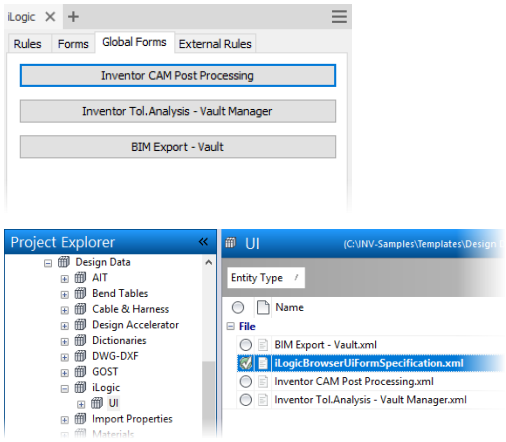


Figure 3

Tip – Include iLogic Rule and Settings in Thunderdome Deployments

The Vault Extension "Project Thunderdome for Autodesk Vault" includes all rules and setting files in Client deployments. Use the option "Enforce Deployment" to automatically share all changes to any client. Click on Figure 5 to navigate to Autodesk App Store.

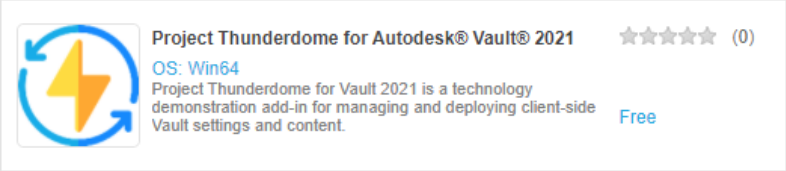


Figure 5

Benefits of Connecting iLogic Rules with Vault

There are many use cases and automation workflows that benefit from direct accessing Vault files by iLogic rules. This class picked two frequently asked solutions demonstrating what you can achieve by searching in Vault or downloading needed files from Vault.

Dynamic Rules

This class defines a static Multi-Value as a Text Parameter Multi-Value list saved in an Inventor file (Figure 6) or rule.

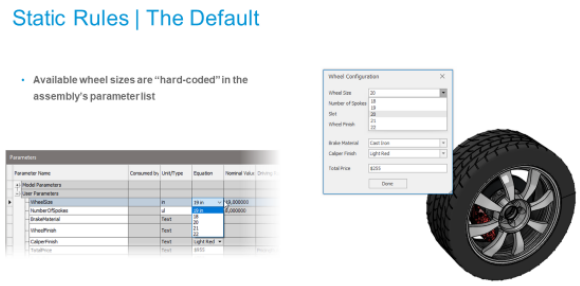


Figure 6

We called rules that actively retrieve list definitions based on a search result from Vault "Dynamic Rules" (Figure 7).

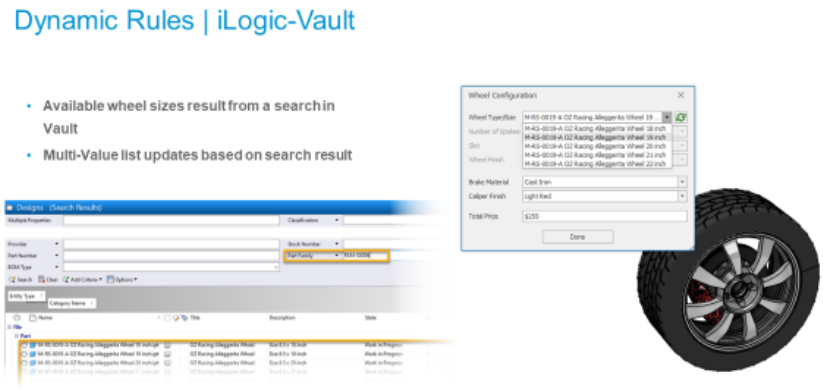


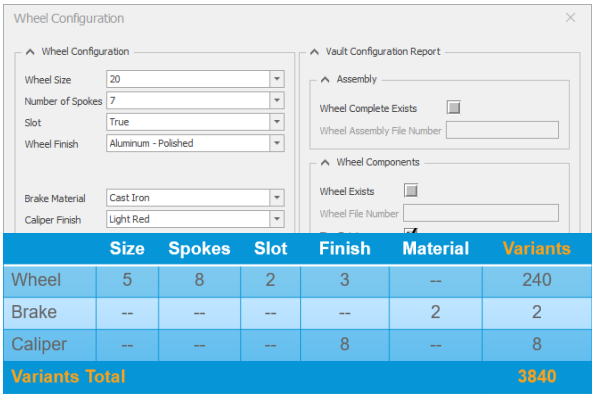
Figure 7

Benefits:

- Dynamic rules deliver individual results based on the Vault connected to or related to an individual company Vault's content.
- Dynamic rules or the Inventor part or assembly file do not require any change to update or edit Multi-Value lists and their consuming iLogic Forms.

Design-Reuse

Building assembly variants by updating multiple parameters may result in an unmanageable number of variants and files. (Figure 8)



	Size	Spokes	Slot	Finish	Material	Variants
Wheel	5	8	2	3	--	240
Brake	--	--	--	--	2	2
Caliper	--	--	--	8	--	8
Variants Total						3840

Figure 8

A valid strategy to avoid this is not to save the configured variant and derive neutral files and drawings like STEP and PDF files from the updated model.

Another strategy is managing all created combinations and variants in the Vault. A lookup of iLogic rules in Vault for existing variants and model sizes is critical to realize this strategy. The sample model does not update the model with each parameter change; instead, the "Refresh Configuration Preview" button runs a rule that updates the current model's sizing and components and reports search results from Vault about matching variants existing in Vault. Based on the search result, the rule can open and re-use existing components or assembly models (Figure 9) or run another rule creating a new assembly by adding new to or re-using existing Vault components. (Figure 10)

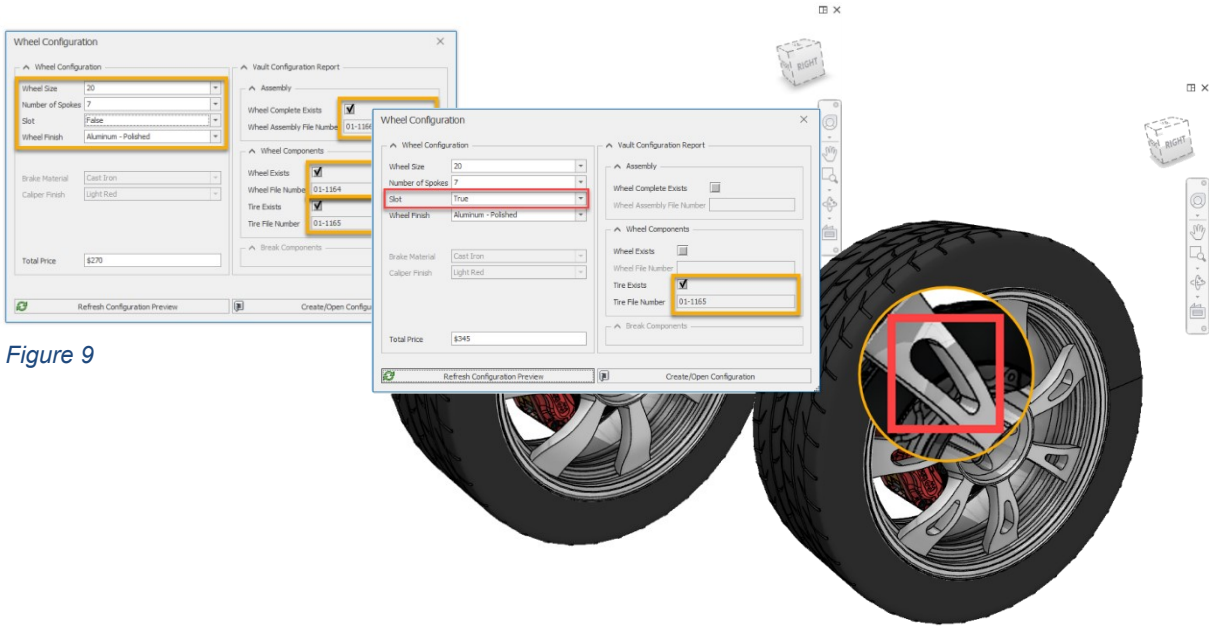


Figure 9

Figure 10

This class provides access to an iLogic extension library that shares methods to search and download files from Autodesk Vault; using this library does not require any Vault API access and programming skills. The next chapter provides information to access, install, and use this library to re-build the class' sample solutions on your computer.

Writing iLogic-Vault Rules

Note – This class target audience is educated iLogic Rule Authors and expects to knowledge about managing files in Autodesk Vault Workgroup or Professional. The instructions on writing iLogic-Vault rules do not talk you through any step to build the sample models and rules. They describe major steps and concepts on how to achieve the results demonstrated in the class' presentation.

Download the Inventor iLogic Trial Sample Dataset and Step-by-Step-Guide

Inventor Dataset:
https://www.autodesk.com/content/dam/autodesk/www/campaigns/inventor-resource/ilogic/inventor-project-ilogic-workflow-dataset.zip

Step-by-Step-Guide:
https://damassets.autodesk.net/content/dam/autodesk/www/campaigns/inventor-resource/ilogic/inventor-project-ilogic-workflow-guide-en.pdf

Open the Inventor main assembly file "WheelAssembly.iam" in Inventor and complete all steps following the step-by-step-guide. Save and check-in the result into your Vault. By completing this step, your model behaves, as shown in Figure 6.

Download and Install the iLogic-Vault sample library, "QuickstartiLogicLibrary."

Download the latest release from this open-source repository:
https://github.com/koechlm/iLogic-Vault/releases/latest

Follow the web page instructions on reading and accepting the disclaimer text file before opening and using this sample.

Install the Extension library by copying the library files to the iLogic Addins Directory configured on your computer (Figure 11).

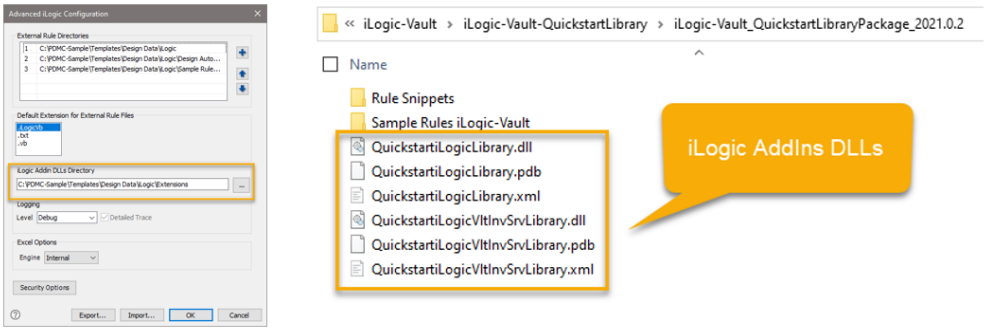


Figure 11

Merge the iLogic-Vault Snippets with your User Snippets, and do not forget saving the snippet file. (Figure 12)

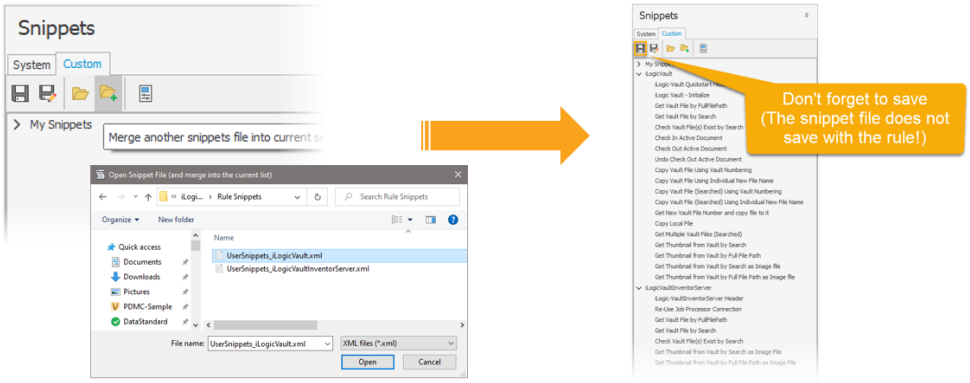


Figure 12

Copy the iLogic-Vault rule sample files to one of your configured External Rule Directories. (Figure 13)

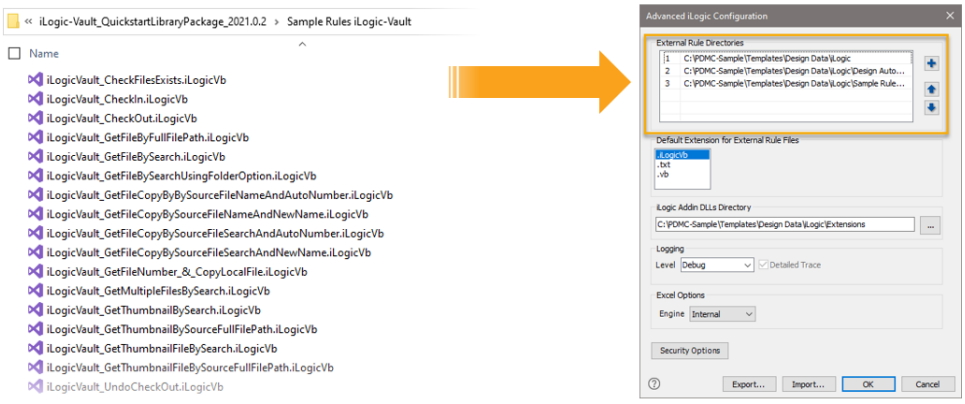


Figure 13

Your first "Dynamic Rule"

Create a copy of the WheelAssembly.iam while replacing the wheel part with a copy using Vault Copy Design.

The file names used in the video demonstrations are:

WheelAssembly_VaultSearch.iam

M-RS-0019-A OZ Racing Alleggerita Wheel 20 inch.ipt

Open the new assembly and select the 20-inch size to update the wheel's size accordingly.

Create a new Text Parameter "Wheel" with text "M-RS-0019-A OZ Racing Alleggerita Wheel 20 inch" (Figure 14, **(1)**) and change the rule to replace the wheel with changing sizes (Figure 14 **(2)**).

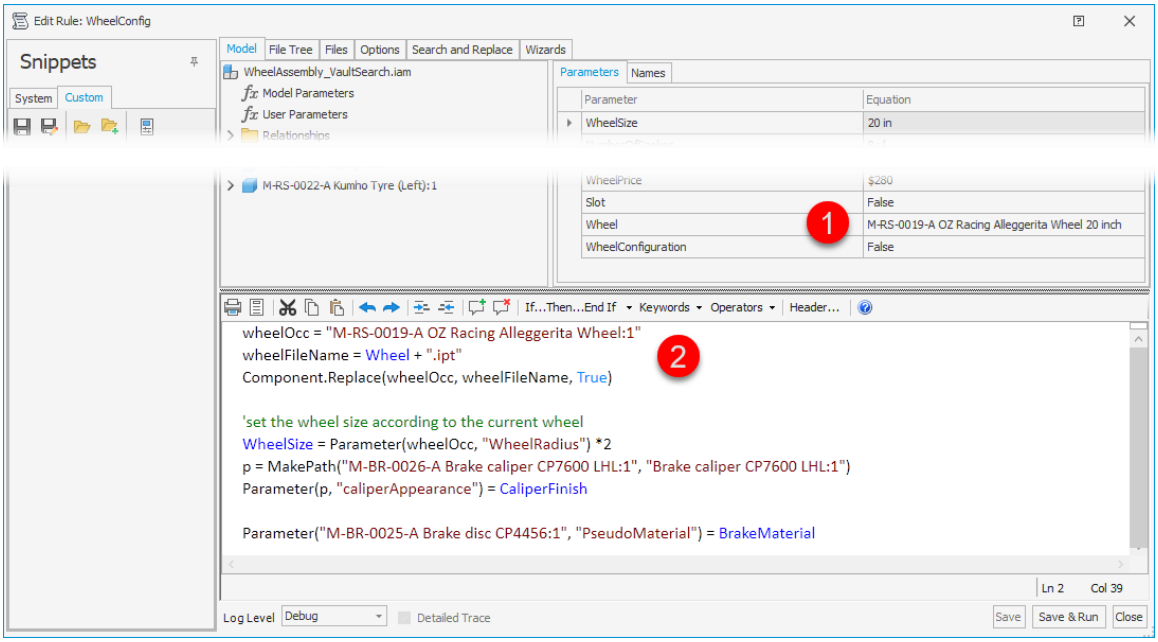


Figure 14

Save the result and create more copies of the wheel in Vault for each size and update the model accordingly. Release two of them. (Figure 15)

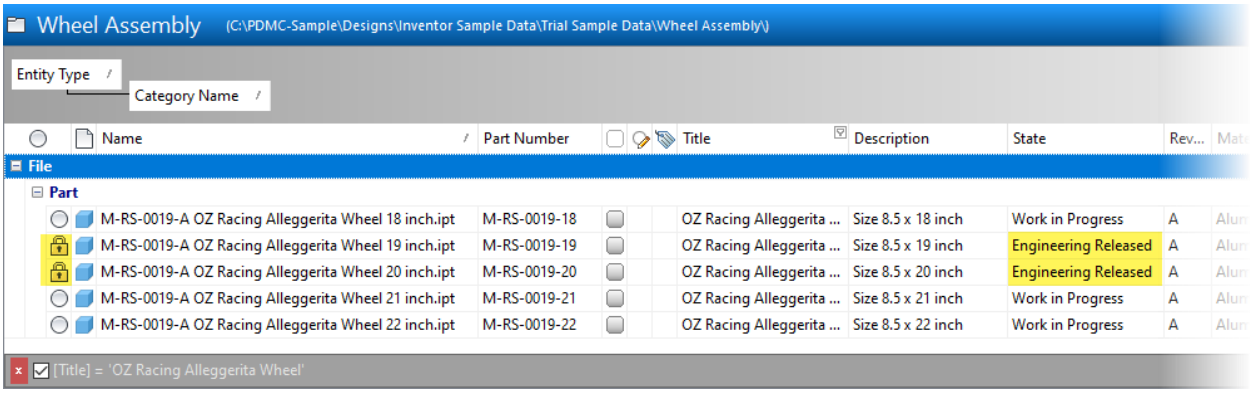


Figure 15

Create the new rule "SearchWheels" as shown in the video: https://youtu.be/Ab8KcxNIxe0

The last necessary step is not shown in the video. You need to update the Form by changing the pulldown list for the size. Bind it to the new rule. (Figure 16)

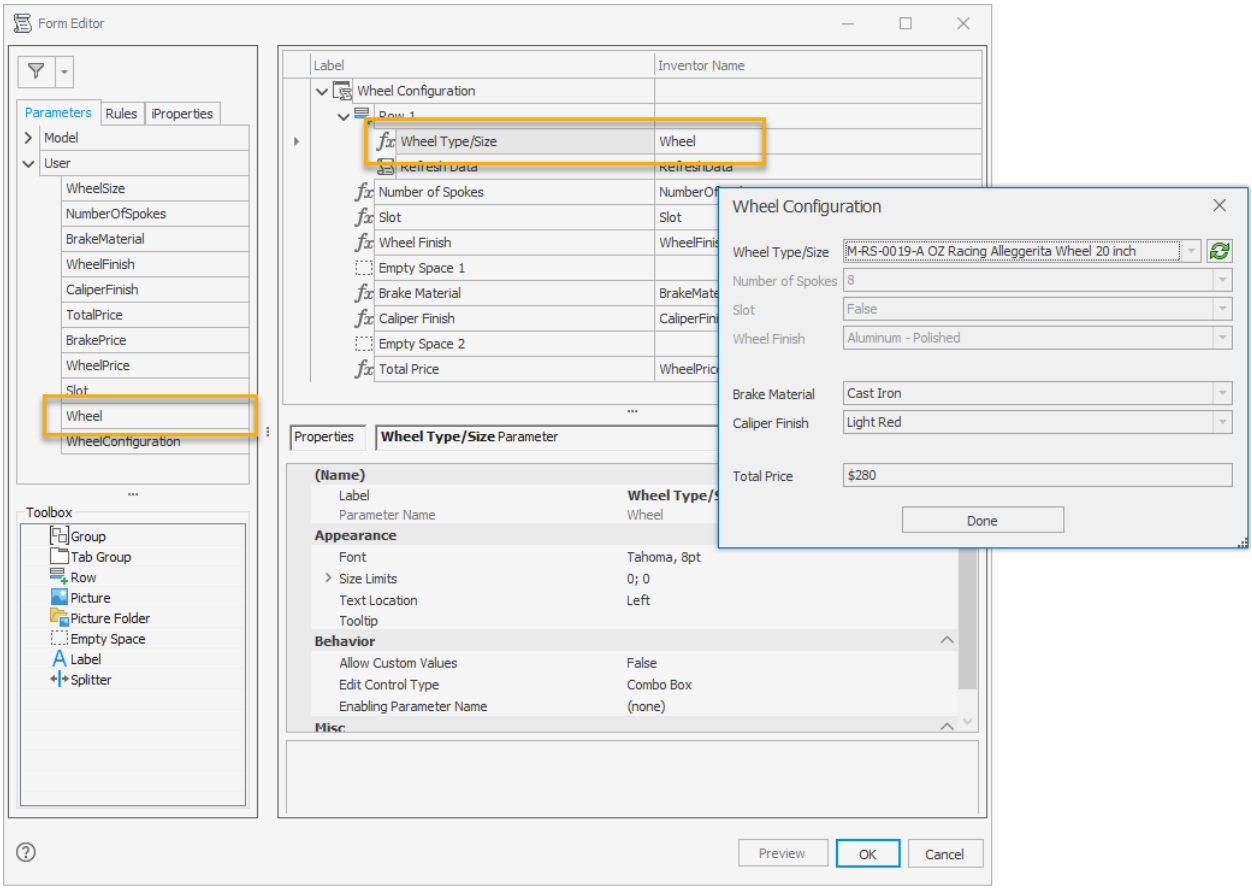


Figure 16

Additional Class Samples

Download the additional class material files package to get insights into implementing the extended configuration re-using existing files from Vault, or creating new components if needed. The package contains all additional files created by Markus Koechl and requires integrating the downloaded Trial Sample model first.

It is recommended to replicate the given file structure in your Vault; it avoids updating paths within the rules. (Figure 17)

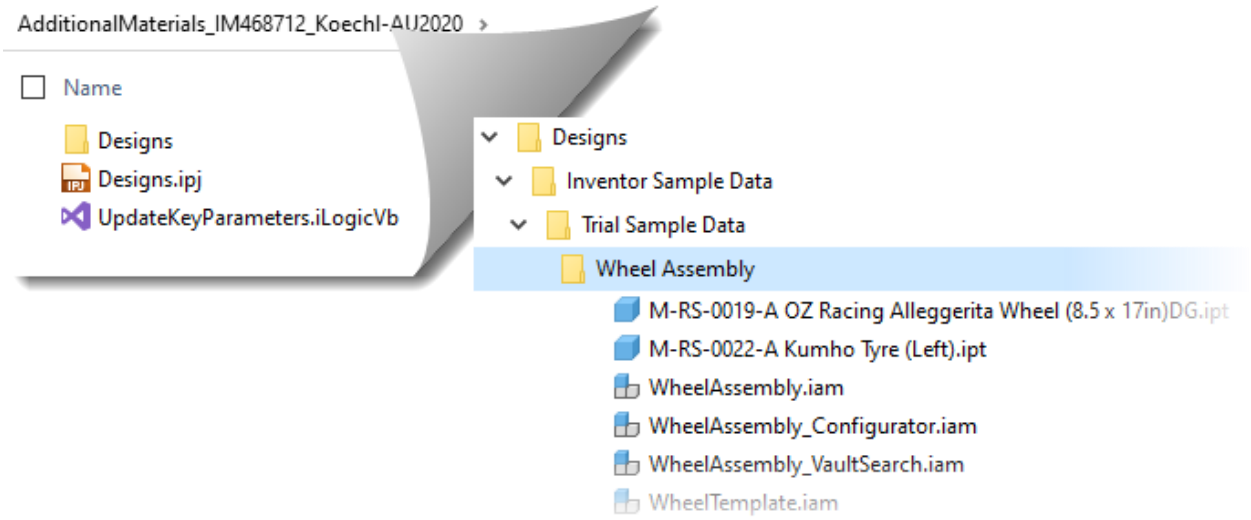


Figure 17

Vault & iLogic

Vault – iLogic Job Processing

Did you ever ask: "How can we use iLogic while running Jobs on Inventor files on Vault Job Processor?"

The answer is, you need to have a custom job that either run document-rules or external rules on the file processed. By following this section, you get access to an installable Custom Job Extension for evaluation purposes. And programmers being experts in one of the domains can access the complete source code sharing insights on solving Vault and iLogic side's automation steps.

Download and install the iLogic Job Extension Sample

This sample requires Vault Professional 2021 or newer Client installed on your computer.

Download the latest release from this open-source repository:
https://github.com/koechlm/Vault-Job-Processor---iLogic-Extension/releases/latest

Follow the web page instructions on reading and accepting the disclaimer text file before opening and using this sample.

Start the Setup.exe (or the corresponding MSI-Installable). The wizard guides you through the installation. (Figure 18)

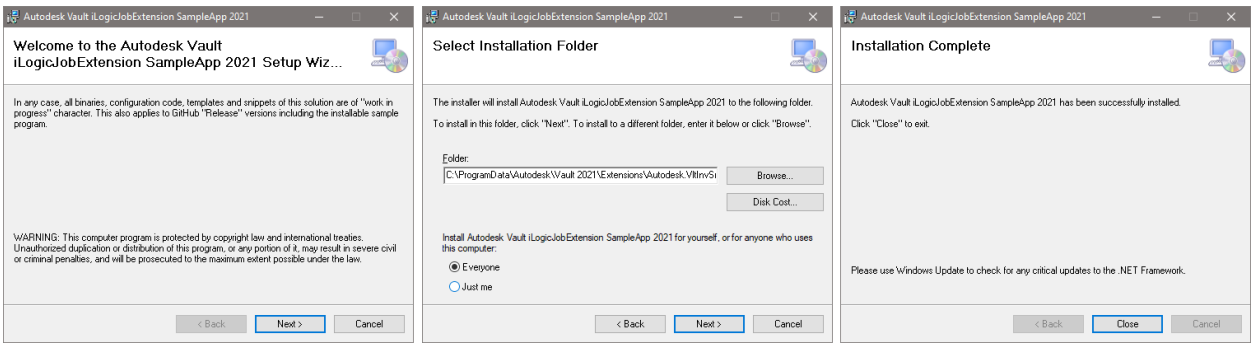


Figure 18

Note – The installer creates an environment variable (Figure 19) defining a path to temporary save rule files for extended debugging. Ensure that you have read/write access to this path or adjust the path to an accessible location. The default setting of this location is read-only.

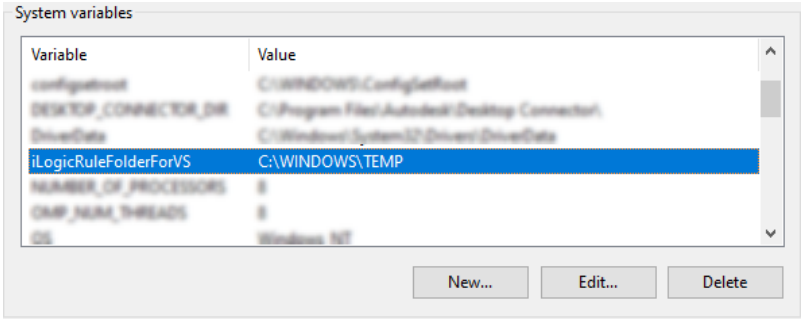


Figure 19

Start your Vault Professional Explorer Client; the extension added a menu command to the Tools and the Actions menu. (Figure 19)

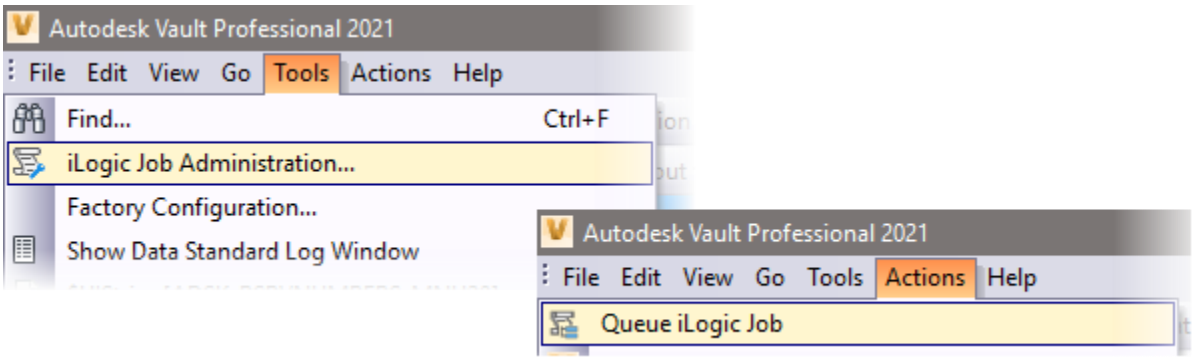


Figure 20

Start the Job Processor and validate the installation reviewing the list of registered jobs. Successful installation of the job extension registered the sample job with the Job Type Name "Autodesk.VltInvSrv.iLogicSampleJob," as shown in Figure 21.

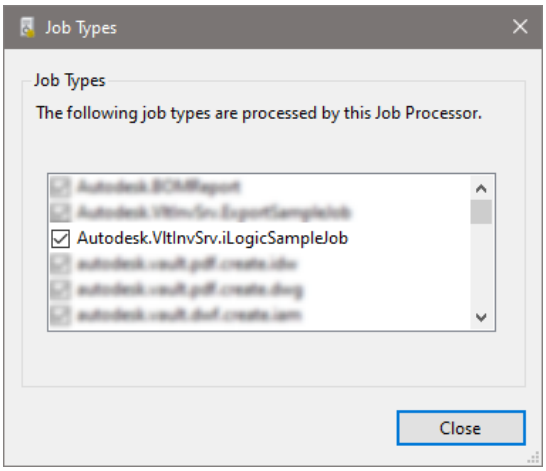


Figure 21

In case you liked to remove or repair-install the extension run the installer again (Figure 22). Both actions can be started from the Programs and Features menu also.

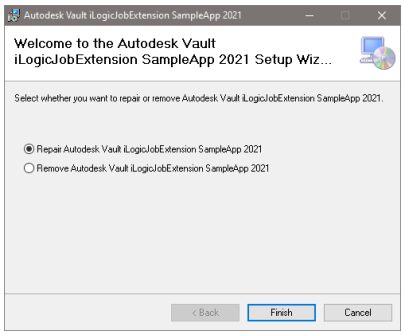


Figure 22

Getting started in using the iLogic Job Extension

The setup copied two sample iLogic Job rules (Figure 23, **(1)**) to the program's installation folder.

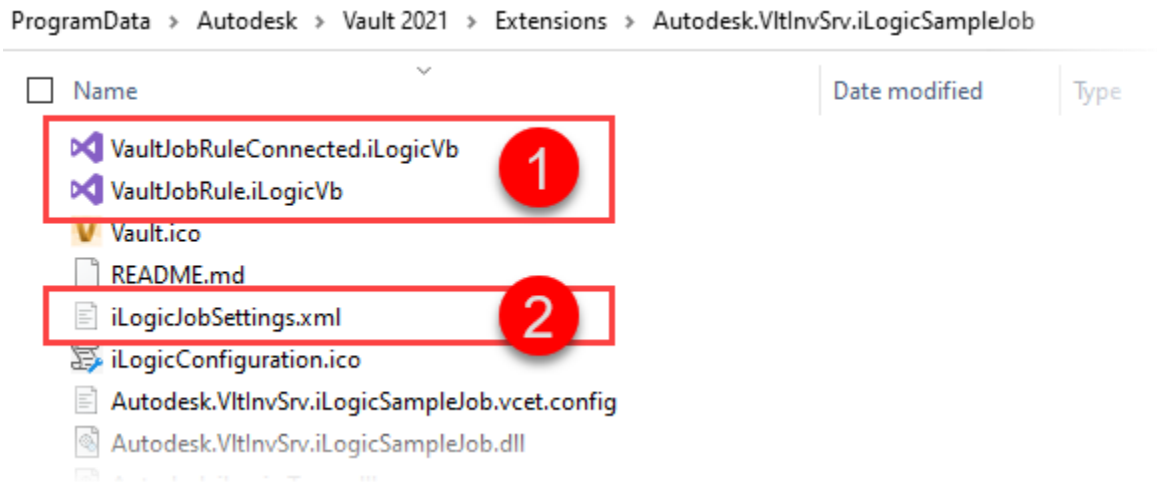


Figure 23

Adding these to Vault is the fastest way to run your first iLogic rule job. The class' sample environment integrated both files into the Design Data, iLogic structure. (Figure 24)

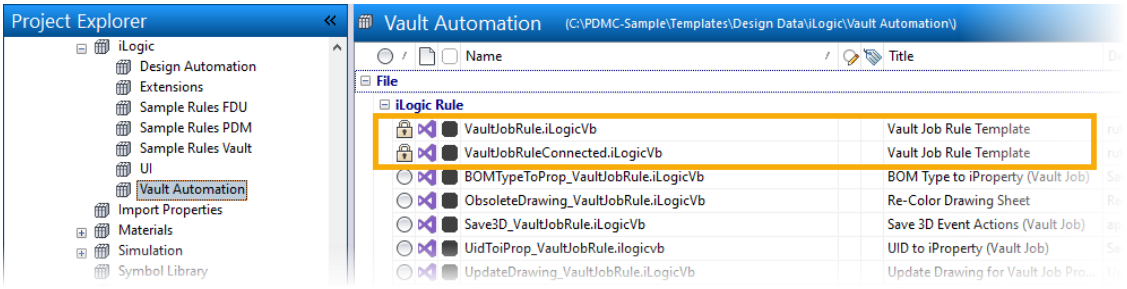


Figure 24

Open the iLogic Job Administration Dialog to get your environment ready, running iLogic rules as Jobs.

Configure all settings as you usually do in Inventor Advanced iLogic Configuration dialog. (Figure 25)

Note – Rules for Vault Jobs download to the local working folder on execution. Therefore, Vault Job rules' external rule directory must be a subfolder within the Vault working folder.

Logging: Set the Log Level "Trace". The log output of this iLogic Job Extension is written to text files. Configure the target location by selecting an existing folder.

Save your settings to Vault.

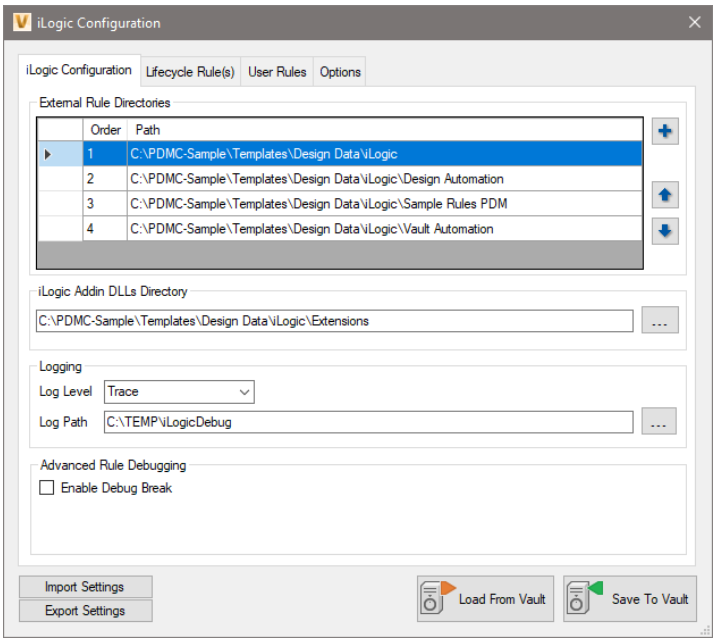


Figure 25

Import/Export Settings reads or writes all settings to the file iLogicJobSettings.xml (Figure 23, **(2)**); this is useful for transferring settings from one Vault to another. The iLogic Job extension reads the configuration, particularly for each job from Vault; therefore, changes in the configuration do not require to restart the Job Processor.

Proceed to the tab Lifecycle Rule(s) to configure the Primary Rule file; this rule runs if a lifecycle transition triggered the job "Autodesk.VltInvSrv.iLogicSampleJob". (Figure 26)

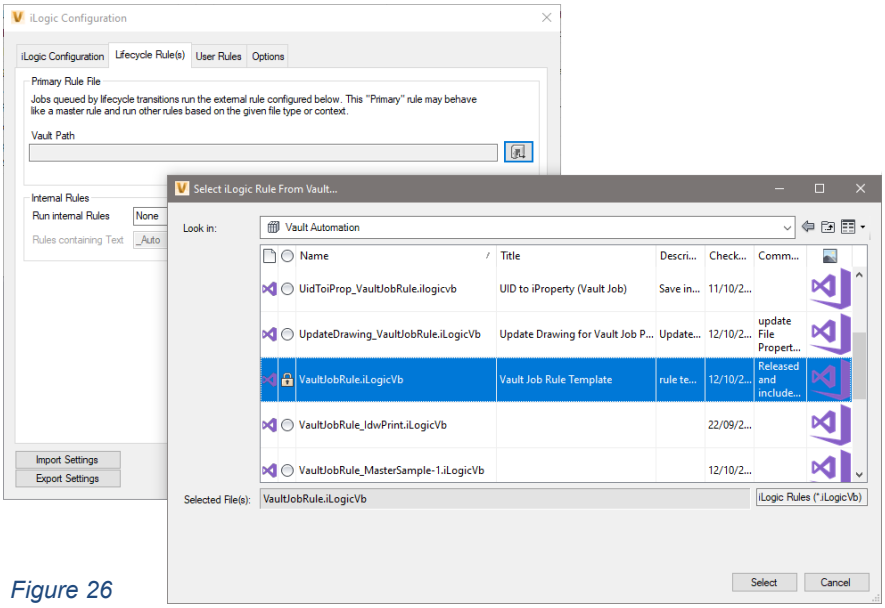


Figure 26

Save your settings to Vault and exit the dialog.

Open the Vault settings and add the job type name to a lifecycle state transition. (Figure 27). We recommend creating a copy of an existing lifecycle before.

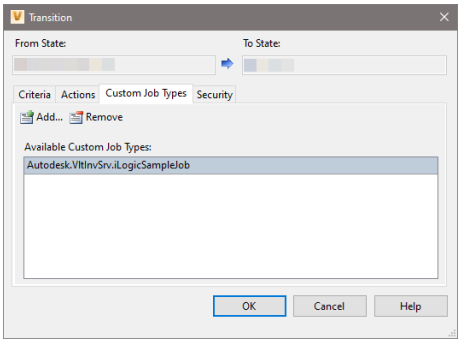


Figure 27

Everything is set to run your first iLogic Rule Job. Select any Inventor file assigned to the lifecycle and move this file through the state transition. Review the Job Queue (Figure 28); you should see the job type Autodesk.VltInvSrv.iLogicSampleJob pending to execute the selected file.

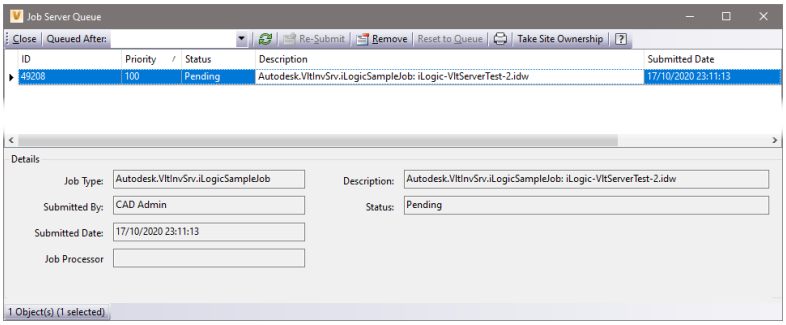


Figure 28

Once the job is finished, review the processed file. A new file version with the comment "Created by Custom Job executing iLogic: External Rule: VaultJobRule.iLogicVb" should be created. (Figure 29)

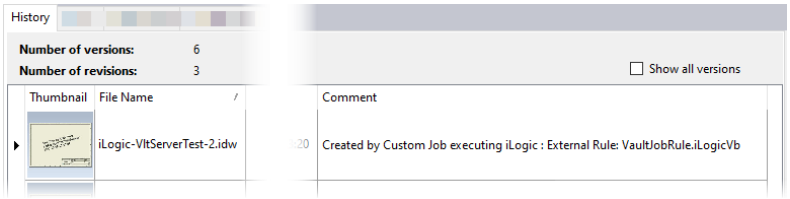


Figure 29

Navigate to the Log File directory as configured in the Vault iLogic Configuration (Figure 25).

The job created a file with the name <Job ID>_<SelectedFile>_iLogicSampleJob.log; open it and review the content. The rule copied the Vault iLogic configuration settings and all Vault property/value pairs to the log file. (Figure 30)

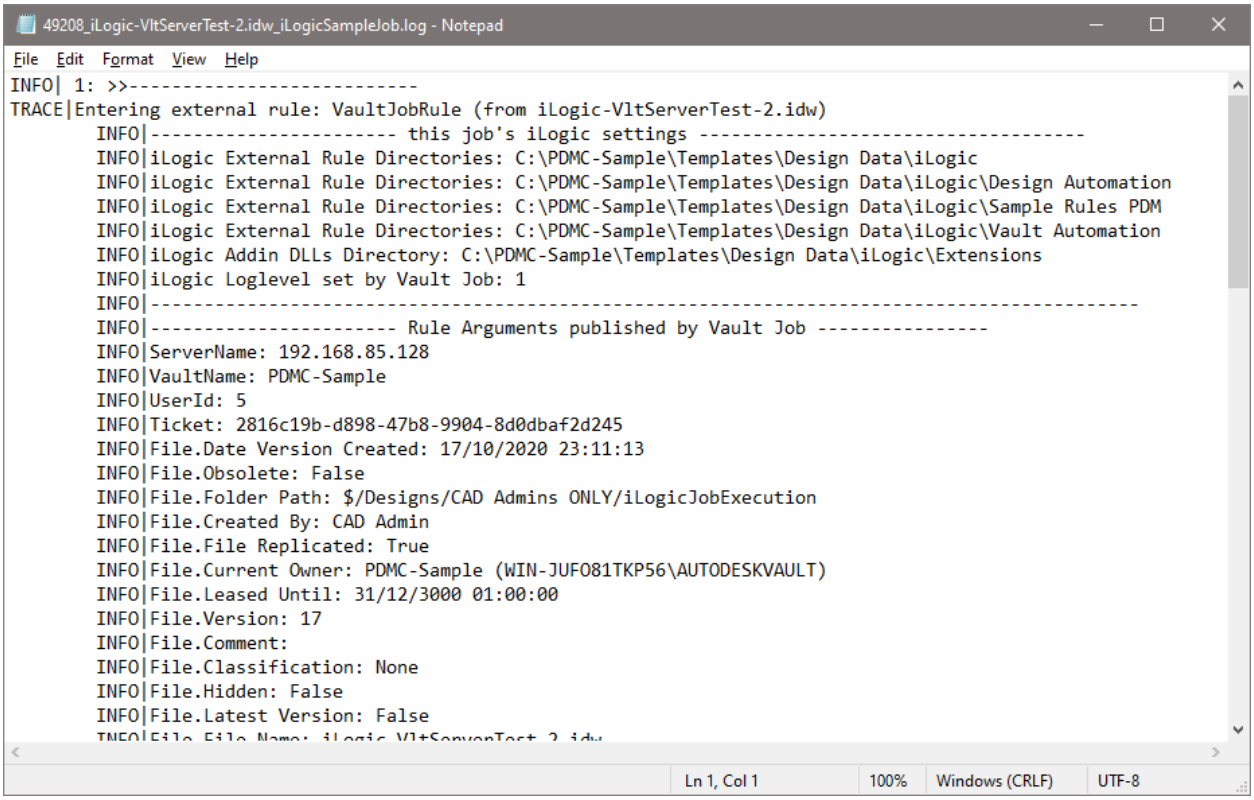


Figure 30

Configure the iLogic Job Extension for manual Job submissions.

The Vault iLogic Job Extension Sample allows run rules on select files. To assign a rule to a file or set of files, the Tab "User Jobs" needs to be configured. Open the Vault iLogic Job Administration and add both rules (Figure 24) to the available User Jobs (Figure 31). Enable the option "Create New File Version" if the iLogic rule changes should be saved as a new Vault file version.

Note - Uncheck "Create New File Version" if the rule's actions do not require saving the file, e.g., exporting CAD BOMs to other systems or printing drawings.

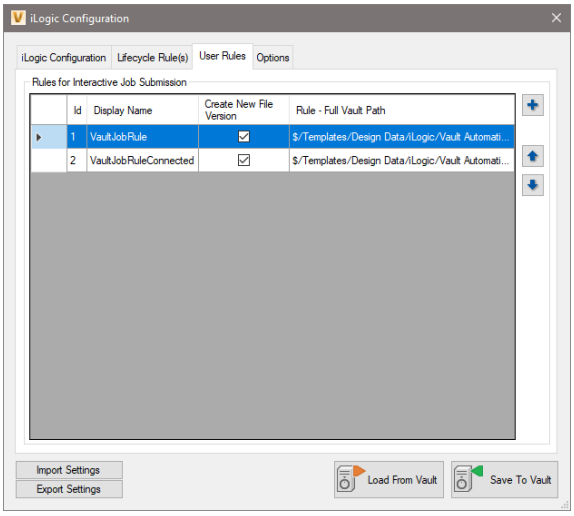


Figure 31

Select one or more Inventor files, open the Dialog "Queue iLogic Job", chose one of the sample rules, and submit the set to the queue. (Figure 32)

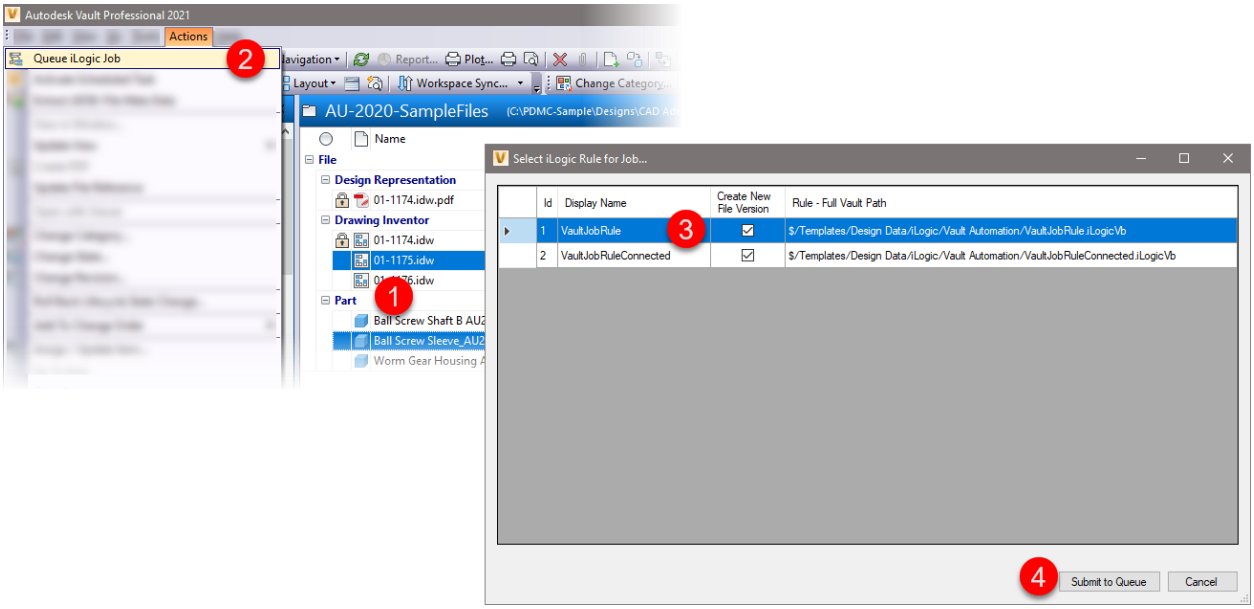


Figure 32

Open the Job Queue (Figure 33) and note that the Job Description indicates the manual submission; the job priority is set to 1. These Manually submitted jobs run before the jobs submitted by lifecycle transitions (Priority=100).

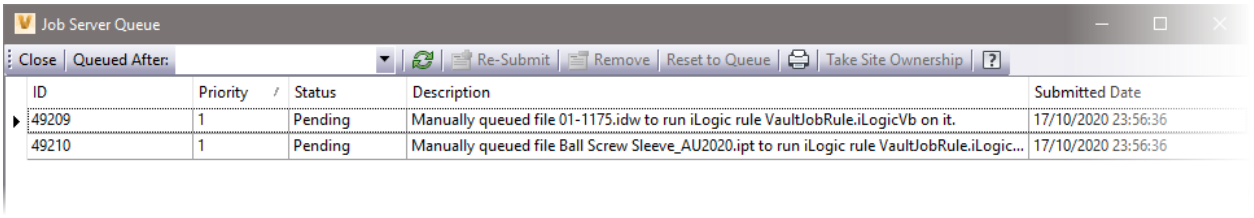


Figure 33

Manual submitted jobs creating new file iteration comment the action as we saw before in Figure 29.

Writing iLogic Job Rules

Having the iLogic Job Sample Extension installed and configured, you can start writing your first own iLogic job rule; as an initial implementation, we recommend to follow the video https://youtu.be/-7Yz-C-aDEE to write a rule optimized for Vault iLogic Job Processing.

The essential differences in writing rules for the Vault Job Processor are highlighted in the rule file's comparison. (Figure 34)

C:\PDMC-Sample\Templates\Design Data\iLogic\Design Automation\ObsoleteDrawing.iLogic

```
1
2  'access the active document
3  Dim mDoc As Document = ThisDoc.Document
4  If mDoc.DocumentType = Inventor.DocumentTypeEnum.kDrawing[
5      Dim mDrwDoc As DrawingDocument = mDoc
6      'set the sheet's color
7      Dim mColor As Inventor.Color = ThisServer.TransientObjects.Cr
8      mDrwDoc.SheetSettings.SheetColor = (mColor)
9  Else
10     Logger.Warn("The rule exited because the expected document
11     Exit Sub
12 End If
```

C:\PDMC-Sample\Templates\Design Data\iLogic\Vault Automation\ObsoleteDrawing_VaultJobRule.iLogicVb

```
1
2  'return unhandled exceptions to Vault Job Processor
3  Dim ex As New Exception
4
5  'leverage Vault Properties
6  If RuleArguments("File.Obsolete") = "True" Then
7      Try
8          'access the active document
9          Dim mDoc As Document = ThisDoc.Document
10         If mDoc.DocumentType = Inventor.DocumentTypeEnum.kDrawingDocumentObje
11             Dim mDrwDoc As DrawingDocument = mDoc
12             'set the sheet's color
13             Dim mColor As Inventor.Color = ThisServer.TransientObjects.CreateColor(255
14             mDrwDoc.SheetSettings.SheetColor = (mColor)
15         Else
16             Logger.Warn("The rule exited because the expected document type Drawing
17             Exit Sub
18         End If
19     Catch
20         Logger.Error("Unhandled exception in rule ObsoleteDrawgin_VaultJobRule.")
21         Throw ex
22     End Try
23 Else
24     Logger.Error("File state is not obsolete, undo check-out.")
25     Throw ex
26 End If
27
```

Figure 34

Review the green-colored additions in the job rule.

The custom Job handles return values of rules and feedbacks to the Vault Job Queue. Unlike as in Inventor Application, we cannot get error and exception dialogs of the iLogic Add-in. The rule actively needs to handle any error or unforeseen exception by throwing a.NET exception instance.

The optimized rule wrapped an IF...Else...End (Figure 34, Line 6..26) statement around the core actions. The if condition leverages the sample job's rule arguments: all property values are available and accessible using the syntax of RuleArguments("File.<Property Name>"). In our rule, the system property "Obsolete" (Figure 34, Line 6) determines to continue or to throw an exception. Note – The file is already checked out while the rule runs. By throwing an exception, we feedback the rule as failed to the parent job. The iLogic Job Sample Extension reverts files to the original state by applying an Undo Check-out to the file.

The same principle is used to handle any unforeseen error in the core actions of the rule. The Try...Catch...End Try statement is the easiest way to achieve it, but we recommend to actively handle as many conditions as possible as a best practice.

The job queue does not list exception details. We can use the log file for all details of the iLogic rule runtime.

Closing Remarks

The AU 2020 class IM470378 iLogic für Vault Job Processor covers "Writing iLogic Job Rules", adding more details. Although this class recording is in the German language, the presentation and class handout document IM470368 are written in the English language.