powerJobs Processor

coolOrange s.r.l

Jun 10, 2025

POWERJOBS PROCESSOR

CHAPTER

ONE

INSTALLATION

1.1 Requirements

As powerJobs Processor is an extension to the Vault Job Processor, the Vault system requirements defined by Autodesk leads.

Operating System: 64-bit only

- Microsoft Windows 10
- Microsoft Windows 11

Autodesk Vault Client: 2026 / 2025 / 2024 / 2023

- Vault Professional
- Vault Workgroup

Windows PowerShell: PowerShell 5.1 or higher

PowerShell 7 ***(optional)***

With Vault 2026, Autodesk has completed the transition of its applications to .NET 8.
Therefore, job scripts for the Vault 2026 Job Processor run in a self-hosted PowerShell 7.4 environment, while powerJobs Processor for Vault 2023-2025 continues to use the installed Windows PowerShell.

PowerShell 7 is backward-compatible with earlier Windows PowerShell versions, so job scripts and modules typically work without changes in all Job Processor versions.

An additional PowerShell 7 installation is therefore not strictly necessary, unless you plan to use the PowerShell 7 Console or IDEs like Visual Studio Code on your development environment, or you want to debug Vault 2026 scripts that use new features of PowerShell 7.

---

coolOrange powerVault: powerVault is installed automatically

Autodesk DWGTrueView: 2026 / 2025 / 2024 / 2023 *(optional)*

- In order to create *PDF or other formats* out of drawings.
- Supported versions:

Vault Version	DWG 2023	TrueView	DWG 2024	TrueView	DWG 2025	TrueView	DWG 2026	TrueView
2023	✅		☑		☑		☑	
2024	☑		✅		☑		☑	
2025	☑		☑		✅		☑	
2026	☑		☑		☑		✅	

☑ … Not tested but supported
✅ … Tested and supported

Autodesk Inventor: 2026 / 2025 / 2024 / 2023 *(optional)*

- In order to create *PDF or other formats* out of drawings, Inventor models and the like.
- Supported versions:

Vault Version	Inventor 2023	Inventor 2024	Inventor 2025	Inventor 2026
2023	☑	☑	☑	☑
2024	☑	✅	☑	☑
2025	☑	☑	✅	☑
2026	☑	☑	☑	✅

☑ … Not tested but supported
✅ … Tested and supported

Recommendation

We recommend to install the Vault JobProcessor and powerJobs Processor on a **dedicated workstation**.
As CAD- and potentially even other *Applications* shall run on such machine, a dedicated Job Processor environment is more appropriate from a hardware prospective.
Additional software licenses must activated on this environment, depending on the used *Applications*.

1.2 Setup

The powerJobs Processor setup is delivered as an executable and accepts the standard windows installer arguments documented here.
To accept the products EULA when starting the setup in silent mode pass the **ACCEPT_EULA=1** argument.

1.3 After the Setup

After installing powerJobs Processor in **silent mode,** *Inventor* **must be started** once, to make sure there are no problems with its installations and it is registered correctly with the operating system.

1.4 Install Locations

powerJobs Processor is installed in the following locations on your system:

- The Cmdlets will be installed to *C:\Program Files\coolOrange\Modules\powerJobs*
- All job scripts and PowerShell modules are stored in *C:\ProgramData\coolOrange\powerJobs*
- The extensions for the Job Processor and Vault Client are located in *C:\ProgramData\Autodesk\[Vault Version]\Extensions\powerJobs Processor*
- The powerJobs Processor executable and all its related files are installed to *C:\Program Files\coolOrange\powerJobs Processor*
- The shared library *powerJobs.Common.dll* is installed in the *GAC* (only for Vault 2023 to 2025) and additionally in *C:\Program Files\coolOrange\Modules\powerJobs*

Following shortcuts are added in the start menu:

- **powerJobs Processor** - Starts the JobProcessor with extended UI
- **powerJobs Processor Configuration** - Opens the location for the configuration files
- **powerJobs Processor ISE** - Opens the IDE and opens a sample job
- **powerJobs Processor Logs** - Opens the log file location
- **powerJobs Processor Information** - Opens the About dialog with product related information
- **coolOrange License Manager** - Opens the License Manager dialog for activating coolOrange products

1.5 Windows permissions

The current Windows user needs the following rights for proper job execution, settings editing, and synchronization of distributed scripts and modules, as well as for product activation via the License Manager dialog.
These rights are automatically installed for *"Everyone"*:

Path	Required Rights
C:\ProgramData\coolOrange\	**Read**, **Write** data
C:\ProgramData\coolOrange\powerJobs\Jobs\	**List folder / read data**, **write data** and **Read extended attributes**
C:\ProgramData\coolOrange\powerJobs\Jobs\Modules	**Execute** - required to load assemblies of *custom-built applications*
C:\ProgramData\Autodesk\[Vault Version]\Extensions\powerJobs Processor\PowerJobs Processor.vcet.config	**Write**

1.6 Environment Variables

powerJobs Processor sets three environment variables when it's installed. They contain the paths to the **jobs** and **modules** folder and the **job handler extension's directory**

The three environment variables are:

- POWERJOBS_JOBSDIR
- POWERJOBS_MODULESDIR
- POWERJOBS_PROCESSORDIR

1.7 Updates

To install a newer version of powerJobs Processor just execute the setup file of the new version. This will automatically update the files in the existing installation.

1.8 Uninstall

In case you want to remove powerJobs Processor from your computer you can:

- Execute the setup file again. This will give you the option to repair or remove powerJobs Processor. Click on "Remove" to uninstall the program.
- Go to "Control Panel - Programs and Features", find "coolOrange powerJobs Processor for Vault" and run "Uninstall".

CHAPTER

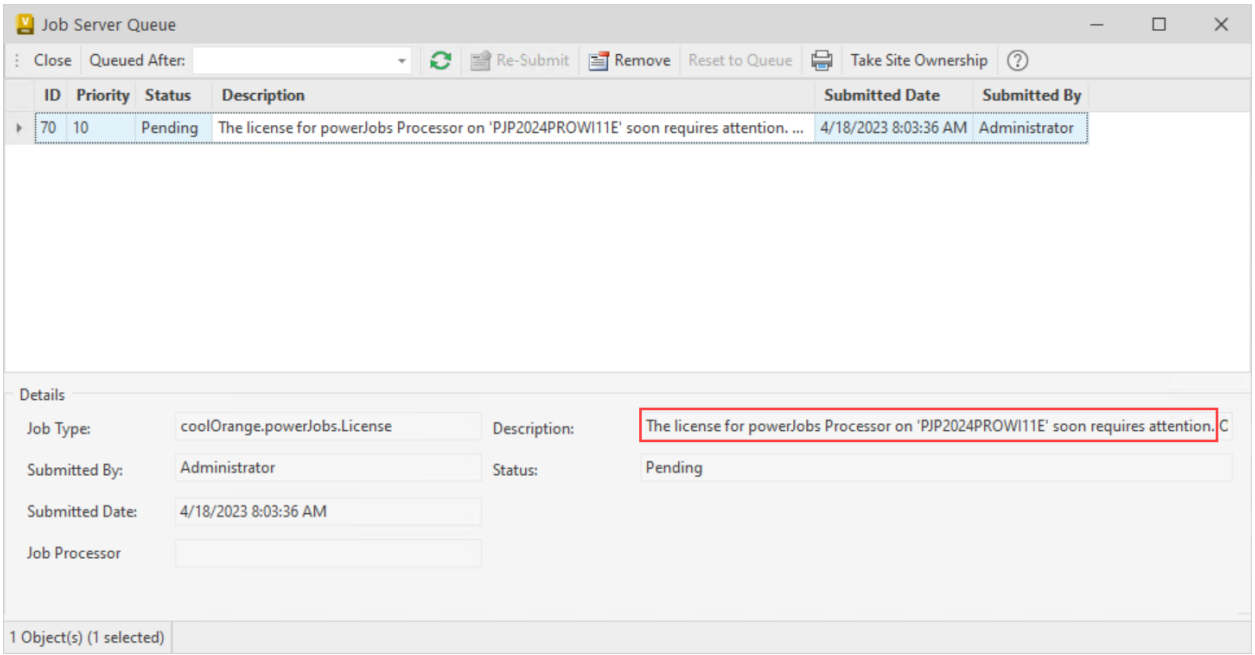
TWO

ACTIVATION AND TRIAL LIMITATIONS

2.1 Trial limitations

There is no difference in functionality between the trial version and the fully licensed product.
After the installation the product is available as a trial version for **30 days**.

During the trial period powerJobs Processor will place a *License Job* to the Job Queue once a powerJobs job is executed.
This way the Vault administrator gets notified in advance about the product that is soon expiring on the Job Processor.



These jobs will not be executed by any Job Processor and needs to be removed manually.

2.2 Activation

To keep using powerJobs Processor after the trial period, the product needs to be activated on every Job Processor where it is in use.

Follow these steps to activate new installations of powerJobs Processor on **Job Processor machines** together with all other products included with the same subscription.
After a successful activation, all products included in the subscription can be used on the Job Processor until the licenses expire.

Note: When the computer has an Internet connection no further activation steps are required even when the license is renewed or after purchasing additional products licenses.

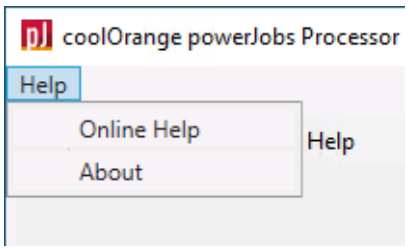
More details on product activations for additional Job Processors machines can be found here.

2.2.1 Serial Number Activation

Similar to previous versions of powerJobs Processor an activation using the serial number is also possible.
This activation method can be used if only a single product should be activated or to activate older product versions (v22 - v24).
Further details can be found in the second section on activations via serial numbers "For products with version 20.x - 24.x".

To activate powerJobs Processor via serial number, the powerJobs Processor specific License Information dialog required. It can be opened by:

- The Start Menu: Navigate to "*All Apps* → *coolOrange* → *License Manager*" and search for "*powerJobs Processor*".
  Then click on the "*Trial*" link (or on the already displayed serial number).
- Or in the *powerJobs Processor* navigate to the '*Help*' menu, click on '*About*' and then use the '*Licensing Information*' link:



- Or register the serial number directly via the command line:

```
"C:\Program Files\coolOrange\powerJobs Processor\License.exe" --Serialnumber="XXXXX-
↪XXXXX-XXXXX-XXXXX-XXXXX" --StandAlone
```

2.2.2 Offline Activation

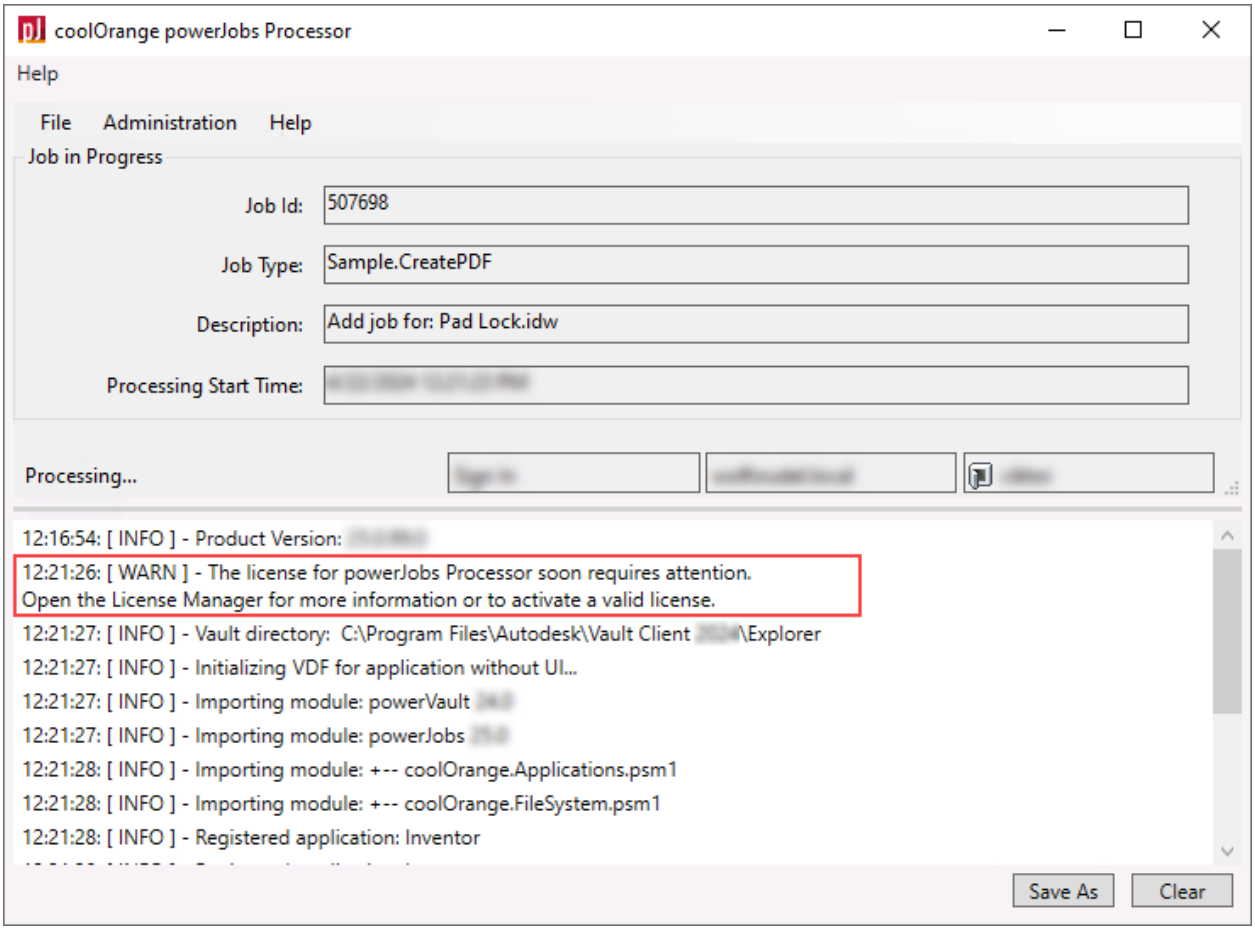
For environments where no internet connection is available a license file can be used to activate the product.
The License file can be downloaded from any device using this link: Offline Activation of powerJobs Processor.
To generate the license file, the serial number for the license as well as the *Machine Code* (for the computer the license should be activated on), which is displayed in the License Information dialog, are required.

Note: For offline activations the activation steps have to be repeated when the license changes (e.g. when the license is renewed). More information can be found here.

2.3 Troubleshooting License Problems

To check on the Job Processor whether the products of your subscription are activated, open the License Manager via the Start Menu by navigating to "*All Apps → coolOrange → License Manager*".
To see more details of the powerJobs Processor license, open the License Information dialog by clicking the link in the *License Manager*.

The last 30 days before this license expires, a *License Job* is added to the Job Queue.
This way, the Vault administrator can recognise in advance that his coolOrange subscription needs to be renewed.



License expired

If the powerJobs Processor license expires, all powerJobs Jobs will fail with a license error and a license job will be queued to inform about the outstanding renewal.

CHAPTER

THREE

GETTING STARTED

3.1 Overview

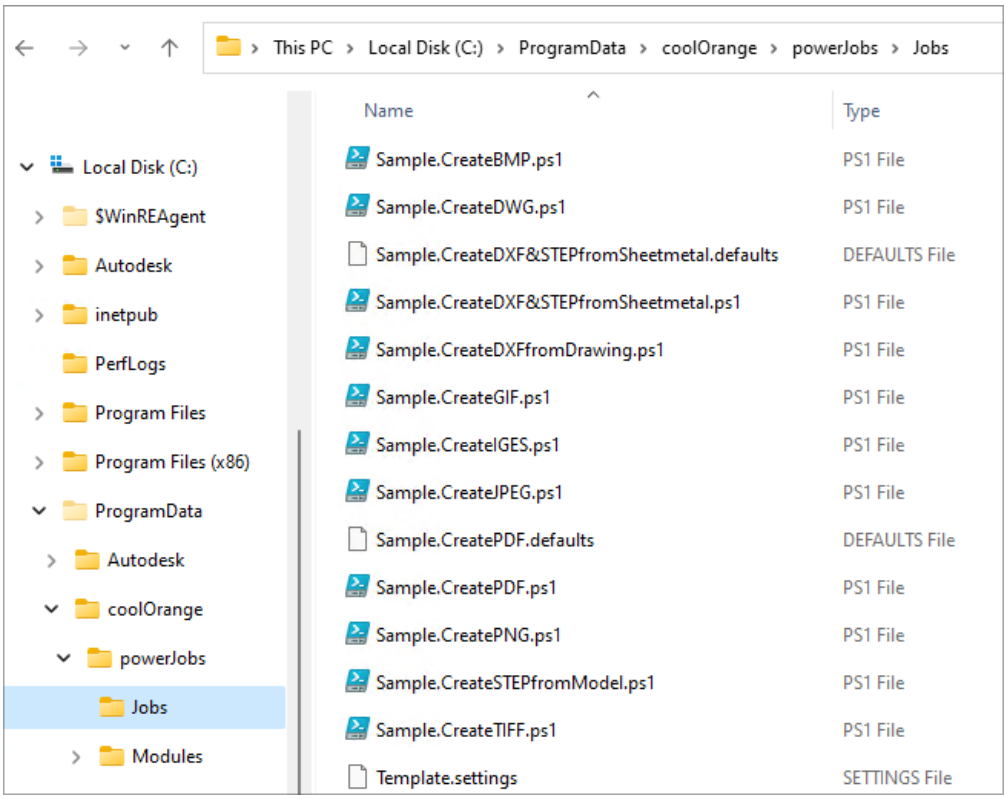
powerJobs Processor is an extension to the Vault JobProcessor. This document will guide you through the first steps after the successful installation of powerJobs Processor on a Vault client machine which acts as a Job Server.

> **Warning:** Even though powerJobs Processor can run on every machine with a Vault client, we recommend that powerJobs Processor is installed on a **dedicated** machine whose purpose is to act as a Job Processor.
> On the other hand powerJobs Client needs to be installed on each Vault client machine where it's behavior is desired.

3.2 Example how to use powerJobs

3.2.1 Where do you find the jobs?

In the folder *C:\ProgramData\coolOrange\powerJobs\Jobs* there are all the sample jobs for the powerJobs Processor.



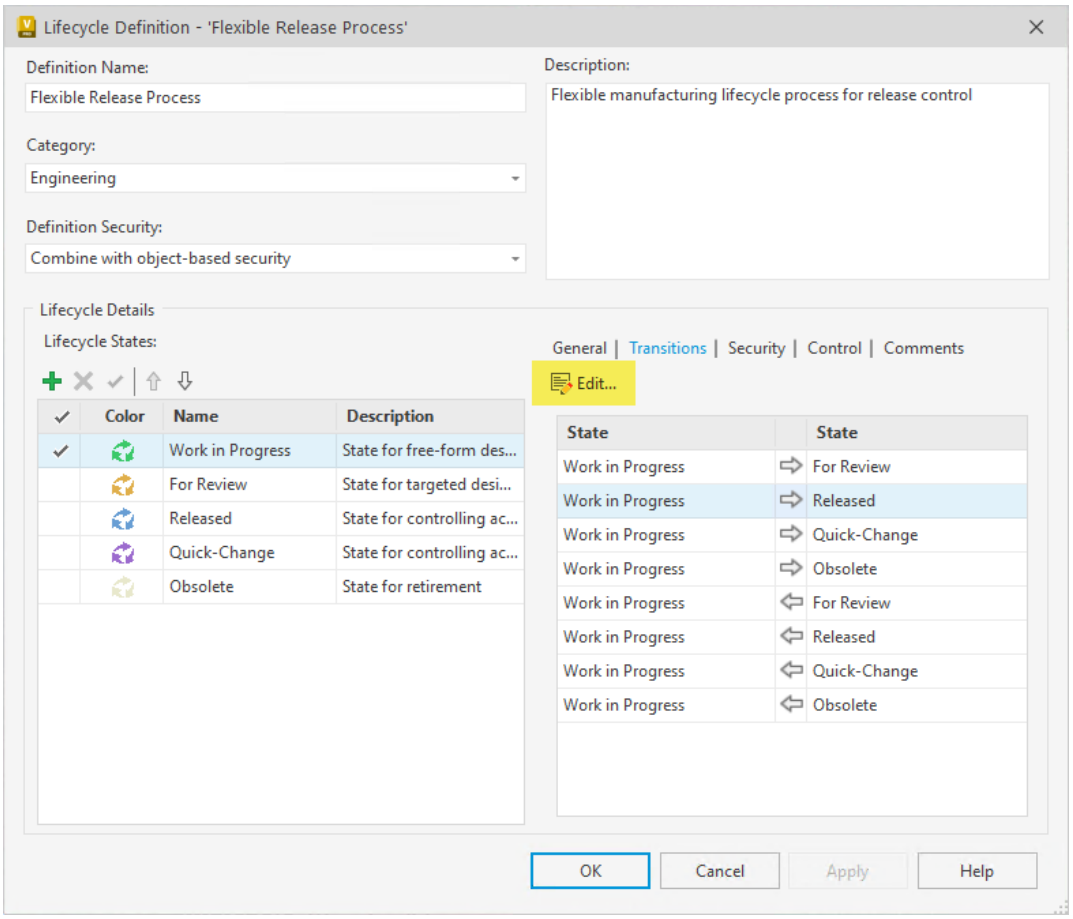
3.2.2 Create your own Job to create PDFs

Rename the job *Sample.CreatePDF.ps1* to *Custom.CreatePDF.ps1* so your PDF job can be identified as your own job and will not be overwritten by future updates. You can modify this copied job as you please.

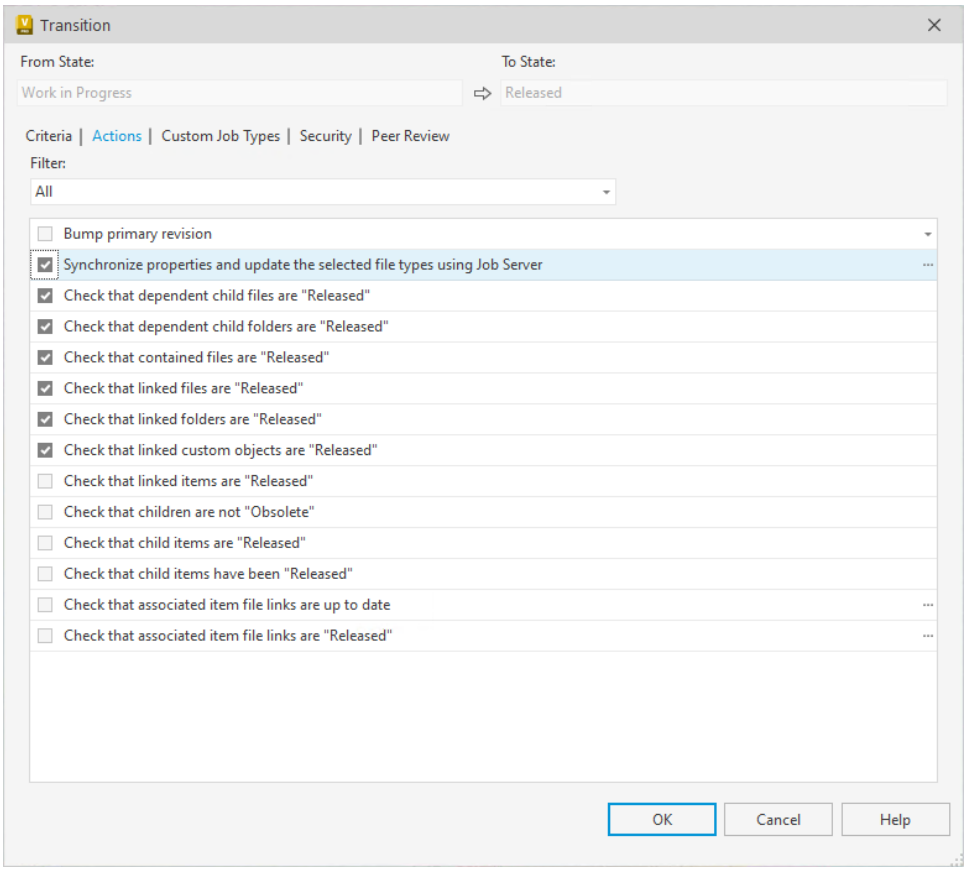
3.2.3 How to embed the Job in a Status Change

If, for example, you would like to create a PDF of drawing files (IDW and DWG) when the drawings are released in Vault, you need to do the following:

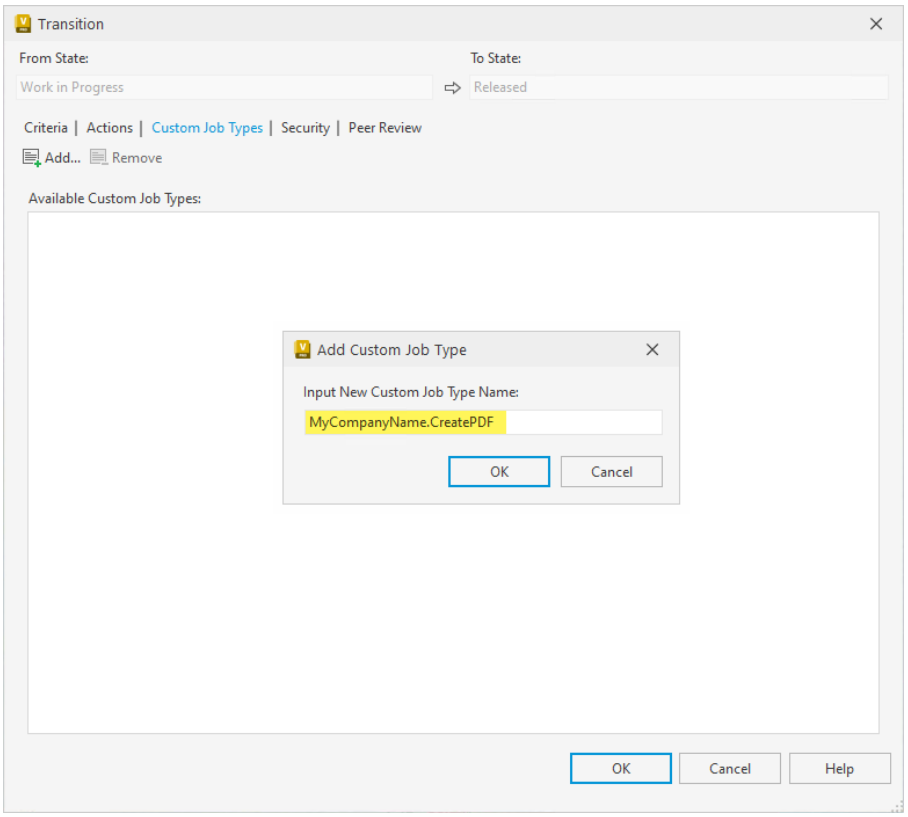
1. Open Vault Professional and edit the Lifecycle you use for your IDW and DWG files. In the “Transitions” tab, select and edit the “Work in Progress to Released” transition



2. In the tab “Actions” activate the option to “Synchronize Properties using Job Server” to be sure all properties will be up to date into the PDF file that powerJobs will create.



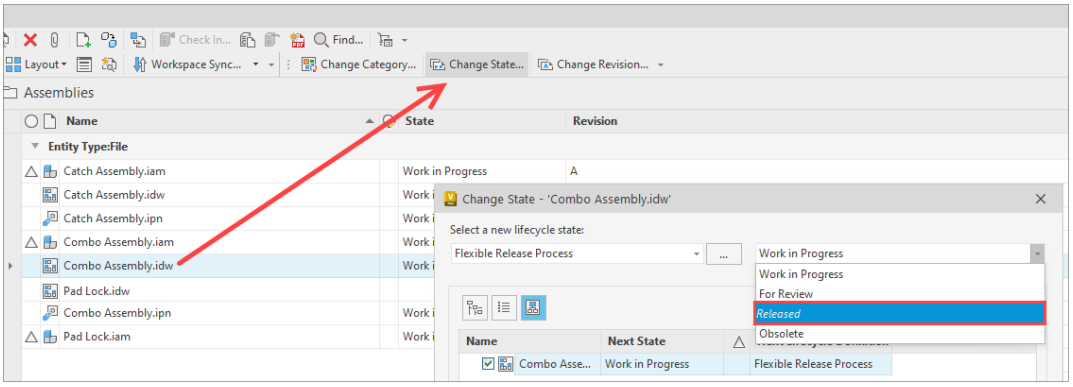
3. In the tab “Custom Job Types” Add a custom job type and write the name of the MyCompany-Name.CreatePDF.ps1 file without the extension. Confirm with “OK” button.



Your Vault is now configured to create a PDF file from IDW and DWG files that transit from Work in Progress to Released.

3.3 Use the Job on a Lifecycle Change

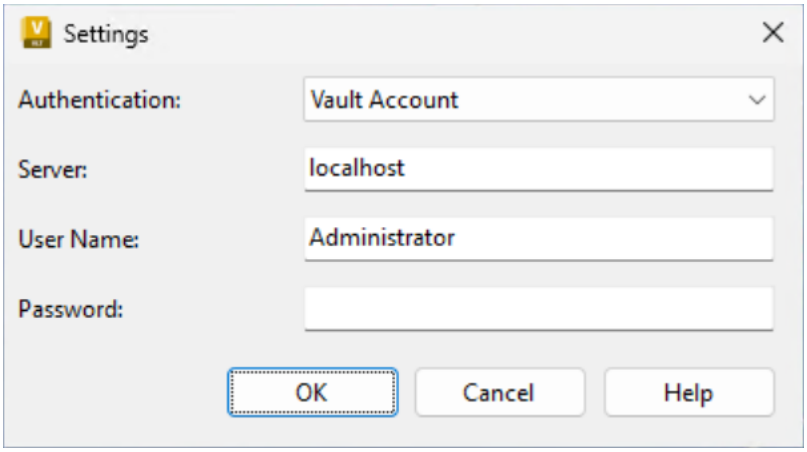
1. Select an Inventor drawing (IDW or DWG) in Vault and select command “Change State”.
2. In the Change State dialog select the “Released” state -the state you have added the job for the transition - and press the OK button:



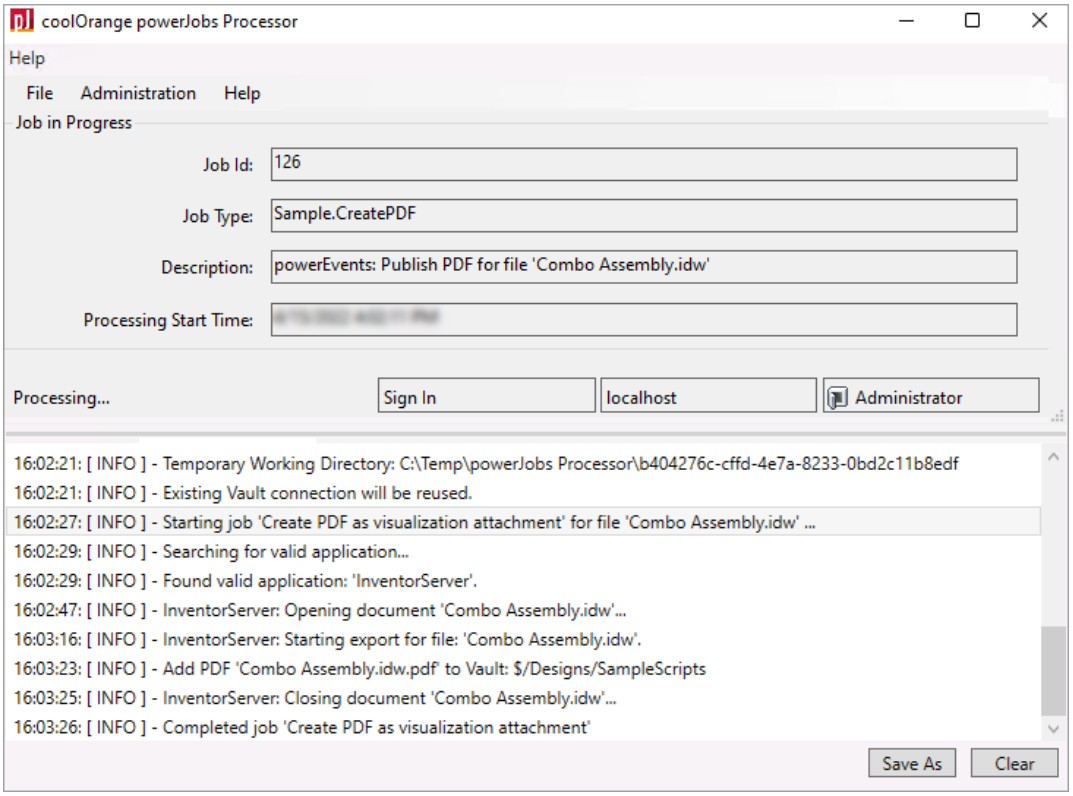
3. In case you have the powerJobs Client installed you will get a Windows notification after a few seconds that the job was added to the Job Queue.

4. Select the command “Job Queue …” from the Tools menu to open the Job Queue.
5. In the Job Server Queue dialog, you will now find 2 jobs in a pending state:
   - One for synchronizing the properties
   - One to publish the PDF for the drawing
6. On the Job Server machine start powerJobs Processor from the shortcut on your desktop

7. Log in to your Vault:



8. After powerJobs Processor started you can see how the jobs are processed:

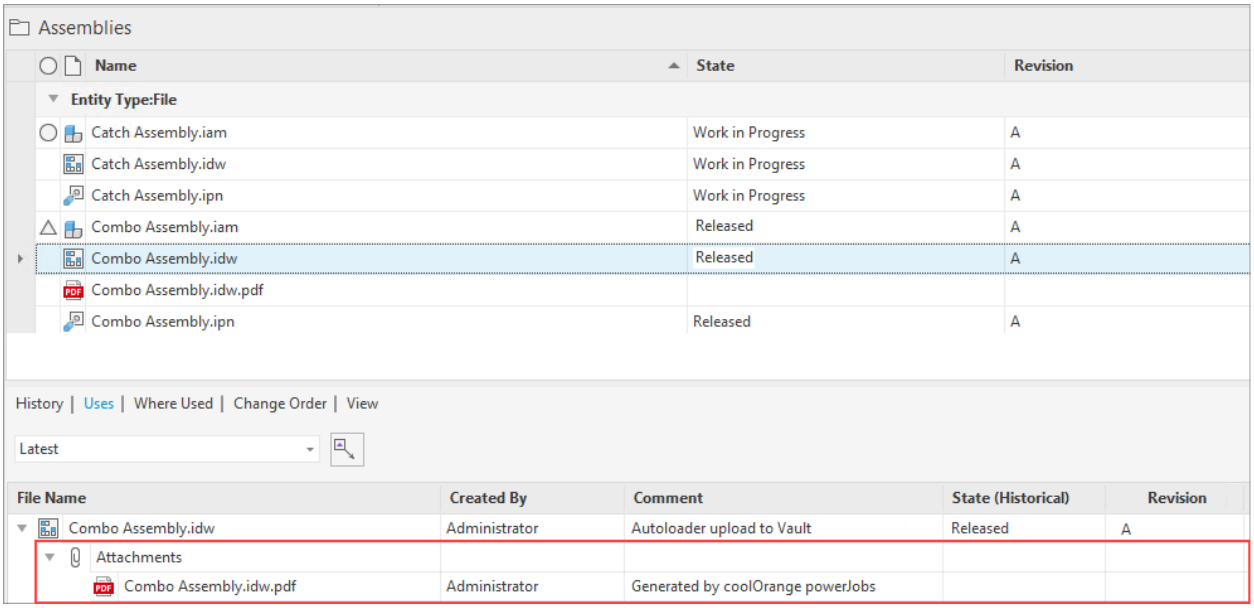


Warning: The Job Processor has a default interval of 10 min to check the job queue for new jobs.
To avoid waiting 10 min to test a job, it is possible to click File → Pause and then File → Resume in the dialog.
All queued jobs should now be processed.

Note: If you have installed powerJobs Client you will get notified if the job runs into an error. So, you will be aware if a job fails.

3.4 Result

Get back to your Vault client and refresh the UI. Beside your drawing you will find a PDF with the same name as the drawing but with pdf as an extension:



The “Used” tab shows that the PDF is attached to the drawing.

3.5 Do you want to know more?

3.5.1 Tutorials

- How to configure PDF creation of Inventor files
- How to add the Part Number to the PDF file name
- How to create PDFs with custom filenames

3.5.2 Product Documentation

- About *Jobs*, what they are and how to register and queue them.
- The *Sample Jobs* that come with the product.
- How custom *job scripts* are created.
- Which *File Conversion* are supported with per default.
- How custom *Applications* can be implemented to support custom file conversions.

CHAPTER

FOUR

JOB PROCESSOR

4.1 Applications

In a powerJobs Processor context, entities that support a specific set of operations e.g. opening or exporting a file, are referred to as Applications.
These Applications are used by the powerJobs Processor *Cmdlets* to perform their actions on documents.

Available Applications can be accessed in the *$Host.Applications* variable in a *Job Environment*.
The product also provides an option to *extend or implement new* and existing Applications.

4.1.1 Supported Applications

The following Applications can be used in *Jobs* out of the box:

Application	Note
 Inventor	Autodesk Inventor supports opening, manipulating and converting 3D mechanical designs, 2D drawings (Inventor and AutoCAD) as well as presentation files.A software license is required and a Single-User or Multi-User **License must be activated on the Job Processor** in order to prevent license issues
 InventorServer	Inventor Server is a headless version of Autodesk Inventor and is noticeably faster in starting and processing jobs, but there are no GUI features available.No additional license must be purchased for the Job Processor environment and no full Inventor installation is required.
 DWG TrueView	DWG TrueView is a file viewer for 2D drawings (AutoCAD and Inventor), which in contrast to AutoCAD, only supports conversions to PDF and DWF formats.No additional license is required and no full version of AutoCAD must be installed.

Note: Remote control of AutoCAD is currently not available out of the box, as exports of AutoCAD drawings are already fully supported through the previously listed applications.

For a detailed overview of supported conversions per application, visit the *File Conversion* page.

Registration

These default applications are registered in the *coolOrange.Applications.psm1* module with the following lines:

```
Register-Application ([powerJobs.Application.Inventor.Application])
Register-Application ([powerJobs.Application.Inventor.Server.Application])
Register-Application ([powerJobs.Application.TrueView.Application])
```

Their registration order may affect which application is used from the *Open-Document* cmdlet to open files, if it is called without *-Application* parameter.

The `Register-Application` function expects an application's Type to be passed and registers this application globally via its `Name`.
Since the passed Type must implement the *IApplication* interface, this function can also be used to register custom applications.

4.1.2 Custom Applications

Custom applications can be implemented to extend the capabilities of all *document related cmdlets*, for example when opening or exporting specific file formats is required.

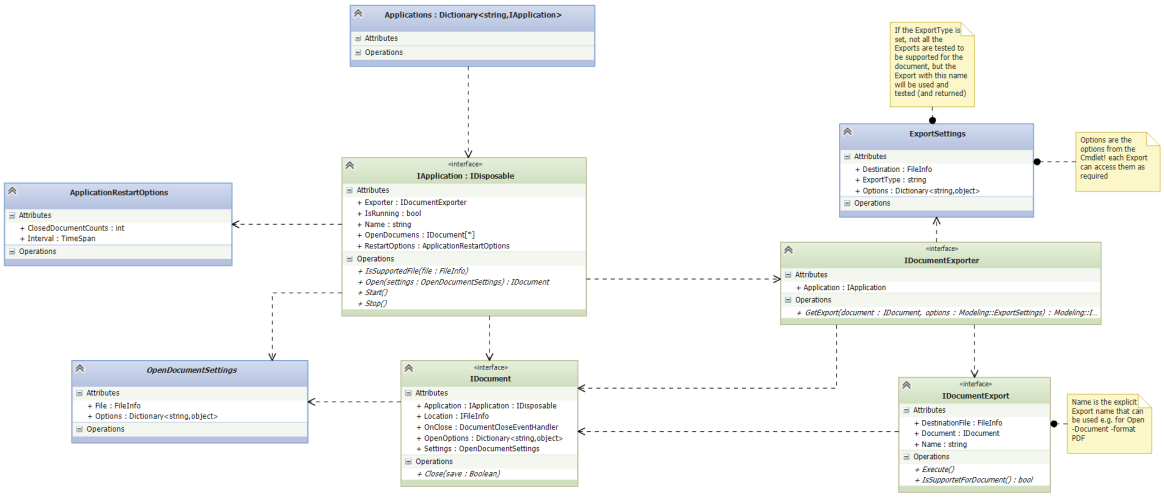
To implement the necessary .NET interfaces, you first need to *install* powerJobs Processor on your development machine.
Your project must target:

- **.NET Framework 4.7** for Vault Job Processor 2023-2025
- **.NET 8** for Vault Job Processor 2026

For compatibility across all versions, we recommend targeting **.NET Standard 2.0** if possible. This ensures that your implementation is automatically supported in all Job Processor and development (debug) environments.

IApplication Interface

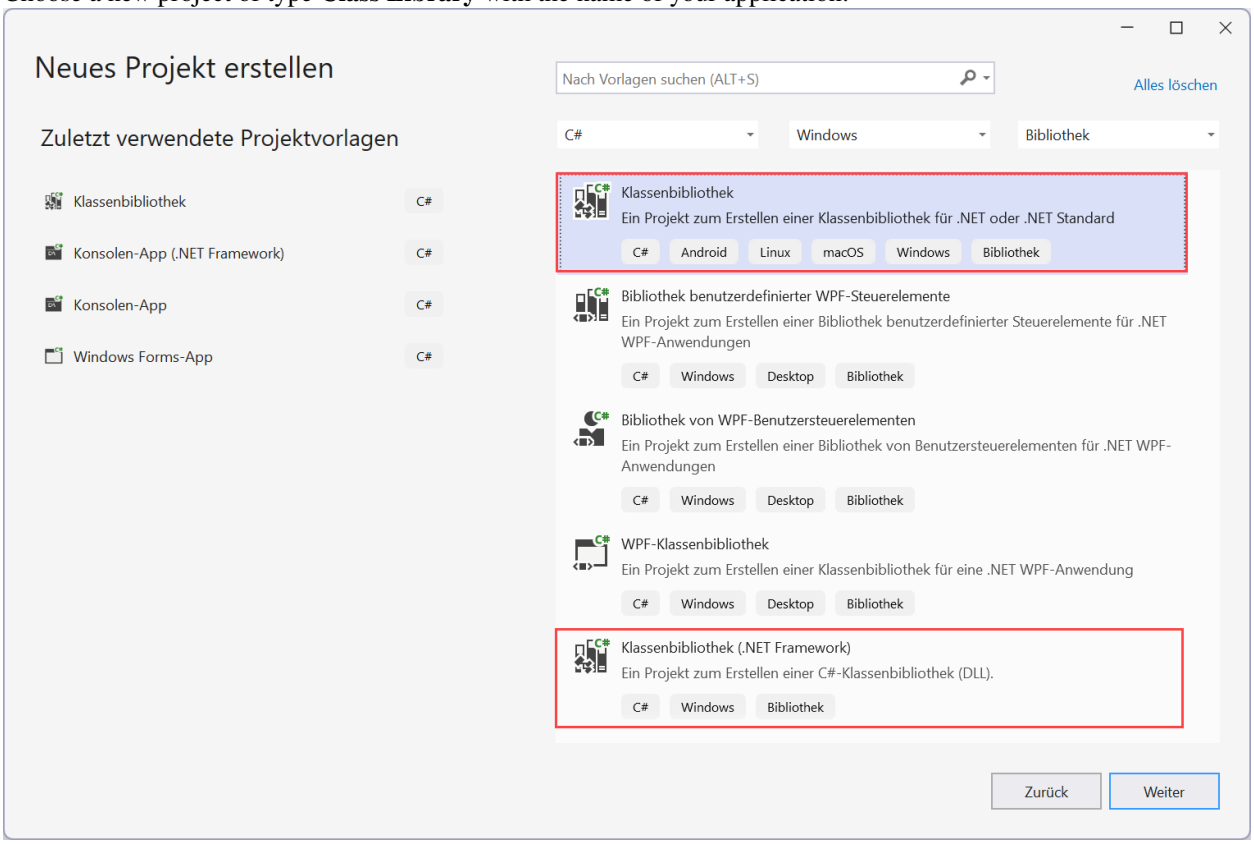
The API for Applications is designed around the *IApplication* interface and also includes types like IDocument, IExport, ect. and is designed as follows:



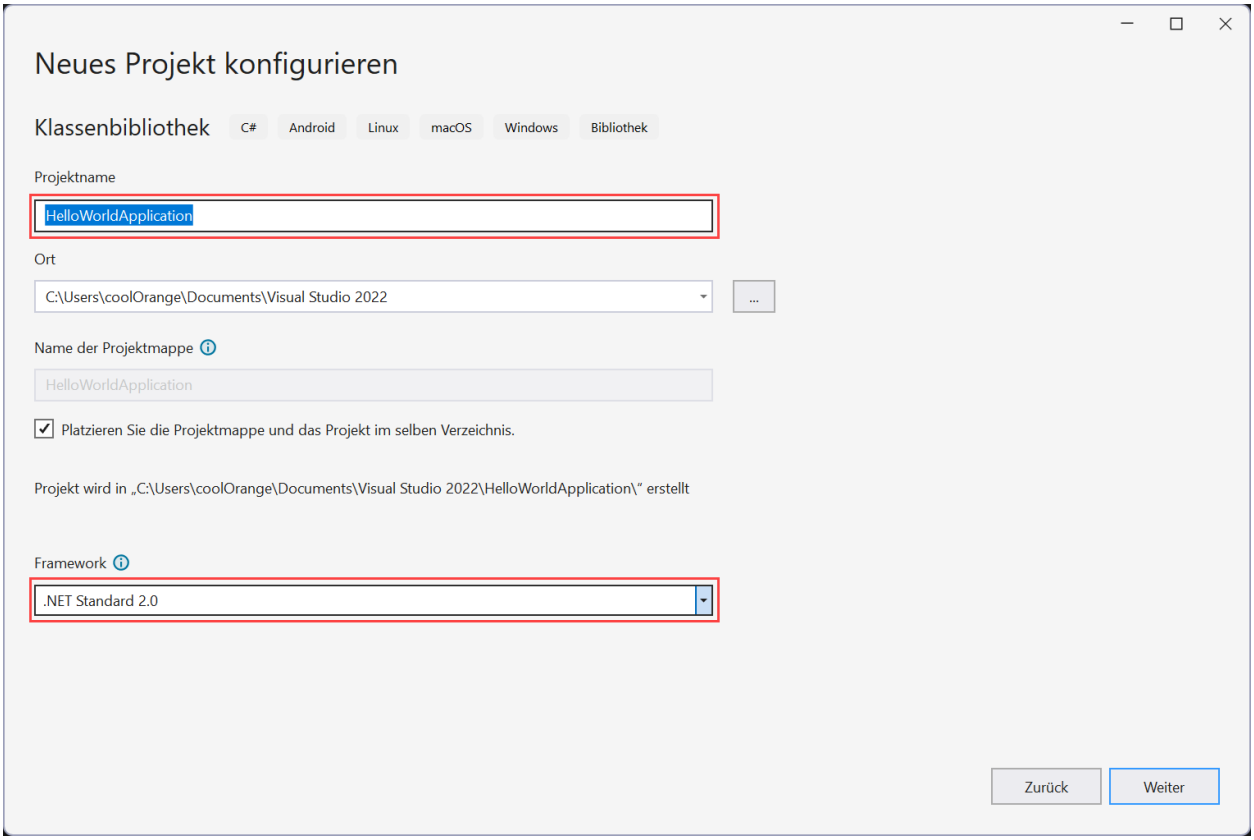
Visual Studio Project setup for Custom Applications

1. Create a new Solution in VisualStudio

Choose a new project of type **Class Library** with the name of your application:



Choose the name of the custom application and the target framework:



The new project should have the following settings:

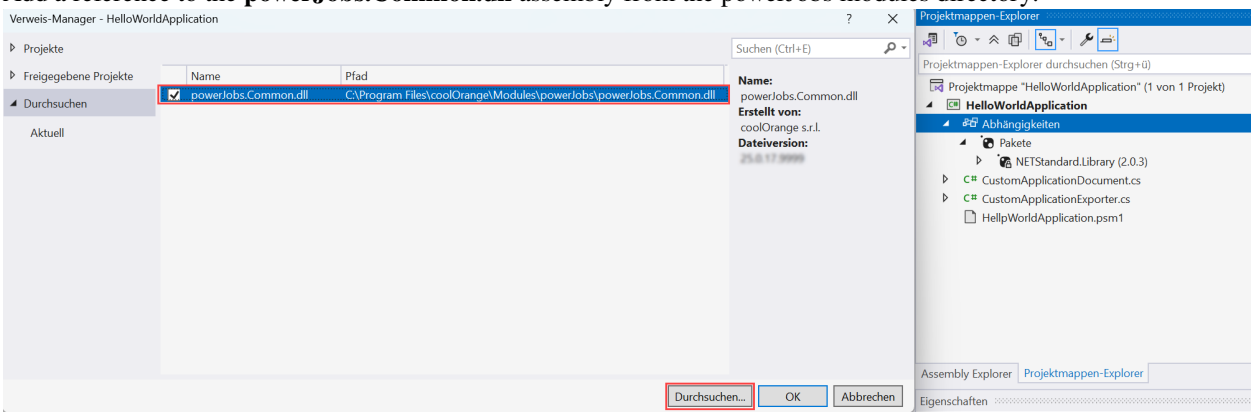
- Output Type: Class Library
- Target Framework: .NET Standard 2.0 (`netstandard2.0`) - *recommended*
  Alternatively, use .NET Framework 4.7 (`net47`) for Vault 2023-2025, or .NET 8 (`net8.0-windows`) for Vault 2026
- Platform Target: Any CPU

To ensure that the correct target framework version of your dependencies (e.g., from NuGet packages) is copied into the output directory, you should add the following MSBuild properties to your .csproj file:

```
<PropertyGroup>
    <RuntimeIdentifier>win-x64</RuntimeIdentifier>
    <CopyLocalLockFileAssemblies>true</CopyLocalLockFileAssemblies>
</PropertyGroup>
```

2. Reference the powerJobs.Common assembly

In Visual Studio right-click on *References* and click *"Add References"*.
Add a reference to the **powerJobs.Common.dll** assembly from the powerJobs modules directory:



For **.NET Framework** projects the library can also be easily referenced from the **GAC**; for **.NET 8** and **.NET Standard** projects, please reference it directly from *C:\Program Files\coolOrange\Modules\powerJobs*.
In both cases, *"Copy Local"* should be set to "false".

3. Create an IApplication implementation

In order to gain access to the IApplicaiton interace, the following **namespace** should be imported:

```
using powerJobs.Common.Applications;
```

Now we can define our Application class which implements the `IApplication` interface and handles the start and stop functionality.
In order to prevent troubles with long running applications, we recommend to derive your class from `powerjobs.Common.Applications.ApplicationBase` and implement the functionality IsRunning properly.
This will allow powerJobs Processor to automatically restart this application whenever required.
Note that the Application will only open documents configured in the IsSupportedFile (or `SupportedFileTypes`) functionality!

4. Create an IDocument implementation

The closing functionality should be implemented properly, in order to have no unmanaged resources lying around in memory later.
The constuctor can use the `OpenDocumentSettings` parameter for opening the correct file.
Additional all the options passed from the *Open-Document* cmldet can be consumed.
We recommend to derive your Document class from the `powerjobs.Common.Applications.DocumentBase` class.
The function `OpenDocument` of the custom application class (Step 3) must be extended to return an instance of this Type.
If the application is derived from `powerjobs.Common.Applications.ApplicationBase`, you just need to return a new instance of your class in `OpenDocument_Internal(OpenDocumentSettings openSettings)`.
Now the cmdlets *Open-Document* and *Close-Document* should already work.

5. Create an IDocumentExport implementation

The exporter takes care about different export formats and decides which export should be used from *Export-Document*.
The Application property `IDocumentExporter Exporter` has to be set to an instance of this IDocumentExporter. We recommend to derive your DocumentExporter class from `powerjobs.Common.Applications.DocumentExporterBase` class which handles multiple DocumentExports depending on the file format.
Even multiple exports can be registered in this class later.

For the moment a single IDocumentExport implementation can be provided.
Based on the name of this export (e.g. "PDF"), the DocumentExporter can decide to take this export when required.
We recommend to derive your DocumentExport class from the `powerjobs.Common.Applications.DocumentExportBase` class. After deriving from the class implement all the abstract functions.
By specifying the `SupportedDocumentTypes` your export will be called only for the specified formats.
The real export logic has to be implemented in the `Execute` function.
You can use the `ExportSettings` to gain access to the options passed to the *Export-Document* cmdlet.

6. Deploy to Modules directory

Next copy your application assembly into a new folder under the Modules directory.

7. Create Module to register the application

Create a new module in the modules directory and *register your application* in there, similar to the previously mentioned default applications:

```
Add-Type -Path "$($Env:POWERJOBS_MODULESDIR)\<YouApplicationFolder>\
↪<YouApplicationAssembly>.dll"
Register-Application ([<NameSpace.YouApplicationClass>])
```

Now you can *create your own job* and pass to the *Open-Document* the application name of your new application:

```
Open-Document -Application <YourApplicationName> -LocalFile $file.LocalPath
```

8. Best practice: Use PowerShell Console as Start Programm

Configure the Visual Studio project to launch and load your assembly in the Windows PowerShell Console or PowerShell 7 Console (if installed) for easier testing and debugging.
In the Project Debug settings, set:
Start external program: `C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe`
Command line arguments: `-noexit -command "Import-Module powerJobs"`

9. Launch the PowerShell Console or Job Processor

Press F5 to start the PowerShell Console and the powerJobs module imports and registers your application.
You can now execute the Cmdlets `Open-Document`, `Export-Document` and `Close-Document` in order to debug your implementations.
At the same time the application can now be used and tested in your job.

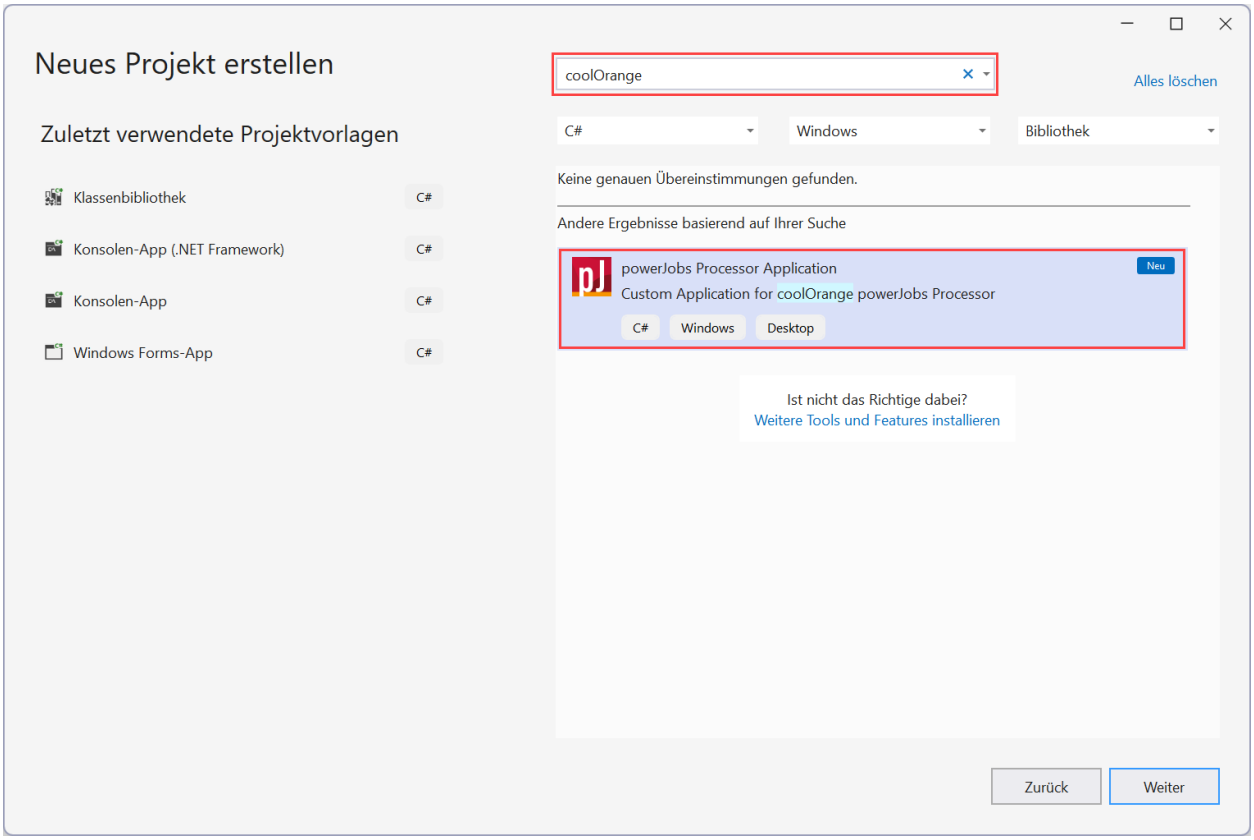
Visual Studio Template

The Visual Studio Template creates a basic project similar to following the *steps above*.

Visual Studio Project Template
Download: `C# Template`

Installation:
Copy the zip file to your VisualStudio Templates directory *%userprofile%\Documents\Visual Studio \Templates\ProjectTemplates\coolOrange*.

Start VisualStudio and *create a new project* by using the **powerJobs Processor Application** project template:



Then give your application a meaningful name (without characters like spaces or dots).

The assembly which contains your application and the according registration module will be deployed to the *Modules* directory, with the name of your project.
It is already configured to automatically launch the Windows PowerShell Console and import the `powerJobs` module when **pressing F5**.

4.2 Code Reference

4.2.1 Cmdlets

Clean-Up

Removes files and folders from the FileSystem.

Syntax

```
Clean-Up [-Directory <string>] [-Files<object>] [<CommonParameters>]
```

Parameters

Type	Name	Description	Optional
String	Directory	Directory that should be removed.	yes
String / FileInfo / powerVault File	Files	The files that will be removed.When passing powerVault file objects they have to be downloaded via Get-VaultFile -Download or Save-VaultFile.	yes

Return type

empty ← On failure the Exception/ErrorMessage can be accessed using $Error.

Remarks

When passing folders or files that does not exist, the cmdlet will not inform you or stop executing.
By passing only a directory path all the files in that folder are getting removed. The empty directories are getting removed too.
When passing files, they and the empty folders will be deleted.
When passing a directory and files to the cmdlet, the cmdlet removes all the passed files and empty directories.

Examples

Removing a file that was downloaded with Get-VaultFile

```
$file = Get-VaultFile -File '$/Designs/Test.ipt' -Download 'C:\Temp\Pad Lock' Clean-Up -
↪files @($file)
```

Removing a list of files that where downloaded with Save-VaultFile

```
$downloadedFiles =Save-VaultFile -File '$/Designs/Sample/Test.iam' -DownloadDirectory
↪"C:\temp\Vault\Sample\"
Clean-Up -files $downloadedFiles
```

Removing a whole directory

```
$file = Get-VaultFile -File '$/Designs/Test.ipt' -Download 'C:\temp\Vault\'
$downloadedFiles = Save-VaultFile -File '$/Designs/Sample/Test.iam' -DownloadDirectory
↪"C:\temp\Vault\Sample\"
Clean-Up 'C:\temp\Vault\'
```

Close-Document

Closes an open document.

Syntax

```
Close-Document [-Save] [-Document <IDocument>] [<CommonParameters>]
```

Parameters

Type	Name	Description	Optional
IDocument	Document	The cmdlet will close this document. Otherwise, the last opened document will be closed	yes
Boolean	Save	The application will save the document on close	yes

Return type

Bool: on success the cmdlet returns $true otherwise $false.
$result.Error ← The result has additionally an Error property which contains the Exception object
$result.Application ← Application is an implementation of IApplication
$result.Document ← Document is an implementation of IDocument

Remarks

After all documents got closed in *Inventor* or *InventorServer*, the active project file will be resetet to the original project file.

Examples

Closing the last opened document

```
Open-Document -localFile 'C:\Vault\Test.ipt'

Close-Document
```

Closing a specific document

```
$open1 = Open-Document -localFile 'C:\Vault\Test.ipt'
$open2 = Open-Document -localFile 'C:\Vault\Test.ipt'

Close-Document -Document $open1.Document
```

Checking if document was closed successfully

```
$result = Close-Document
if(-not $result) {
        throw "Failed to save and close document with error: ". $result.Error.Message
}
```

Close and Save document

```
$openResult = Open-Document -localFile 'C:\Vault\Test.ipt'
...
Close-Document -Document $openResult.Document -Save $true
```

Using the close result

```
$result = Close-Document if($result.Application.Name -eq 'Inventor') {
        $result.Application.Instance.Visible = $true
}
```

Export-Document

BMP

When opening the following file formats in  Inventor, *.bmp* files can be created for them (via *SaveAs*):

- Inventor IAM
- Inventor IPT
- Inventor IPN
- Inventor IDW
- Inventor DWG
- (AutoCAD DWG - has not been tested)

Unfortunately the export cannot be controlled via configuration.

DWF / DWFX

Inventor and InventorServer

When opening the following file formats in Inventor or InventorServer, *.dwf* and *.dwfx* files can be created for them (via the *TranslatorAddin*):

- Inventor IAM
- Inventor IPT
- Inventor IPN
- Inventor IDW
- Inventor DWG
- AutoCAD DWG

Configuration

A complete list of export options for the *"Tranlsator: DWG - Addin"* can be found here.

To understand how the various ini settings relate to the Inventor user interface see the *PDF - Inventor and InventorServer* section.
There you can also find configuration examples for 2D drawings. For 3D formats use following ini as submission:

Listing 1: Sample configuration for 3D documents - DWF_3D.ini

```
[EXPORT SELECT OPTIONS]
Publish_All_Sheets=0
Publish_3D_Models=0
Launch_Viewer=0
Password_Protect=0
Password=
Publish_Mode=62723
Enable_Large_Assembly_Mode=0
Enable_Measure=1
Enable_Printing=1
Enable_Markups=1
Enable_Markup_Edits=1
Output_Path=
Include_Sheet_Tables=1
Sheet_Metal_Flat_Pattern=0
Sheet_Metal_Style_Information=0
Sheet_Metal_Part=1
Weldment_Preparation=0
Weldment_Symbol=0
BOM_Structured=1
BOM_Parts_Only=1
Animations=0
Instructions=0
iAssembly_All_Members=0
iAssembly_3D_Models=0
iPart_All_Members=0
```

(continues on next page)

(continued from previous page)

```
iPart_3D_Models=0
Publish_Component_Props=1
Publish_Mass_Props=1
Include_Empty_Properties=0
Publish_Screenshot=0
Screenshot_DPI=1
Facet_Quality=69379
Force_Facet_Recompute=0
Facet_Recompute_Tolerance=0.001
Override_Sheet_Color=0
Sheet_Color=14085613
Sheet_Range=14081
Custom_Begin_Sheet=1
Custom_End_Sheet=-1
All_Color_AS_Black=0
Remove_Line_Weights=0
Vector_Resolution=400
TranscriptAPICall=0
```

DWG TrueView

DWG files (AutoCAD and Inventor) can be opened in DWG TrueView and *converted* to *.dwf* and *.dwfx*.

Configuration

To understand how the various dsd settings are generated by DWG TrueView and how to set them, see the *PDF - DWG TrueView* section.
There you can also find a dsd-file example.

To enable **multisheet- or singlesheet DWF or DWFx generation**, the `Type` setting can used:

```
Export-Document -format DWFx -To C:\Temp\Test.pdf -Options @{'Type'=4}
```

These are all possible `Type` settings for DWF and DWFx exports:

Value	Description	
0	SingleSheet(s) dwf	Separate dwf files are created for the model and for each layout of the drawing
1	MultiSheet dwf *(default for DWF)*	A single dwf file is created, with multiple pages for model and layouts
2	PlotterPageSetup	The plotter named in Page Settings is used for the export
3	SingleSheet(s) dwfx	Separate dwfx files are created for the model and for each layout of the drawing
4	MultiSheet dwfx *(default for DWFX)*	A single dwfx file is created, with multiple pages for model and layouts

DWG

This will export an AutoCAD dwg.

Drawings

When opening the following 2D drawing formats in Inventor or InventorServer, AutoCAD 2000 - *.dwg* files can be created for them (via the *TranslatorAddin*):

- Inventor IDW
- Inventor DWG
- AutoCAD DWG

Configuration

When exporting multisheet IDW or DWG files, **separate dwg files** are created foreach sheet by default.
To pack them into a single **zip file** - which has the same name as the dwg destination file - the *"Pack and Go"* option can be enabled by setting the flag `USE TRANSMITTAL` to Yes.

Listing 2: Sample configuration - DWG_2D.ini

```
[EXPORT SELECT OPTIONS]
Export_Acad_IniFile=C:\ProgramData\coolOrange\powerJobs\Modules\Export\DWG_2D.ini
AUTOCAD VERSION=AutoCAD 2000
CREATE AUTOCAD MECHANICAL=No
USE TRANSMITTAL=No
USE CUSTOMIZE=No
CUSTOMIZE FILE=C:\Users\Public\Documents\Autodesk\Inventor 2015\Design Data\DWG-DXF\
↪FlatPattern.xml
CREATE LAYER GROUP=No
PARTS ONLY=No
REPLACE SPLINE=No
CHORD TOLERANCE=0.001000
[EXPORT PROPERTIES]
SELECTED PROPERTIES=
[EXPORT DESTINATION]
SPACE=Model
SCALING=Geometry
ALL SHEETS=Yes
MAPPING=MapsBest
MODEL GEOMETRY ONLY=No
EXPLODE DIMENSIONS=No
SYMBOLS ARE BLOCKED=Yes
AUTOCAD TEMPLATE=
DESTINATION DXF=No
USE ACI FOR ENTITIES AND LAYERS=No
[EXPORT LINE TYPE & LINE SCALE]
LINE TYPE FILE=C:\Users\Public\Documents\Autodesk\Inventor 2015\COMPATIBILITY\Support\
↪invANSI.lin
Continuous=Continuous;0.
```

(continues on next page)

(continued from previous page)

```
Dashed=DASHED;0.
Dashed Space=DASHED_SPACE;0.
Long Dash Dotted=LONG_DASH_DOTTED;0.
Long Dash Double Dot=LONG_DASH_DOUBLE_DOT;0.
Long Dash Triple Dot=LONG_DASH_TRIPLE_DOT;0.
Dotted=DOTTED;0.
Chain=CHAIN;0.
Double Dash Chain=DOUBLE_DASH_CHAIN;0.
Dash Double Dot=DASH_DOUBLE_DOT;0.
Dash Dot=DASH_DOT;0.
Double Dash Dot=DOUBLE_DASH_DOT;0.
Double Dash Double Dot=DOUBLE_DASH_DOUBLE_DOT;0.
Dash Triple Dot=DASH_TRIPLE_DOT;0.
Double Dash Triple Dot=DOUBLE_DASH_TRIPLE_DOT;0.
```

Inventor Models

When opening the following 3D documents in Inventor or InventorServer, *.dwg* files can be created for them (via the *TranslatorAddin*):

- Inventor IAM
- Inventor IPT

Configuration

Listing 3: Sample configuration - DWG_3D.ini

```
[EXPORT SELECT OPTIONS]
Solid=True
Surface=True
Sketch=True
DwgVersion=23
```

Sheet Metal Parts

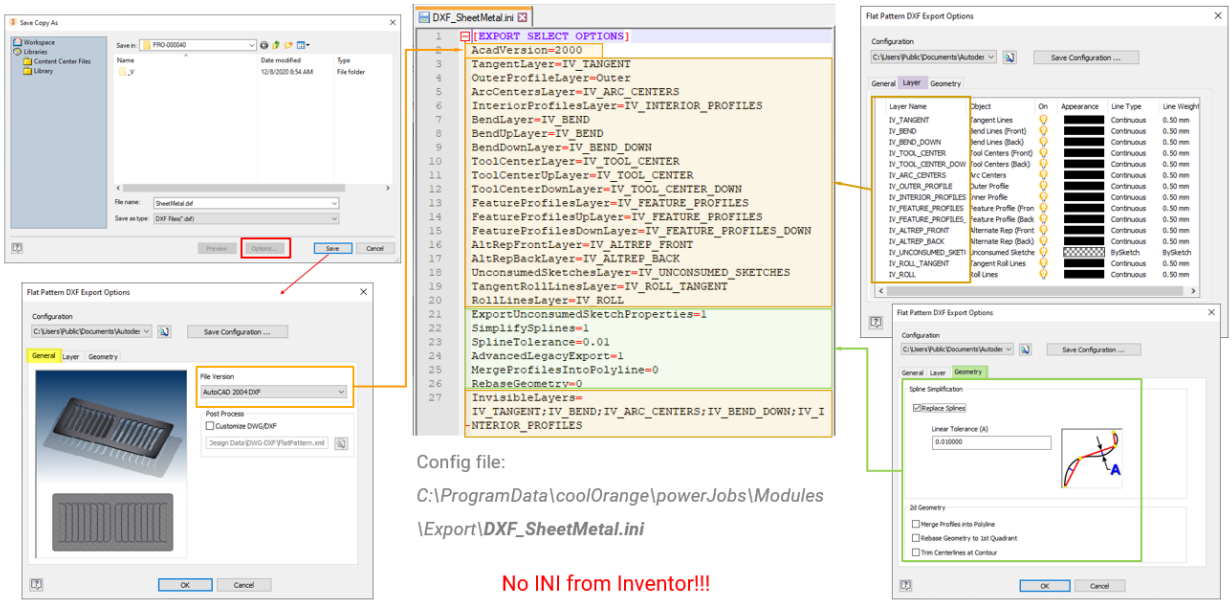
Unfolder Inventor sheet metal ipt files can be opened and converted to *.dwg* files in Inventor and InventorServer (via *DataIO*).

Configuration

A complete list of arguments for the DataIO addin can be found here.

A spline is a curve that passes through a set of points which influence the shape of the curve.
To disable the **replacement of splines** with linear segments, the flag `SimplifySplines` must be turned off.
This setting is activated by default and the corresponding **chord tolerance** (see option *"Linear Tolerance (A)"*) can be ajusted with the setting `SplineTolerance`.
When this double value is set to 0, the spline passes directly through the fit points. With larger tolerance values, the spline passes near the fit points.

How the various ini-settings relate to the Inventor user interface:



Warning: It is **NOT possible to use .ini-files created by Inventor** for this export!
The sheetmetal export internally uses DataIO, which is not compatible to the configuration files created in the Inventor user interface (see the Autodesk forum for more details).

Listing 4: Sample configuration - DWG_SheetMetal.ini

```
[EXPORT SELECT OPTIONS]
AcadVersion=2000
TangentLayer=IV_TANGENT
OuterProfileLayer=Outer
ArcCentersLayer=IV_ARC_CENTERS
InteriorProfilesLayer=IV_INTERIOR_PROFILES
BendLayer=IV_BEND
BendUpLayer=IV_BEND
BendDownLayer=IV_BEND_DOWN
ToolCenterLayer=IV_TOOL_CENTER
ToolCenterUpLayer=IV_TOOL_CENTER
ToolCenterDownLayer=IV_TOOL_CENTER_DOWN
FeatureProfilesLayer=IV_FEATURE_PROFILES
FeatureProfilesUpLayer=IV_FEATURE_PROFILES
FeatureProfilesDownLayer=IV_FEATURE_PROFILES_DOWN
AltRepFrontLayer=IV_ALTREP_FRONT
AltRepBackLayer=IV_ALTREP_BACK
UnconsumedSketchesLayer=IV_UNCONSUMED_SKETCHES
TangentRollLinesLayer=IV_ROLL_TANGENT
RollLinesLayer=IV_ROLL
ExportUnconsumedSketchProperties=1
SimplifySplines=1
SplineTolerance=0.01
AdvancedLegacyExport=1
MergeProfilesIntoPolyline=0
RebaseGeometry=0
InvisibleLayers=
```

DXF

Drawings

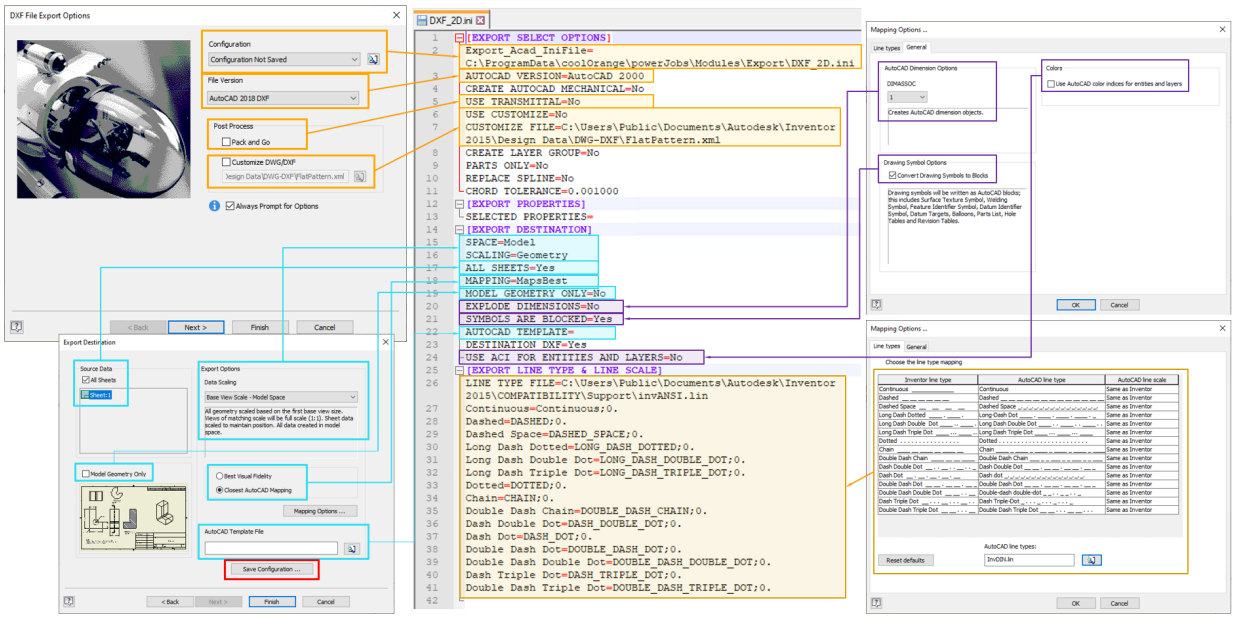
When opening the following 2D drawing formats in Inventor or InventorServer, *.dxf* files can be created for them (via the *TranslatorAddin*):

- Inventor IDW
- Inventor DWG
- AutoCAD DWG

Configuration

When exporting multisheet IDW or DWG files, **separate dxf files** are created foreach sheet by default.
To pack them into a single **zip file** - which has the same name as the dxf destination file - the *"Pack and Go"* option can be enabled by setting the flag `USE TRANSMITTAL` to Yes.

How the various ini-settings relate to the Inventor user interface:



Warning: If you created an **.ini-file via Inventor** you must add a setting named **Export_Acad_IniFile** which value has to be set to the path of the *.ini*-file itself.
This setting is used by *Export-Document -Options* parameter, which is why the commandlet then reads all the contained settings.
An example can be seen below.

Listing 5: Sample configuration - DXF_2D.ini

```
[EXPORT SELECT OPTIONS]
Export_Acad_IniFile=C:\ProgramData\coolOrange\powerJobs\Modules\Export\DXF_2D.ini
AUTOCAD VERSION=AutoCAD 2000
CREATE AUTOCAD MECHANICAL=No
USE TRANSMITTAL=No
USE CUSTOMIZE=No
CUSTOMIZE FILE=C:\Users\Public\Documents\Autodesk\Inventor 2015\Design Data\DWG-DXF\
↪FlatPattern.xml
CREATE LAYER GROUP=No
PARTS ONLY=No
REPLACE SPLINE=No
CHORD TOLERANCE=0.001000
[EXPORT PROPERTIES]
SELECTED PROPERTIES=
[EXPORT DESTINATION]
SPACE=Model
```

(continues on next page)

(continued from previous page)

```
SCALING=Geometry
ALL SHEETS=Yes
MAPPING=MapsBest
MODEL GEOMETRY ONLY=No
EXPLODE DIMENSIONS=No
SYMBOLS ARE BLOCKED=Yes
AUTOCAD TEMPLATE=
DESTINATION DXF=Yes
USE ACI FOR ENTITIES AND LAYERS=No
[EXPORT LINE TYPE & LINE SCALE]
LINE TYPE FILE=C:\Users\Public\Documents\Autodesk\Inventor 2015\COMPATIBILITY\Support\
↪invANSI.lin
Continuous=Continuous;0.
Dashed=DASHED;0.
Dashed Space=DASHED_SPACE;0.
Long Dash Dotted=LONG_DASH_DOTTED;0.
Long Dash Double Dot=LONG_DASH_DOUBLE_DOT;0.
Long Dash Triple Dot=LONG_DASH_TRIPLE_DOT;0.
Dotted=DOTTED;0.
Chain=CHAIN;0.
Double Dash Chain=DOUBLE_DASH_CHAIN;0.
Dash Double Dot=DASH_DOUBLE_DOT;0.
Dash Dot=DASH_DOT;0.
Double Dash Dot=DOUBLE_DASH_DOT;0.
Double Dash Double Dot=DOUBLE_DASH_DOUBLE_DOT;0.
Dash Triple Dot=DASH_TRIPLE_DOT;0.
Double Dash Triple Dot=DOUBLE_DASH_TRIPLE_DOT;0.
```

Warning: The *.ini*-file above contains paths which reference "Inventor 2015". If you are using another version than Inventor 2015 you have to update them.

Sheet Metal Parts

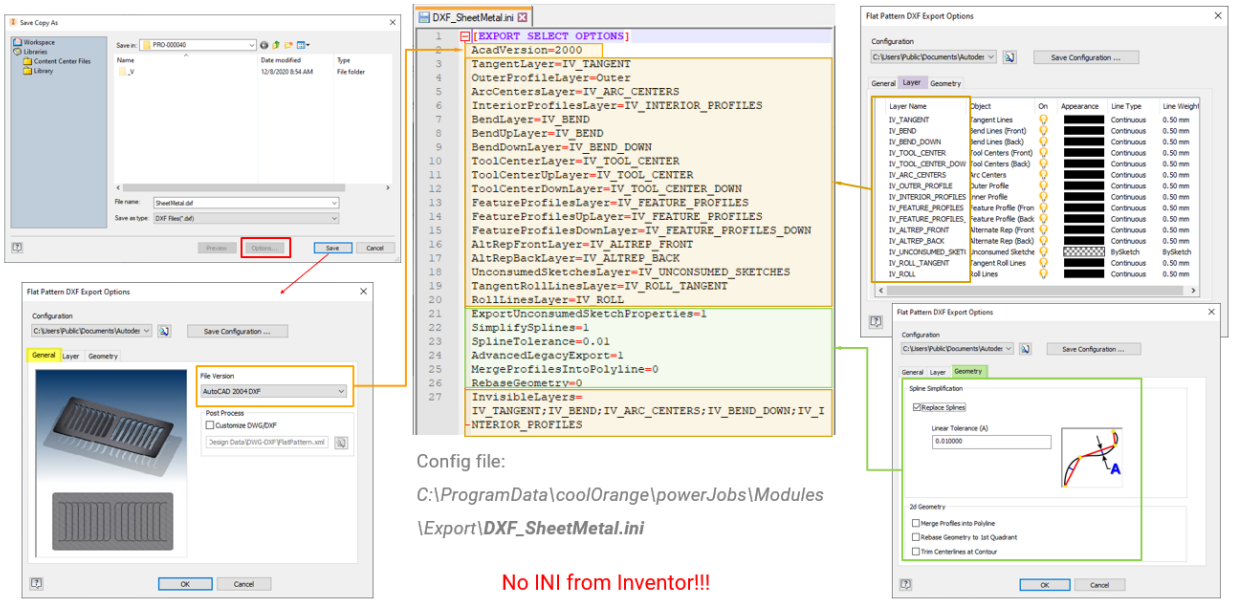
Unfolder Inventor sheet metal ipt files can be opened and converted to *.dxf* files in Inventor and InventorServer (via *DataIO*).

Configuration

A complete list of arguments for the DataIO addin can be found here.

A spline is a curve that passes through a set of points which influence the shape of the curve.
To disable the **replacement of splines** with linear segments, the flag `SimplifySplines` must be turned off.
This setting is activated by default and the corresponding **chord tolerance** (see option *"Linear Tolerance (A)"*) can be ajusted with the setting `SplineTolerance`.
When this double value is set to 0, the spline passes directly through the fit points. With larger tolerance values, the spline passes near the fit points.

How the various ini-settings relate to the Inventor user interface:



Warning: It is **NOT possible to use .ini-files created by Inventor** for this export!
The sheetmetal export internally uses DataIO, which is not compatible to the configuration files created in the Inventor user interface (see the Autodesk forum for more details).

Listing 6: Sample configuration - DXF_SheetMetal.ini

```
[EXPORT SELECT OPTIONS]
AcadVersion=2000
TangentLayer=IV_TANGENT
OuterProfileLayer=Outer
ArcCentersLayer=IV_ARC_CENTERS
InteriorProfilesLayer=IV_INTERIOR_PROFILES
BendLayer=IV_BEND
BendUpLayer=IV_BEND
BendDownLayer=IV_BEND_DOWN
ToolCenterLayer=IV_TOOL_CENTER
ToolCenterUpLayer=IV_TOOL_CENTER
ToolCenterDownLayer=IV_TOOL_CENTER_DOWN
FeatureProfilesLayer=IV_FEATURE_PROFILES
FeatureProfilesUpLayer=IV_FEATURE_PROFILES
FeatureProfilesDownLayer=IV_FEATURE_PROFILES_DOWN
AltRepFrontLayer=IV_ALTREP_FRONT
AltRepBackLayer=IV_ALTREP_BACK
UnconsumedSketchesLayer=IV_UNCONSUMED_SKETCHES
TangentRollLinesLayer=IV_ROLL_TANGENT
RollLinesLayer=IV_ROLL
ExportUnconsumedSketchProperties=1
SimplifySplines=1
SplineTolerance=0.01
AdvancedLegacyExport=1
MergeProfilesIntoPolyline=0
RebaseGeometry=0
```

(continues on next page)

(continued from previous page)

```
InvisibleLayers=IV_TANGENT;IV_BEND;IV_ARC_CENTERS;IV_BEND_DOWN;IV_INTERIOR_PROFILES
```

GIF

When opening the following file formats in Inventor, *.gif* files can be created for them (via *SaveAs*):

- Inventor IAM
- Inventor IPT
- Inventor IPN
- Inventor IDW
- Inventor DWG
- (AutoCAD DWG - has not been tested)

Unfortunately the export cannot be controlled via configuration.

IGES

Inventor Models

When opening the following 3D documents in Inventor or InventorServer, *.dwg* files can be created for them (via the *TranslatorAddin*):

- Inventor IAM
- Inventor IPT

Configuration

See this *"Tranlsator: IGES - Addin"* page for details about possible export options.

The following ini can be used as a submission:

Listing 7: Sample configuration - IGES.ini

```
[EXPORT SELECT OPTIONS]
IncludeSketches=True
GeometryType=1
SurfaceType=0
SolidFaceType=0
export_fit_tolerance=0.00100076
```

JPEG

When opening the following file formats in Inventor, *.jpg* files can be created for them (via *SaveAs*):

- Inventor IAM
- Inventor IPT
- Inventor IPN
- Inventor IDW
- Inventor DWG
- (AutoCAD DWG - has not been tested)

PDF

Inventor and InventorServer

When opening the following 2D drawing formats in Inventor or InventorServer, *.pdf* files can be created for them (via the *TranslatorAddin*):

- Inventor IDW
- Inventor DWG
- AutoCAD DWG

The used *"Tranlsator: PDF - Addin"* also supports generating pdf files out of 3D documents **but without any settings**:

- Inventor IAM
- Inventor IPT
- Inventor IPN

Configuration

To enable **multisheet- or singlesheet PDF** generation, the `Publish_All_Sheets` flag can be turned on or off. Additionally you have to change the value of `Sheet_Range` to 14082, since this is the value for printing all Sheets, instead of the current sheet.

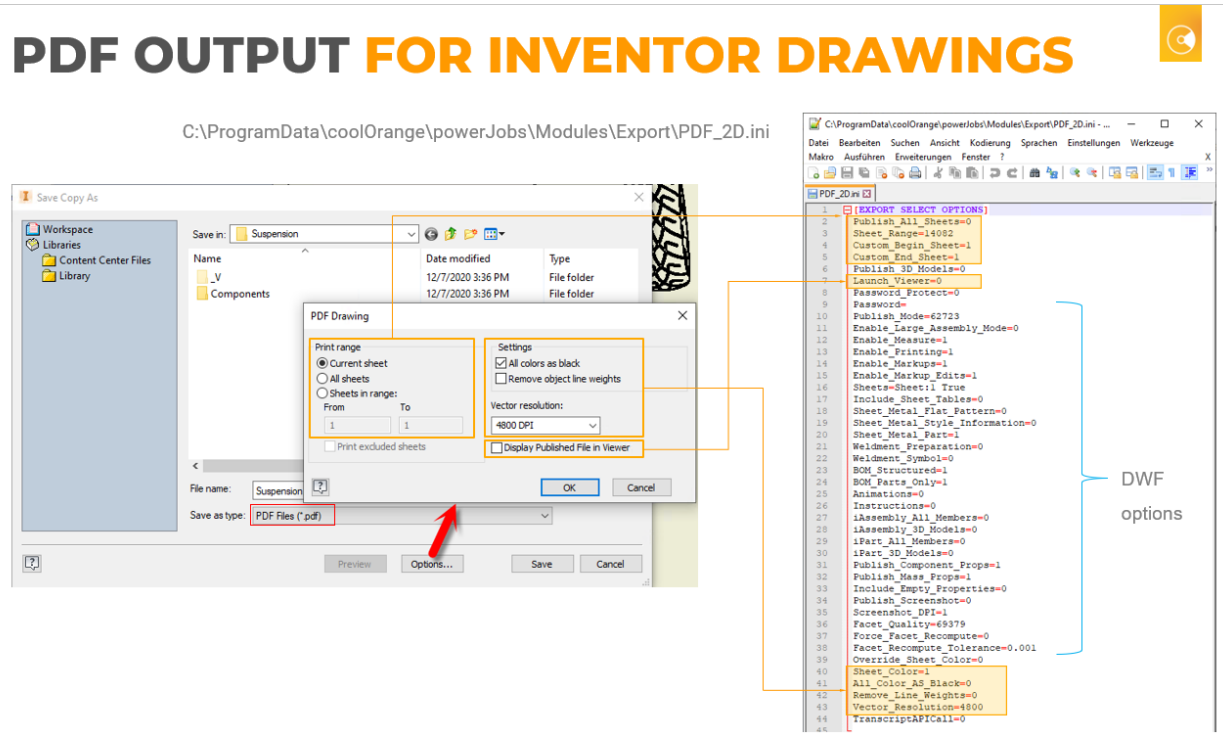
As **Print range** you can specify if the **current sheet**, **all sheets** or **a range of sheets** should be exported. You have to set the `Sheet_Range` setting to the respective value.

Value	Description
1408	Print all Sheets of the Drawing.
1408	Print the current Sheet in the Drawing.
1408	A sheet range allows to export only the sheets specified in a specifc range.Specify the starting sheet page via `Custom_Begin_Sheet` and the end page via `Custom_End_Sheet`

The `Publish_Mode` defines what of your document will be exported to the PDF. Per default it is set to print your **selected data**.

Value	Description
62721	Only the current view and the state of the document will be published
62722	The full context of the current document will be published
62723	Only the selected data will be published

How the various ini-settings relate to the Inventor user interface:



Listing 8: Sample configuration for Drawings - PDF_2D.ini

```
[EXPORT SELECT OPTIONS]
Publish_All_Sheets=0
Sheet_Range=14082
Publish_3D_Models=0
Launch_Viewer=0
Password_Protect=0
Password=
Publish_Mode=62723
Enable_Large_Assembly_Mode=0
Enable_Measure=1
Enable_Printing=1
Enable_Markups=1
Enable_Markup_Edits=1
Sheets=Sheet:1 True
Include_Sheet_Tables=0
Sheet_Metal_Flat_Pattern=0
Sheet_Metal_Style_Information=0
Sheet_Metal_Part=1
Weldment_Preparation=0
Weldment_Symbol=0
BOM_Structured=0
BOM_Parts_Only=0
Animations=0
Instructions=0
iAssembly_All_Members=0
iAssembly_3D_Models=0
iPart_All_Members=0
iPart_3D_Models=0
Publish_Component_Props=1
Publish_Mass_Props=1
Include_Empty_Properties=0
Publish_Screenshot=0
Screenshot_DPI=1
Facet_Quality=69379
Force_Facet_Recompute=0
Facet_Recompute_Tolerance=0
Override_Sheet_Color=0
Sheet_Color=1
Custom_Begin_Sheet=1
Custom_End_Sheet=1
All_Color_AS_Black=0
Remove_Line_Weights=0
Vector_Resolution=4800
TranscriptAPICall=0
```

DWG TrueView

DWG files (AutoCAD and Inventor) can be opened in DWG TrueView and *converted* to *.pdf* (or *.dwf, .dwfx*).

Configuration

DWG TrueView creates a ".dsd" file that contains the export settings that allow to manipulate the result of the export (unlike Inventor-exports there are no ".ini" files).

These settings are read from the **page settings** of the dwg file itself or from another specified dwg file.
Our sample jobs use the installed *powerJobs\Module\Export\PDF.dwg* by default.

Here's how the *Export-Document* cmdlet can be used to accomplish this:

- When no configuration file is specified, the default **page setup of the source drawing** itself will be used.
- A **configuration file can be specified** in the *-Options* parameter. Files of type .dwg, .dws, .dwt or .dxf can be provided.
  When doing so, DWG TrueView will look for a page setup named "*AutoloaderModelSetup*" for **model space** configuration and "*AutoloaderLayoutSetup*" for **layout space** configuration.

Examples:

```
Open-Document -Application 'DWG TrueView' -LocalFile 'C:\Vault\Tablet.dwg'

# Using the default Page Setup of the source drawing Tablet.dwg:
Export-Document -format PDF -To 'C:\Vault\Tablet_1.pdf'

# Usign the Page Setup of the specified drawing PDF.dwg:
Export-Document -format PDF -To 'C:\Vault\Tablet_2.pdf' -Options "$($env:POWERJOBS_
↪MODULESDIR)Export\PDF.dwg"
```

The settings in the automatically generated **DSD file** can be further manipulated with additional *-Options* or with the *-OnExport* script block.

For example, to enable **multisheet- or singlesheet PDF generation**, the `Type` setting can used:

```
Export-Document -format PDF -To C:\Temp\Test.pdf -Options @{'Type'='6'}
```

These are all possible `Type` settings for PDF exports:

Value	Description	
2	PlotterPageSetup	The plotter named in Page Settings is used for the export
5	SingleSheet(s)	Separate pdf files are created for the model and for each layout of the drawing
6	MultiSheet *(default)*	A single pdf file is created, with multiple pages for model and layouts

More information on how to customize the PDF creation can be found on our knowledge base.

Listing 9: Sample dsd file - Sample.dsd

```
[DWF6Version]
Ver=1
[DWF6MinorVersion]
```

(continues on next page)

(continued from previous page)

```
MinorVer=1
[DWF6Sheet:migration-Model]
DWG=C:\Temp\migration.dwg\migration.dwg
Layout=Model
Setup=
OriginalSheetPath=C:\Temp\migration.dwg\migration.dwg
Has Plot Port=0
Has3DDWF=0
[DWF6Sheet:migration-Layout1]
DWG=C:\Temp\migration.dwg\migration.dwg
Layout=Layout1
Setup=
OriginalSheetPath=C:\Temp\migration.dwg\migration.dwg
Has Plot Port=0
Has3DDWF=0
[Target]
Type=6
DWF=C:\Temp\migration.dwg\migration.dwg.pdf
OUT=C:\Temp\migration.dwg
PWD=
[MRU Local]
MRU=0
[MRU Sheet List]
MRU=0
[PdfOptions]
IncludeHyperlinks=TRUE
CreateBookmarks=TRUE
CaptureFontsInDrawing=TRUE
ConvertTextToGeometry=FALSE
VectorResolution=1200
RasterResolution=400
[AutoCAD Block Data]
IncludeBlockInfo=0
BlockTmplFilePath=
[SheetSet Properties]
IsSheetSet=FALSE
IsHomogeneous=FALSE
SheetSet Name=
NoOfCopies=1
PlotStampOn=FALSE
ViewFile=FALSE
JobID=0
SelectionSetName=
AcadProfile=<<Unnamed Profile>>
CategoryName=
LogFilePath=
IncludeLayer=FALSE
LineMerge=FALSE
CurrentPrecision=
PromptForDwfName=FALSE
PwdProtectPublishedDWF=FALSE
PromptForPwd=FALSE
```

(continues on next page)

(continued from previous page)

```
RepublishingMarkups=FALSE
PublishSheetSetMetadata=FALSE
PublishSheetMetadata=FALSE
3DDWFOptions=0 1
[PublishOptions]
InitLayouts=TRUE
```

PNG

When opening the following file formats in Inventor, *.png* files can be created for them (via *SaveAs*):

- Inventor IAM
- Inventor IPT
- Inventor IPN
- Inventor IDW
- Inventor DWG
- (AutoCAD DWG - has not been tested)

Unfortunately the export cannot be controlled via configuration.

STEP

Inventor Models

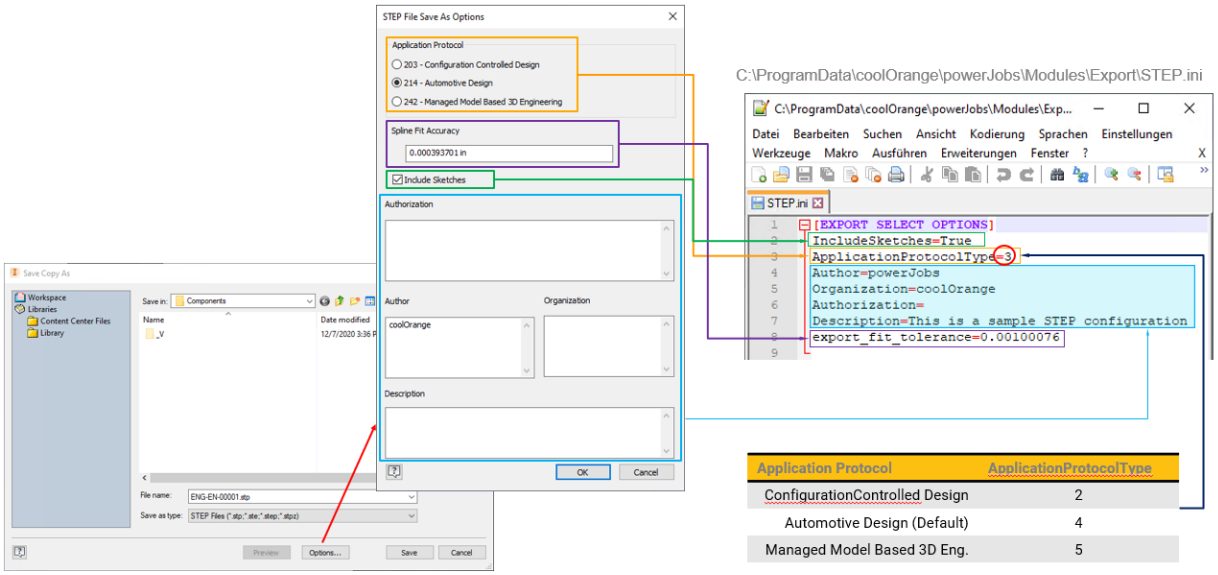
When opening the following 3D documents in Inventor or InventorServer, *.stp* and *.step* files can be created for them (via the *TranslatorAddin*):

- Inventor IAM
- Inventor IPT

Configuration

A complete list of export options for the *"Tranlsator: STEP - Addin"* can be found here.

How the various ini-settings relate to the Inventor user interface:



Listing 10: Sample configuration - STEP.ini

```
[EXPORT SELECT OPTIONS]
IncludeSketches=True
ApplicationProtocolType=3
Author=powerJobs
Organization=coolOrange
Authorization=
Description=This is a sample STEP configuration
export_fit_tolerance=0.00100076
```

TIFF

When opening the following file formats in Inventor, *.tiff* files can be created for them (via *SaveAs*):

- Inventor IAM
- Inventor IPT
- Inventor IPN
- Inventor IDW
- Inventor DWG
- (AutoCAD DWG - has not been tested)

Note: Inventor SaveAs function supports only destination files ending with extension *.tiff*, and **not .tif**!
When passing a destination file with extension .tif, the Inventor API will not throw exception but possibly no Tiff file will be created.

Unfortunately the export cannot be controlled via configuration.

Exports the document in the specified format.

Syntax

```
Export-Document -To <String> [-Format <String>] [-Options <Hashtable | String>] [-Document <IDocument>] [-OnExport <Scriptblock | String>] [<CommonParameters>]
```

Parameters

Type	Name	Description	Optional
String	To	Location where the converted file should be saved	no
String	Format	File extensions:*PDF*, *DWF / DWFX* , *IGES* , *STEP* , *DXF* , *DWG* , *GIF* , *BMP* , *TIFF* , *PNG* , *JPEG*. If not specified the export allows to place your custom logic in *-OnExport*	yes
Hashtable / String	Options	The document will be exported with the passed options, or configuration Files	yes
IDocument	Document	The cmdlet will export this document, otherwise the last opened document will be exported	yes
Scriptblock / String	OnExport	The script block or function that will be executed after collecting the options but before the real export starts	yes

Return type

Bool: on success the cmdlet returns $true otherwise $false.
$result.Error ← The result has additionally an Error property which contains the Exception object
$result.Application ← Application is an implementation of IApplication
$result.Document ← Document is an implementation of IDocument

Remarks

When passing invalid **-Options** the export will not fail, in that case the **default settings** will be used.

OnExport

For the **-OnExport** parameter it is either possible to pass a function name or a script block, which will be executed before the actual export starts.
Therefore the export of the document fails when an exception is thrown within this parameter.

Following parameters are available: $export, $document (=IDocument), $application (=IApplication).
Depending on the application and the output format, the export object has different types.
Independent of the type, the **$export** object has always following base properties:

Type	Name	Description	Access type
string	Name	The name of the export is the fromat that you pass to the cmdlet	read-only
IApplication	Application	The application which is used for the export.	read-only
IDocument	SourceDocument	The document that will be exported	read-only
IFileInfo	DestinationFile	The output location and name that was passed to the cmdlet	read-only
ExportSettings	Settings	The exporting options that are used for the export. ExportSettings.Options real HashTable entiries of -Options	read-only
HashSet	SupportedDocumentTypes	Conatins all the supported input formats of the export	read-only

TranslatorAddin

Exports: *PDF*, *DWF/DWFX*, *DWG*, *DXF*, *IGES*, *STEP*,
Application: Inventor, InventorServer

Exports that are using the TranslatorAddin are working with a rich $export object, that can be manipulated in different ways.
Additional to the base properties, a TranslatorAddinExport has following properties:

Type	Name	Description	Access type
string	TranslatorName	The ID of the translator that is used. Usually this is a GUID	read-write
TranslatorAddIn	TranslatorAddin	Represents the Inventor API TranslatorAddin object. By default it is created by using the TranslatorName	read-write
TranslationContext	Context	Represents the Inventor API TranslationContext. By default a new one is created with Type IOMechanismEnum.kFileBrowseIOMechanism	read-write
Hashtable	Options	Represents the -Options that are passed to the cmldet. When passing INI files, this hash contains already it's data. Options are only set, when HasSaveCopyAsOptions returns True	read-write
DataMedi	DataMed	Represents the Inventor API DataMedium. By default FileName is set to the destination file from -To argument	read-write

DataIO

Exports: *DWG*, *DXF*
Application: Inventor, InventorServer

Exports that are using the DataIO are the ones for exporting SheetMetal documents.
Additional to the base properties, a SheetMetalExport has following properties:

Type	Name	Description	Access type
string	Format	The first options part for DataIO export, e.g. *FLAT PATTERN DWG*?...	read
SheetMetalComponentDefinition	SheetMetalComponentDefinition	Represents the Inventor API SheetMetalComponentDefinition	read-write
bool	Unfold	when set to True, the SheetMetal will be unfold. By default it is set to *!HasFlatPattern*	read-write
DataIO	Data	Represents the Inventor API DataIO object. It is the DataIO value of the SheetMetalComponentDefinition	read
Hashtable	Options	Represents the -Options that are passed to the cmldet. When passing INI files, this hash contains already it's data. All the Options are formatted as this when passing to DataIO options: FLAT PATTERN DXF?*AcadVersion=2000&BendLayer=IV_BEND*"	read-write

SaveAs

Exports: *BMP*, *GIF*, *JPEG*, *PNG*, *TIFF*
Application: Inventor, InventorServer

The $export object of exports that are working with SaveAs, has only the property *SaveAsCopy* to configure as an *bool*. By default this setting is set to True.

TrueView

Exports: *PDF*, *DWF/DWFx*
Application: DWG TrueView

Exports that are using DWG TrueView as an application have in addtion to the base properties, a property DSDFile.

Type	Name	Description	Access type
FileInfo	DSDFile	This property exposes multitple properties which allow you to easy manipulate the dsd file	read-write

Examples

Exporting the last opened document to DXF with default options

```
Open-Document -LocalFile 'C:\Vault\AutoCad\bottom_plate.dwg' -Application 'InventorServer
↪'

Export-Document -Format DWF -To 'C:\Vault\AutoCad\bottom_plate.dxf'
```

Exporting with Options

```
Export-Document -format DXF -To C:\Temp\Test.dxf -Options C:\Temp\test.ini
Export-Document -format PDF -To C:\Temp\Test.dxf -Options @{'Vector_Resolution'='50000'}
```

Exporting multiple documents

```
$partDocument = Open-Document -LocalFile 'C:\Vault\Designs\Padlock\External\Case Back.ipt
↪'
$assemblyDocument = Open-Document -LocalFile 'C:\Vault\Designs\Padlock\Assemblies\Pad␣
↪Lock.iam'

Export-Document -Document $partDocument.Document -format JPEG -To C:\Temp\Case Back.jpg
Export-Document -Document $assemblyDocument.Document -format BMP -To C:\Temp\Pad Lock.bmp
```

Validating Export result

```
$result = Export-Document -format PDF -To C:\Temp\Test.pdf
if(-not $result) {
        throw "Failed to export with error: ". $result.Error.Message
}
```

Using the export result

```
$result = Export-Document -format PDF -To C:\Temp\Test.pdf if($result.Application.Name -
↪eq 'Inventor') {
        $result.Document.Instance
        #...
}
```

Handling exceptions thrown in OnExport script block

```
$exportResult = Export-Document -Format DWF -To C:\Temp\Test.dwf -OnExport {
        #Statement throwing exception
}
if(!$exportResult -and $null -ne $exportResult.Error.InnerException) {
        Write-Host ("Exception thrown in OnExport: " + $exportResult.Error.
↪InnerException.ErrorRecord)
}
```

Exporting with even more possibilities in OnExport script block

```
Export-Document -format PDF -To C:\Temp\Test.pdf -Options C:\Temp\PDF.ini -onExport {
        param($export)

```

(continues on next page)

(continued from previous page)

```
        $export.Options['All_Color_AS_Black']='1'
        $export.Options.Remove('Password_protect')
        $export.Options.Remove('Password')
        #...
}
```

Doing special stuff in the OnExport, by using the Inventor API

```
Open-Document -LocalFile 'C:\Vault\Designs\Padlock\Assemblies\Pad Lock.iam'
$result = Export-Document -Format DWF -To C:\Temp\Test.dwf -OnExport {
        param($export)

        $document = $export.SourceDocument
        $export.Application.Instance.Visible=$true
        if($document.Instance.DocumentType -eq [Inventor.
↪DocumentTypeEnum]::kAssemblyDocumentObject ) {
                $export.Options['Design_Views'] = ...
        }
}
```

Doing special stuff in the OnExport, by manipulating the DSD for TrueView

```
Open-Document -LocalFile 'C:\Vault\Designs\Acad 2015\ACADE Tutorial\DEMO02.DWG'
Export-Document -Format PDF -To C:\Temp\DEMO02.DWG.pdf -OnExport {
        param($export)

        $export.DsdFile.PublishExportOptions.Type=5
}
```

Open-Document

Opens the specified document in one of the *registered application*.

Syntax

```
Open-Document -LocalFile <string> [-Application <string>] [-Options <HashTable>] [
↪<CommonParameters>]
```

Parameters

Type	Name	Description	Optional
String	LocalFile	The path to the document that should be opened	no
String	Application	Can be used to force opening the document with a specific *application* by specifying its Name.Otherwise a valid application will be detected automatically.	yes
HashTa	Options	The application will open the document with the passed options	yes

Return type

Bool: on success the cmdlet returns $true otherwise $false.

The result has additional properties:
$result.Error ← Error contains the Exception object
$result.Application ← Application is an implementation of the *IApplication Interface*
$result.Document ← Document is an implementation of the *IDocument Interface*

Remarks

When the **-Application** parameter is **not** specified, the best matching application for the document is determined automatically:

- Only *registered applications* that are installed and support the **-LocalFile** are taken into account.
- AutoCAD documents are preferably always opened in *DWG TrueView*, and for Inventor documents *Inventor* is used by default (if installed).
- Many documents can be opened by multiple applications, so it ultimately depends on the registration order which application is used to open the specified file.
  For example, Inventor documents can be opened from *Inventor* and *InventorServer*. In this case the first-registered application in the *coolOrange.Applications.psm1* module is used for opening by default
  (see *example: Prefer opening Inventor documents with InventorServer by default instead of Inventor*).

Depending on the application that opens the document, application specific default **-Options** are be used.
To check which options were used to open the document, `$result.Document.OpenSettings` can be used: it contains all the properties used to open the document.

The following section lists all possible options for the default applications:

Inventor and InventorServer

For *Inventor* and *InventorServer* the project file can be specified via `@{'Project'='C:\temp\test.ipj'}`.
Additionally all the options of the API method OpenWithOptions can be used, eg `@{'Fast Open'=$true; 'ModelState'='Master'}`:

Type	Name	Description	Default
string / System.IO.FileIr	Project	The ProjectFile that Inventor uses to open the document. Possible: `'C:\somefile.ipj'`, `[file]'C:\somefile.Ipj'`	Current Inventor Project
bool / string	Visible	Specifies whether the document should be opened visible or invisible. Possible: `$true`,`'False'`,`'TRUE'`	True
string	DesignViewRepresentation	Only for assemblies and parts	For assemblies the first DesignViewRepresentations of the document
string	PositionalRepresentation	Only for assemblies	
string	ModelState	Only for assemblies and parts.	The 'Master' model state
bool	DeferUpdates	Only for drawings	
FileVersionEnum	FileVersionOption		
bool	ImportNonInventorDwg		False
string	Password		
ExpressModeBehavior	ExpressModeBehavior	Only for assemblies	
bool	SkipAllUnresolvedFiles		False
bool	DeferFlatPatternUpdate	Only for sheet metal parts	False
bool	FastOpen	Only for drawings	False

DWG TrueView

DWG TrueView has only the following options:

Type	Name	Description	Default
bool / string	Readonly	Specifies wether the document should be opened in readonly mode or not. Possible: `$true`,`'False'`,`'TRUE'`	False

Examples

Opening an AutoCAD DWG file

```
$result = Open-Document -LocalFile 'C:\Vault\AutoCad\Drawings\Tablet.dwg'

$result.Application.Name
#Returns: DWG TrueView
```

Opening an AutoCAD DWG file with Inventor by forcing it

```
$result = Open-Document -LocalFile 'C:\Vault\AutoCad\Drawings\Tablet.dwg' -Application
↪'Inventor'

$result.Application.Name
#Returns: Inventor
```

Prefer opening Inventor documents with InventorServer by default instead of Inventor

Because *Inventor* and *InventorServer* can both open the same document formats, the *Open-Document* cmdlet will consider the order of the *application registrations* by default (see *Remarks*).
Therefore, if you prefer to use InventorServer as default over Inventor, then just switch the registration lines in the *coolOrange.Applications.psm1* module, so that InventorServer is registered first:

```
Register-Application ([powerJobs.Application.Inventor.Server.Application])
Register-Application ([powerJobs.Application.Inventor.Application])
```

```
$result = Open-Document -LocalFile 'C:\Vault\Inventor\Drawings\Pad Lock.idw'

$result.Application.Name
#Returns: InventorServer
```

Opening an Inventor drawing with specified project file and FastOpen

```
Open-Document -LocalFile 'C:\Vault\Inventor\Drawings\Pad Lock.idw' -Options @{
        'Project'='C:\Vault\Default.ipj';
        'FastOpen'=$true
}
```

Using the result to validate if document was opened successfully

```
$result = Open-Document C:\Temp\Test.ipt
if(-not $result) {
        throw "Failed with error: $($result.Error.Message)"
}
```

Using the result to access the Inventor API (application made visible for Debugging)

```
$result = Open-Document 'C:\Vault\Inventor\SheetMetal.ipt'

if($result.Application.Name -eq 'Inventor') {
        $result.Application.Instance.Visible = $true # makes the instrumented Inventor␣
↪window visible
        $result.Application.Instance.UserInterfaceManager.UserInteractionDisabled =
```

(continues on next page)

(continued from previous page)

```
↪$false # allows to click around in the window for testing purposes
}

# the following API accesses work for Inventor and InventorServer:
if ($result.Document.Instance.ComponentDefinition.Type -ne [Inventor.
↪ObjectTypeEnum]::kSheetMetalComponentDefinitionObject) {
        throw "The file 'SheetMetal.ipt' is not a sheet metal part!"
}
```

Name	Description
Open-Document	Opens the specified document in a registered powerJobs application.
Export-Document	Exports the document in the specified format.
Close-Document	Closes an open document.
Clean-Up	Removes files and folders from the FileSystem.

4.2.2 Objects

ChangeOrder

The ChangeOrder object is of type powerVault ChangeOrder and grants direct access to the Vault changeOrder where the Job is triggered on.

Syntax

```
$changeOrder._Number
```

See also:

For more informations see: powerVault ChangeOrder.

Remarks

The $changeOrder variable is available when a Job is triggered with the parameter:

- ChangeOrderId = <Id of a ChangeOrder>

Also the LifecycleEventEditor triggers the Job with this parameter

CustomObject

The CustomObject object is of type *Autodesk.DataManagement.Client.Framework.Vault.Currency.Entities.CustomObject* and grants direct access to the Vault custom Object where the Job is triggered on.

Syntax

```
$customObject.Definition
```

Following properties are available :

Type	Name	Description	Access type
	Category	Gets the category that is assigned to this Custom Object.	read-only
	Definition	Gets the Custom Object Definition which contains specifics about the type of custom object this is	read-only
	EntityClass	Gets the entity class associated with this CustomObject.	read-only
long	EntityIterationId	Gets the Iteration ID of this Change Order. Since Change Orders are not iteration based, this will always have the same value as the EntityMasterId.	read-only
long	EntityMasterId	Gets the Master Id of this Change Order. The Master Id uniquely identifies this Change Order object	read-only
string	EntityName	Gets the descriptive name for this Custom Object.	read-only
bool	IsCloaked	Gets if this Custom Object is cloaked. A cloaked object is one that the caller does not have permissions to view.	read-only
	LinkInfo	Gets if this is a Link to a Custom Object. If the value is not null, then LinkInfo provides information about the Link to this CustomObject.	read-only
	VaultConnection	Gets the associated Vault Connection for this ChangeOrder.	read-only

Remarks

The $customObject variable is available when a Job is triggered with these parameters:

- EntityClassId = CUSTENT or <internal GUID of the custom object definition>
- EntityId = <Id of the custom entity>

Also the LifecycleEventEditor triggers the Job with this parameters.

All the properties of the CustomObject object are part of the Vault API.
You can look them up in the VaultSDK documentation under *CustomObject Class Members*.

File

The File object is of type powerVault File and grants direct access to the Vault file where the Job is triggered on.
The $file object returns the *latest iteration* of the actual file in Vault.

Syntax

```
$file.'Full Path'
```

See also:

For more informations see: powerVault File.

Remarks

The $file variable is available when a Job is triggered with these parameters:

- EntityClassId = File
- EntityId = <Id of the file>

Also the powerJobs Vault addin and the LifecycleEventEditor trigger the Job with this parameters.

Unlike the VDF file object the powerJobs file object has members for things like **lifecycle state**, **category** and most importantly **properties**.
So if you want to know for example the Author of a file you just have to write `$file.Author`.

You don't have to care about Ids or file iterations.
You always work with the latest iteration and the powerJobs file object is an exact representation of the file in vault.
This saves a lot of code, because you don't have to get lifecycle objects or property objects and work through them with loops.

If you have very special use cases, where our Cmdlets are not flexible enough, you have to use functions from the VDF or directly the webservices. In that case, you can just convert the powerVault file object into an Autodesk file object by using the Id or MasterId property.

Folder

The Folder object is of type powerVault Folder and grants direct access to the Vault folder where the Job is triggered on.

Syntax

```
$folder.'Full Path'
```

See also:

For more informations see: powerVault Folder.

Remarks

The $folder variable is available when a Job is triggered with these parameters:

- EntityClassId = FLDR
- EntityId = <Id of the folder>

These parameters are also set by the LifecycleEventEditor to trigger jobs for Folders.

Host

The automatic variable $Host is of type PSHost and represents the current host application for PowerShell which in context of powerJobs this can be either the *JobHandler* or your *powershell IDE*.

The object also provides access to all the *Applications* that can be used by the Cmdlets *Open-Document*, *Export-Document* and *Close-Document*.

Syntax

```
$Host.Applications['Application Name']
```

See also:

For the complete list of properties and methods and for more informations see: PSHost Class.

Following properties are always available :

Type	Nam	Description	Access type
String	Name	An identification for the PowerShell hosting application.Within powerJobs Processor jobs this should be *'powerJobs JobHandler Extension'*.	read-only
IDictionary<st *IApplication*>	Applications	A dictionary of registered applications.A list of default applications can be found *here*.	read-only
PSObject	PrivateData	Each Host can provide private data that is editable. For example the property ErrorBackgroundColor that is available in several PowerShell IDEs for changing the background color of error messages.The powerJobs Processor Host allows extending the way how Vault users are notified about Terminating Errors using the `OnTerminatingError` property.	read-write

Remarks

Same as for the PowerShell IDE even the powerJobs Processor Host displays the written output to the user in a designated *Trace Window*.
The format of the output provided by the Write-Host cmdlet can be customized within the according *Logging* sections.

In addition to the **Applications** property available on the $Host variable, the *coolOrange.Applications.psm1* module provides additionally functions to simplify the access:

- `Get-Application` can be used to retrieve an application by name, e.g.:

```
$inventor = Get-Application 'Inventor'
```

- `Test-ApplicationInstalled` can be used to check whether the required application is installed on the Job Processor machine:

```
if(-not Test-ApplicationInstalled 'Inventor') {
  throw("Inventor is not installed on this machine!")
}
```

Automatic Restart

PowerJobs Processor handles the starting and stopping of the registered applications due these cases:

- timer interval
- limit of close documents

Timer interval:
Each application has as timer interval, by default this setting is set to **30 minutes**.
This settings can be changed in the *Setup_job.ps1* file for a specific application:

```
$inventor = Get-Application 'Inventor'
$inventor.RestartOptions.Interval = [timespan]::FromSeconds(60)
```

The **timer starts** with the configured interval, when the first *open-document* call is made.
When the timer expires and the application is not busy (has no open documents), the application will be restarted immediately. Otherwise the restart for the application will be reserved.
This means that on the next Open-Document call that uses this application, it will be restarted. Watch out, again: only if the application has no open documents!

Limit of close documents:
Each application has a limit of close documents, the default value is set to **10**.
This settings can be changed in the *Setup_job.ps1* file for a specific application:

```
$inventor = Get-Application 'Inventor'
$inventor.RestartOptions.DocumentCloseCounts = 10
```

Every time a document gets closed this documentCloseCount increments as long as it reaches the configured limit. Then, the restart of this application will be reserved.
This means on the next *Open-Document* call that uses this application, it will be restarted.

Only if the applications has no open documents! Otherwise the restart remains reserved as long as a restart is allowed.

The restarts will be reserved, in order to not restart itself and close the open documents.
This could cause issues, when opening many documents one after another and the timer expires, or the closeCounter is reached.

Error Notifications

By default the powerJobs Processor host is configured to notify *about failing jobs* by writing exception details to the Job Queue Result as well as to the Trace Window and logfile.

The way how Vault users are notified about terminating errors in job scripts can be extended using the **PrivateData** property and its `OnTerminatingError` property that provides all the relevant error details:

```
$global:Host.PrivateData.OnTerminatingError = [System.Delegate]::Combine([Action[System.
↪Management.Automation.RuntimeException]] {
        param($terminatingError)
        show-inspector 'terminatingError'
}, $global:Host.PrivateData.OnTerminatingError)
```

IAmRunningInJobProcessor

The IAmRunningInJobProcessor object is of type *bool* and tells whether the current script gets executed in JobProcessor or not.

Syntax

```
$IAmRunningInJobProcessor
```

Remarks

The $IAmRunningInJobProcessor flag returns only $true when the Job is executed in the JobProcessor.
When running the script in a PowerShell IDE the variable returns $false.

Item

The Item object is of type powerVault Item and grants direct access to the Vault item where the Job is triggered on.

Syntax

```
$item
```

See also:

For more informations see: powerVault Item.

Remarks

The $item variable is available when a Job is triggered with these parameters:

- EntityClassId = ITEM
- EntityId = <Id of the item>

Also the LifecycleEventEditor triggers the Job with this parameters

Job

The Job object is of type powerVault Job and grants direct access to the Vault Job that is currently executing.

Syntax

```
$job.Name
```

See also:

For more informations see: powerVault Job

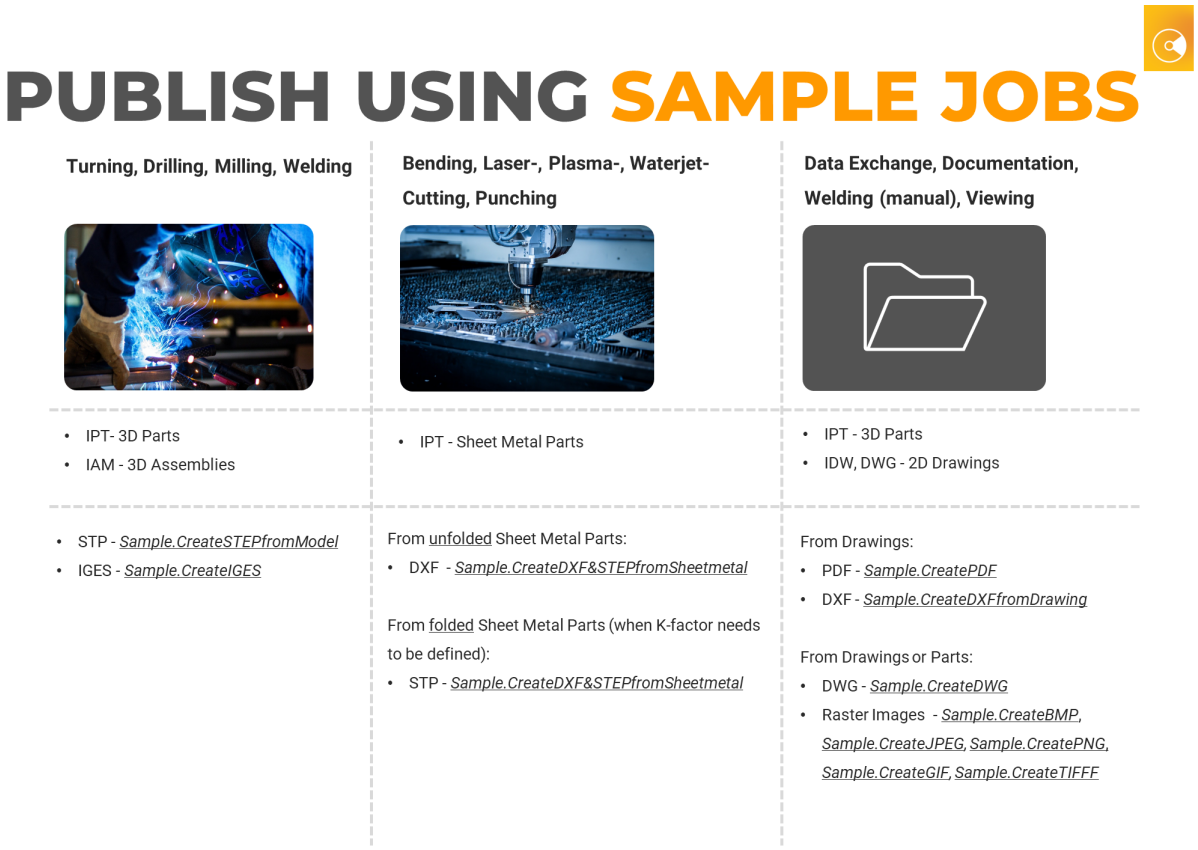
- *IAmRunningInJobProcessor*
- *Host*
- *Job*

Depending for which entity the job is triggered, the following objects are available:

- *File*
- *Folder*
- *Customobject*
- *Item*
- *Changeorder*

4.3 File Conversion

Out of the box powerJobs Processor installs a *sample job* for most supported formats with the name 'Sample.Create*FORMAT*.ps1' that converts a file to that format.
Overview of which sample job can be used for which purpose or format:



All the jobs are using the *Export-Document* cmdlet for the conversion as this cmdlet supports all the formats listed below.

4.3.1 Supported Format Conversions

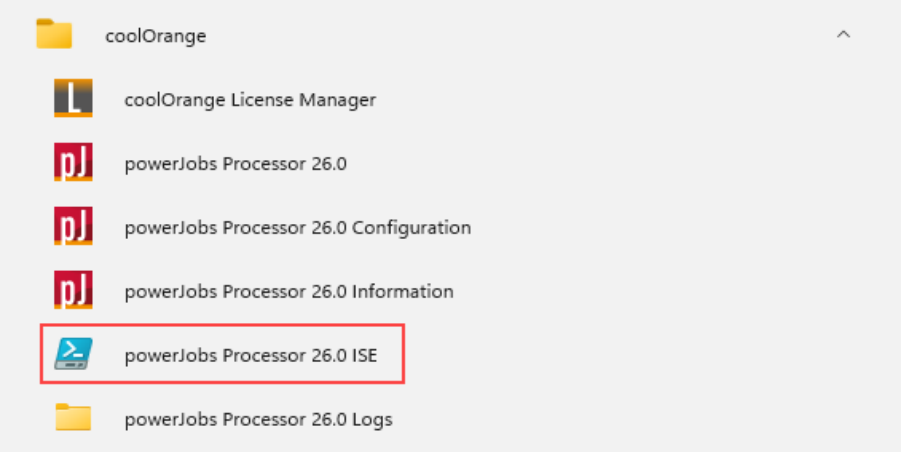
The product supports some *CAD applications* (Inventor, InventorServer and DWG TrueView) and many conversions out-of-the-box.
See all exports marked with a '✓' sign in the following table:

	PDF	DWF / DWFX	DXF	DWG	IGES	STEP	JPEG	BMP	GIF	TIFF	PNG
Inventor:											
IAM - 3D Assemblies	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓
IPT - 3D Parts	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓
IPT - Sheet Metal Parts		✓	✓	✓	✓	✓					
IDW - 2D Drawings	✓	✓	✓	✓			✓	✓	✓	✓	✓
DWG - 2D Drawings	✓	✓	✓	✓			✓	✓	✓	✓	✓
IPN - Presentation files	✓	✓					✓	✓	✓	✓	✓
AutoCAD:											
DWG - 2D Drawings	✓	✓	✓	✓							
Valid extensions:	.pdf	.dwf / .dwfx	.dxf	.dwg	.iges / .igs	.stp / .step	.jpg	.bmp	.gif	.tiff	.png

Tip: You can customize the default functionality of powerJobs Processor to your needs.
Get started with *Creating Job Scripts* and if necessary find more details and instructions how a *custom application* can be implemented.

4.4 Jobs

4.4.1 Creating Job Scripts



Choose PowerShell IDE

To get started with creating a custom job, open the PowerShell IDE or editor of choice. The *powerJobs Processor ISE* shortcut in the start menu can also be used which already opens one of our *Sample Jobs*.

Prepare PowerShell environment

During development it can be useful to be able to start the whole script or parts of it without having to run it on the job processor.
To achieve this, some important steps - which are *performed automatically in the Job Processor* - must be carried out manually:

A connection to Vault must be established with the *Open-VaultConnection* cmdlet so that the variables `$vault`, `$vaultConnection` and `$vaultExplorerUtil` variables available.

Now a test file or any other Vault entity can be retrieved and assigned to a variable just like the *Setup_job.ps1* would do in the Job Processor.
For example set the `$file` variable to the first *.idw* file that is found in Vault:

```
$file = Get-VaultFile -Properties @{'File Extension'='idw'} #Search for a Drawing file
```

Any modules placed in the *powerJobs modules directory* are typically auto-imported as soon as the powerJobs module is used.
To make them available immediately, you can also manually import them by calling `Import-Module powerJobs`.

The variable `$workingDirectory` holds the path to a temporary directory (located in *C:\Temp\powerJobs Processor...*) where e.g. Vault files can be downloaded or file exports will be generated.
All the files which are created in this directory during the development phase, will be automatically cleaned up at the next job execution.
Alternatively, they can be removed at any time using the *Clean-Up* cmdlet.

```
Clean-Up -folder $workingDirectory
```

Choose from Sample Jobs

Our standard delivery comes with a couple of *Sample Jobs* that can be used as templates for new jobs.
Choose one of those job scripts and place a copy of the script in the *%ALLUSERSPROFILE%\coolOrange\powerJobs\Jobs* directory.
Give your new Job a meaningful name in the format *YourCompanyName.SomeJobName.ps1*.

If you prefer to do customizations, for example some sample jobs don't allow all supported formats, the new script can be extend by adding the desired extensions like *iam*, *ipt* or *ipn*:

```
if( @("idw","dwg","iam","ipt","ipn") -notcontains $file._Extension ) {
        Write-Host "Files with extension: '$($file._Extension)' are not supported"
        return
}
```

An other example would be to changed the name of the exported target file.

In this example the according filename variable is extended with the revision number of the Vault source file:

```
$pdfFileName = "$($file._Name)_$($file._Revision).pdf"
```

Start from Scratch

Create a new script file with a meaningful name in the IDE and place it in *%ALLUSERSPROFILE%\coolOrange\powerJobs\Jobs* with the file extension *.ps1*.
In this example we are going to download the file, open it in the Inventor application and export it as PDF.

Download the file from Vault

We will use the Save-VaultFile cmdlet to download the file to our working directory.

```
$downloadedFiles = Save-VaultFile -File $file.'Full Path' -DownloadDirectory
↪$workingDirectory
```

Open the file in Inventor

After downloading the file we will use the *open-document* cmdlet to open the file in Inventor.

```
$file = $downloadedFiles | select -First 1
Open-Document -LocalFile $file.LocalPath -Application 'Inventor'
```

Exporting the file

Now we can export the document as PDF, using the *export-document* cmdlet.

```
$pdfFileName = "$($file._Name).pdf"
Export-Document -Format 'PDF' -To "$workingDirectory/$pdfFileName"
```

Closing the opened file

Finally we close the file using the *Close-Document* cmdlet.

```
Close-Document
```

To run the job on the job processor see the *Registered Jobs* section and *Queueing custom jobs*.
If values in the job regularly change, it is also possible to declare *settings* in the job that can be easily modified in the *powerJobs Settings Dialog* user interface.

4.4.2 Sample Jobs

All jobs that come with the product are starting with the name '*Sample.*'. Their purpose is mainly to help getting started with creating their custom jobs and to be useful and easy to configure for some common scenarios. It is recommended to copy these jobs and *create new ones* when they need to be customized.

All sample jobs contain a declaration of their *supported entity type*.

To queue sample jobs see *queueing custom jobs* section. When a sample job is queue for a not supported file it will write a message to the log but will not fail. All the sample jobs which support exports from Inventor Drawings can make use of the Inventor FastOpen feature. This means only the drawing without dependencies gets downloaded and *opened*.

Default job settings

Each sample job has *configurable settings* to control its behavior. The following settings are not included in the *settings region* of the sample jobs and are therefore not configurable:

The variable `$[type]fileName` - where *[type]* is the extension of your target file - holds the name of the file to export. It is set to the name of the Vault source file with an additional '*.[type]*' suffix. For example, when queuing the *Sample.CreatePDF job* for the Inventor Drawing *Pad Lock.idw*, the value of the variable would be *Pad Lock.idw.**pdf***.

The variable `$workingDirectory` holds the path to a temporary directory. In the sample jobs it is used as temporary direcotry where Vault files are downloaded or file exports are generated. At the end of each job it will be purged automatically.

Default state change triggers

The following sample jobs are delivered with default settings files for *Lifecycle State Change Triggers* which can be configured in the *Settings Dialog*.

- *Sample.CreatePDF*
- *Sample.CreateDXF&STEPfromSheetmetal*

Sample.CreatePDF

Supported entity type: *FILE* **Supported files:** *idw, dwg (Inventor and AutoCAD)* **Configuration file:**

- Inventor drawings: *PDF_2D.ini*
- AutoCAD Drawings: *PDF.dwg*

By default, the Job creates a *PDF* and adds it to the Vault as a *DesignVisualization*. The PDF file is stored in the same location and with the same file name as the drawing and attached to it.

Settings Important settings like the file name for the output format, if and where the file should be added to Vault, whether the file should be published to a network share, and other settings are specified in the first lines of the job so that they can be easily changed:

```
#region Settings
# To include the Revision of the main file in the PDF name set Yes, otherwise No
$pdfFileNameWithRevision = $false

# The character used to separate file name and Revision label in the PDF name such as␣
↪hyphen (-) or underscore (_)
$pdfFileNameRevisionSeparator = "_"

# To include the file extension of the main file in the PDF name set Yes, otherwise No
$pdfFileNameWithExtension = $true

# To add the PDF to Vault set Yes, to keep it out set No
$addPDFToVault = $true

# To attach the PDF to the main file set Yes, otherwise No
$attachPDFToVaultFile = $true

# Specify a Vault folder in which the PDF should be stored (e.g. $/Designs/PDF), or␣
↪leave the setting empty to store the PDF next to the main file
$pdfVaultFolder = ""

# Specify a network share into which the PDF should be copied (e.g. \\SERVERNAME\Share\
↪Public\PDFs\)
```

(continues on next page)

(continued from previous page)

```
$pdfNetworkFolder = ""

# To enable faster opening of released Inventor drawings without downloading and opening␣
↪their model files set Yes, otherwise No
$openReleasedDrawingsFast = $true
#endregion
```

Default state change triggers This job comes with a *default file* (*Sample.CreatePDF.defaults*) which defines the following settings:

- Filter to only trigger for drawings (files with extension `idw` or `dwg`)
- Queue the job with priority *101*

Sample.CreateDXF&STEPfromSheetmetal

Supported entity type: *FILE* **Supported files:** *ipt (sheet metal)* **Configuration file:**

- For DXF: *DXF_SheetMetal.ini*
- For STEP: *STEP.ini*

By default, the Job creates a *DXF* file and a *STEP* file that are added to the Vault as *DesignVisualization* files. The DXF and STEP files are stored in the same location and with the same file name as the sheet metal part and attached to it. Usually the DXF is used for manufacturing (laser cutter, bending machine, etc…), while the STEP is used as 3D model for suppliers as it can be better used with the K-Factor of the machines.

Settings Important settings like the file name for the output format, if and where the file should be added to Vault, if only the DXF or STEP should be created, whether the file should be published to a network share, and other settings are specified in the first lines of the job so that they can be easily changed:

```
#region Settings
# To include the Revision of the sheet metal part in the DXF name set Yes, otherwise No
$dxfFileNameWithRevision = $false

# The character used to separate file name and Revision label in the DXF name such as␣
↪hyphen (-) or underscore (_)
$dxfFileNameRevisionSeparator = "_"

# To include the file extension '.ipt' in the DXF name set Yes, otherwise No
$dxfFileNameWithExtension = $true

# To add the DXF to Vault set Yes, to keep it out set No
$addDXFToVault = $true

# To attach the DXF to the sheet metal part set Yes, otherwise No
$attachDXFToVaultFile = $true

# Specify a Vault folder in which the DXF should be stored (e.g. $/Designs/DXF), or␣
↪leave the setting empty to store the DXF next to the sheet metal part
$dxfVaultFolder = ""

# Specify a network share into which the DXF should be copied (e.g. \\SERVERNAME\Share\
↪Public\DXFs\)
```

(continues on next page)

(continued from previous page)

```
$dxfNetworkFolder = ""

# To include the Revision of the sheet metal part in the STEP name set Yes, otherwise No
$stepFileNameWithRevision = $false

# The character used to separate file name and Revision label in the STEP name such as␣
↪hyphen (-) or underscore (_)
$stepFileNameRevisionSeparator = "_"

# To include the file extension '.ipt' in the STEP name set Yes, otherwise No
$stepFileNameWithExtension = $true

# To add the STEP to Vault set Yes, to keep it out set No
$addSTEPToVault = $true

# To attach the STEP to the sheet metal part set Yes, otherwise No
$attachSTEPToVaultFile = $true

# Specify a Vault folder in which the STEP should be stored (e.g. $/Designs/STEP), or␣
↪leave the setting empty to store the STEP next to the sheet metal part
$stepVaultFolder = ""

# Specify a network share into which the STEP should be copied (e.g. \\SERVERNAME\Share\
↪Public\STEPs\)
$stepNetworkFolder = ""
#endregion
```

Default state change triggers This job comes with a *default file* (*Sample.CreateDXF&STEPfromSheetmetal.defaults*) which defines the following settings:

- Filter to only trigger for parts (files with extension `ipt`)
- Queue the job with priority *101*

The client customizations SubmitJobsOnLifecycleTransition and SubmitJobsOnVaultMenuItemClick also validate that this job is only queued for Sheetmetal parts.

Sample.CreateDXFfromDrawing

Supported entity type: *FILE* **Supported files:** *idw, dwg (Inventor and AutoCAD)* **Configuration file:** *DXF_2D.ini*

By default, the Job creates a *DXF* with  InventorServer and adds it to the Vault as a *DesignVisualization*. For **multisheet idw or dwg** files depending on configuration either a .zip file is created or a .dxf file for each sheet. The file(s) are stored in the same location and with the same file name as the source file and are attached to it.

Settings Important settings like the file name for the output format, if- and where the file should be added to Vault, whether the file should be published to a network share, and settings for troubleshooting are specified in the first lines of the job so that they can be easily changed:

```
#region Settings
# To include the Revision of the main file in the DXF name set Yes, otherwise No
$dxfFileNameWithRevision = $false

# The character used to separate file name and Revision label in the DXF name such as␣
```

(continues on next page)

(continued from previous page)

```
↪hyphen (-) or underscore (_)
$dxfFileNameRevisionSeparator = "_"

# To include the file extension of the main file in the DXF name set Yes, otherwise No
$dxfFileNameWithExtension = $true

# To add the DXF to Vault set Yes, to keep it out set No
$addDXFToVault = $true

# To attach the DXF to the main file set Yes, otherwise No
$attachDXFToVaultFile = $true

# Specify a Vault folder in which the DXF should be stored (e.g. $/Designs/DXF), or␣
↪leave the setting empty to store the DXF next to the main file
$dxfVaultFolder = ""

# Specify a network share into which the DXF should be copied (e.g. \\SERVERNAME\Share\
↪Public\DXFs\)
$dxfNetworkFolder = ""

# To enable faster opening of released Inventor drawings without downloading and opening␣
↪their model files set Yes, otherwise No
$openReleasedDrawingsFast = $true

# To enable faster processing via InventorServer set Yes, otherwise set No if an␣
↪Inventor installation and license are available
$useInventorServer = $true
#endregion
```

Sample.CreateSTEPfromModel

Supported entity type: *FILE* **Supported files:** *iam, ipt* **Configuration file:** *STEP.ini*

By default, the Job creates *STEP* files and adds them to Vault as a *DesignVisualization*. The STEP files are stored in the same location and with the same file name as the source file and are attached to it.

Settings Important settings like the file name for the output format, if and where the file should be added to Vault, whether the file should be published to a network share, and other settings are specified in the first lines of the job so that they can be easily changed:

```
#region Settings
# To include the Revision of the main file in the STEP name set Yes, otherwise No
$stepFileNameWithRevision = $false

# The character used to separate file name and Revision label in the STEP name such as␣
↪hyphen (-) or underscore (_)
$stepFileNameRevisionSeparator = "_"

# To include the file extension of the main file in the STEP name set Yes, otherwise No
$stepFileNameWithExtension = $true

# To add the STEP to Vault set Yes, to keep it out set No
```

(continues on next page)

(continued from previous page)

```
$addSTEPToVault = $true

# To attach the STEP to the main file set Yes, otherwise No
$attachSTEPToVaultFile = $true

# Specify a Vault folder in which the STEP should be stored (e.g. $/Designs/STEP), or␣
↪leave the setting empty to store the STEP next to the main file
$stepVaultFolder = ""

# Specify a network share into which the STEP should be copied (e.g. \\SERVERNAME\Share\
↪Public\STEPs\)
$stepNetworkFolder = ""
#endregion
```

Sample.CreateDWG

Supported entity type: *FILE* **Supported files:** *ipt (incl. sheet metal), iam , idw, dwg (Inventor and AutoCAD)* **Configuration file:**

- Sheet metal: *DWG_SheetMetal.ini*
- Drawings: *DWG_2D.ini*
- Models: *DWG_3D.ini*

By default, the Job creates *AutoCAD 2000 DWG* with  InventorServer and adds it to the Vault as a *DesignVisualization*. For **multisheet idw or dwg** depending on configuration either a .zip file is created or a .dwg file for each sheet. The Drawings files are stored in the same location and with the same file name as the source file and are attached to it.

Settings Important settings such as the file name for the output format, wether- and where to add the file to Vault and settings for troubleshooting are specified in the first lines of the job so that they can be easily changed:

```
#region Settings
# To include the Revision of the main file in the DWG name set Yes, otherwise No
$dwgFileNameWithRevision = $false

# The character used to separate file name and Revision label in the DWG name such as␣
↪hyphen (-) or underscore (_)
$dwgFileNameRevisionSeparator = "_"

# To include the file extension of the main file in the DWG name set Yes, otherwise No
$dwgFileNameWithExtension = $true

# Specify a Vault folder in which the DWG should be stored (e.g. $/Designs/DWG), or␣
↪leave the setting empty to store the DWG next to the main file
$dwgVaultFolder = ""

# To enable faster opening of released Inventor drawings without downloading and opening␣
↪their model files set Yes, otherwise No
$openReleasedDrawingsFast = $true

# To create DWGs out of AutoCAD source files set Yes, otherwise set No if only Inventor␣
↪files should be processed
```

(continues on next page)

(continued from previous page)

```
$processAutoCAD = $true

# To enable faster processing via InventorServer set Yes, otherwise set No if an␣
↪Inventor installation and license are available
$useInventorServer = $true
#endregion
```

Sample.CreateIGES

Supported entity type: *FILE* **Supported files:** *iam, ipt* **Configuration file:** *IGES.ini*

By default, the Job creates *IGES* files and adds them to Vault as a *DesignVisualization*. The IGES files are stored in the same location and with the same file name as the source file and are attached to it.

Settings Important settings such as the file name for the output format, whether and where to add the file to Vault, and other settings can be easily changed:

```
#region Settings
# To include the Revision of the main file in the IGES name set Yes, otherwise No
$igesFileNameWithRevision = $false

# The character used to separate file name and Revision label in the IGES name such as␣
↪hyphen (-) or underscore (_)
$igesFileNameRevisionSeparator = "_"

# To include the file extension of the main file in the IGES name set Yes, otherwise No
$igesFileNameWithExtension = $true

# Specify a Vault folder in which the IGES should be stored (e.g. $/Designs/IGES), or␣
↪leave the setting empty to store the IGES next to the main file
$igesVaultFolder = ""
#endregion
```

Sample.CreateJPEG

Supported entity type: *FILE* **Supported files:** *iam, ipt, ipn, idw, dwg (Inventor)*

By default, the Job creates *JPEG* files and adds them to Vault as a *DesignVisualization*. The JPEG file is stored in the same location and with the same file name as the source file and is attached to it.

Settings Important settings such as the file name for the output format, whether and where to add the file to Vault, and other settings can be easily changed:

```
#region Settings
# To include the Revision of the main file in the JPG name set Yes, otherwise No
$jpgFileNameWithRevision = $false

# The character used to separate file name and Revision label in the JPG name such as␣
↪hyphen (-) or underscore (_)
$jpgFileNameRevisionSeparator = "_"

# To include the file extension of the main file in the JPG name set Yes, otherwise No
```

(continues on next page)

(continued from previous page)

```
$jpgFileNameWithExtension = $true

# Specify a Vault folder in which the JPG should be stored (e.g. $/Designs/JPG), or␣
↪leave the setting empty to store the JPG next to the main file
$jpgVaultFolder = ""

# To enable faster opening of released Inventor drawings without downloading and opening␣
↪their model files set Yes, otherwise No
$openReleasedDrawingsFast = $true
#endregion
```

Sample.CreateGIF

Supported entity type: *FILE* **Supported files:** *iam, ipt, ipn, idw, dwg (Inventor)*

By default, the Job creates *GIF* files and adds them to Vault as a *DesignVisualization*. The GIF file is stored in the same location and with the same file name as the source file and is attached to it.

Settings Important settings such as the file name for the output format, whether and where to add the file to Vault, and other settings can be easily changed:

```
#region Settings
# To include the Revision of the main file in the GIF name set Yes, otherwise No
$gifFileNameWithRevision = $false

# The character used to separate file name and Revision label in the GIF name such as␣
↪hyphen (-) or underscore (_)
$gifFileNameRevisionSeparator = "_"

# To include the file extension of the main file in the GIF name set Yes, otherwise No
$gifFileNameWithExtension = $true

# Specify a Vault folder in which the GIF should be stored (e.g. $/Designs/GIF), or␣
↪leave the setting empty to store the GIF next to the main file
$gifVaultFolder = ""

# To enable faster opening of released Inventor drawings without downloading and opening␣
↪their model files set Yes, otherwise No
$openReleasedDrawingsFast = $true
#endregion
```

Sample.CreatePNG

Supported entity type: *FILE* **Supported files:** *iam, ipt, ipn, idw, dwg (Inventor)*

By default, the Job creates *PNG* files and adds them to Vault as a *DesignVisualization*. The PNG file is stored in the same location and with the same file name as the source file and is attached to it.

Settings Important settings such as the file name for the output format, whether and where to add the file to Vault, and other settings can be easily changed:

```
#region Settings
# To include the Revision of the main file in the PNG name set Yes, otherwise No
$pngFileNameWithRevision = $false

# The character used to separate file name and Revision label in the PNG name such as␣
↪hyphen (-) or underscore (_)
$pngFileNameRevisionSeparator = "_"

# To add the PNG to Vault set Yes, to keep it out set No
$pngFileNameWithExtension = $true

# Specify a Vault folder in which the PNG should be stored (e.g. $/Designs/PNG), or␣
↪leave the setting empty to store the PNG next to the main file
$pngVaultFolder = ""

# To enable faster opening of released Inventor drawings without downloading and opening␣
↪their model files set Yes, otherwise No
$openReleasedDrawingsFast = $true
#endregion
```

Sample.CreateTIFF

Supported entity type: *FILE* **Supported files:** *iam, ipt, ipn, idw, dwg (Inventor)*

By default, the Job creates *TIFF* files and adds them to Vault as a *DesignVisualization*. The TIFF file is stored in the same location and with the same file name as the source file and is attached to it.

Settings Important settings such as the file name for the output format, whether and where to add the file to Vault, and other settings can be easily changed:

```
#region Settings
# To include the Revision of the main file in the TIFF name set Yes, otherwise No
$tiffFileNameWithRevision = $false

# The character used to separate file name and Revision label in the TIFF name such as␣
↪hyphen (-) or underscore (_)
$tiffFileNameRevisionSeparator = "_"

# To include the file extension of the main file in the TIFF name set Yes, otherwise No
$tiffFileNameWithExtension = $true

# Specify a Vault folder in which the TIFF should be stored (e.g. $/Designs/TIFF), or␣
↪leave the setting empty to store the TIFF next to the main file
$tiffVaultFolder = ""

# To enable faster opening of released Inventor drawings without downloading and opening␣
↪their model files set Yes, otherwise No
$openReleasedDrawingsFast = $true
#endregion
```

Sample.CreateBMP

Supported entity type: *FILE* **Supported files:** *iam, ipt, idw, dwg (Inventor)*

By default, the Job creates *BMP* files and adds them to Vault as a *DesignVisualization*. The BMP file is stored in the same location and with the same file name as the source file and is attached to it.

Settings Important settings such as the file name for the output format, whether and where to add the file to Vault, and other settings can be easily changed:

```
#region Settings
# To include the Revision of the main file in the BMP name set Yes, otherwise No
$bmpFileNameWithRevision = $false

# The character used to separate file name and Revision label in the BPM name such as␣
↪hyphen (-) or underscore (_)
$bmpFileNameRevisionSeparator = "_"

# To include the file extension of the main file in the BMP name set Yes, otherwise No
$bmpFileNameWithExtension = $true

# Specify a Vault folder in which the BMP should be stored (e.g. $/Designs/BMP), or␣
↪leave the setting empty to store the BMP next to the main file
$bmpVaultFolder = ""

# To enable faster opening of released Inventor drawings without downloading and opening␣
↪their model files set Yes, otherwise No
$openReleasedDrawingsFast = $true
#endregion
```

4.4.3 Job Environment

The PowerShell environment in which the *PowerShell job scripts* are executed is pre-configured.
Before a job script is executed the following actions are automatically performed:

- All required PowerShell modules such as *powerJobs*, *powerVault* and those in the *modules directory*, are imported.
- `$workingDirectory` variable of type DirectoryInfo specifies foreach job a unique temporary directory, located under *C:\Temp\powerJobs Processor*.
- *$job* variable of type powerVault Job is provided that represents the job that is currently being executed.
- *$host* variable is extended with an *Applications* property that contains all registered *Applications*.
- *Setup_Job.ps1* script is invoked.

Setup_Job.ps1

The *Setup_Job.ps1* file is a special script that is always executed **before a job is executed**.
It is recommended to only modify this script for actions that need to be performed before **every** job execution regardless of job type.
It is located in the products ProgramData directory *%ProgramData%\coolOrange\powerJobs\Setup_Job.ps1*

This script is responsible for setting up the following global variables:

- *$ErrorActionPreference*
- *$IAmRunningInJobProcessor*

Depending on the *entity* the Job was triggered for, the following objects are automatically created using the global *$job* variable:

- *$file*
- *$folder*
- *$customObject*
- *$item*
- *$changeOrder*

With every job execution, temporary files from previous jobs are automatically removed from the working directory by using the *Clean-Up* cmdlet.

$ErrorActionPreference

When a script is executing in *powershell.exe*, the default behavior is to continue the execution of the script when an error occurs.
powerJobs handles **Errors in scripts** differently: the job execution is Stopped when an *exception is thrown*!

This default behaviour can be changed by using the *$ErrorActionPreference* variable:

```
$ErrorActionPreference = "Continue"
```

All the possible options are:

Identifier	Description
Continue	This is the default PowerShell setting. The error object is written to the output pipe and added to $error, and $? is set to false.Execution then continues at the next script line.
SilentlyContinue	When this action preference is set, the error message is not written to the output pipe before continuing execution. Note that it is still added to $error and $? is still set to false.Again, execution continues at the next line.
Stop	This error action preference changes an error from a non-terminating error to a terminating error. The error object is thrown as an exception instead of being written to the output pipe. $error and $? are still updated.Execution does not continue.
Inquire	Prompts the user requesting confirmation before continuing on with the operation.At the prompt, the user can choose to continue, stop or suspend the operation.**Warning:** It is not recommended to use this option with powerJobs Processor!

powerJobs Processor extends the Vault Job Processor and allows the creation of custom job types in form of PowerShell scripts.

These Scripts and Modules are then executed in a prepared environment by the job processor when a job of that type is queued.
The *PowerShell environment* is partly prepared by the *Setup_Job script* which can also be customized.
These jobs can be used to perform various task.

4.4.4 Job Scripts

All Job scripts need to be valid PowerShell scripts with the file extension *.ps1* that are stored in the directory *%ProgramData%\coolOrange\powerJobs\Jobs*
This directory contains a few *Sample Jobs* for most supported *File Conversions* using the provided *Cmdlets*.
New job scripts can be created or placed in the Jobs directory but they need to be *registered* before they can be executed by the job processor.

Job Entity type

A job can be associated to a Vault entity like *Files* or *Items*.

The declaration of the entity type for a job is a pre-requisite for using the *powerJobs Settings Dialog*. Each job should be designed to support only a **single entity type** which must be specified in the according script file of the job. The entity type can not be edited in the settings dialog.

To assign a job to the entity type '*File*', place the following Powershell comment in any line of the job script:

```
# JobEntityType = FILE
```

Possible entity types are:

Entity type	Description
FILE	Files
FLDR	Folders
ITEM	Items
CO	Change Orders
{Custom Object Definition Name}	For specific Custom Object Definitions *(e.g. Task, Contact)*

Note: The definition, that one job should be desgined only for a single entity type and the subsequent assignment of the according entity type to the job has currently only impacts on the functionality of the ***powerJobs Settings Dialog***.

Job Settings

Each job script can contain a one section that declares settings for the job.
The beginning of this section is marked by `#region Settings` and the end by `#endregion`.

All the PowerShell variables which are declared in this region can be easily configured in the *powerJobs Settings Dialog*.
If values are declared for those, then these are the default values of the displayed settings.
They can simply be edited without PowerShell experience if they are declared with simple data types:

- **Switch** settings are displayed for Boolean values like `$true` or `$false`.
- **Text** settings must be declared as single-line String variables. Their value must be enclosed in double quotation marks (`"..."`) or single quotation marks (`'...'`).
  Text strings which contain such quote characters are stored with second consecutive quote characters.

- **Powershell** settings are represented by text strings that begin with a dollar sign ($), such as `$a`, `$my_var` or `$object.property`.

A simple settings section could look like the following:

```
#region Settings
$pdfExportLocation = "C:\Exports\PDFs\"

$hidePDFinVault = $false

$myComputerName = $env:computername

$pdfNetworkFolder
#endregion
```

In order to provide more detailed information about a setting, a PowerShell comment can be specified in the line before the variable declaration.
The description is displayed alongside the setting in the *powerJobs Settings Dialog*.

```
#region Settings

# Path to the directory the PDF files should be exported to
$pdfExportLocation = "C:\Exports\PDFs\"

#endregion
```

More information on *Creating custom Job Scripts*.

Note: If changes are made to a script while the Job Processor is running, the changes will be recognized the next time the job is executed.

4.4.5 Modules

Module files need to be valid *PowerShell script modules* with the file extension *.psm1*. It is recommended to give the Modules a descriptive name, for instance *myCompany.NetworkStorage.psm1*.
All Module files are located in %ProgramData%\coolOrange\powerJobs\Modules and are automatically imported for jobs executed by the Job Processor and when the *powerJobs* module is imported.
Following modules are delivered with the product:

- coolOrange.Applications.psm1
- coolOrange.FileSystem.psm1

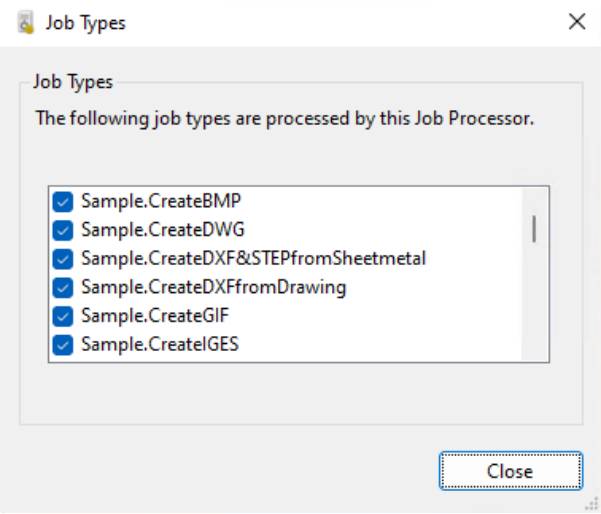
coolOrange.FileSystem

Contains functions to work with the computers FileSystem.
One important function is the *Clean-up* for cleaning up folders downloaded and generated files from your FileSystem after executing the job.

coolOrange.Applications

Contains functions to simplify the access to the Applications used by the powerJobs Processor.
These functions are just wrapping calls to the *$Host.Applications* object in order to use it in a more PowerShell fashioned way e.g *Get-Application*.

Registered Job Types



Each Job Processor only handles jobs of specific types that are assigned to that instance. To see which types are enabled, open the "Job Types" dialog, which can be found in the "Administration" menu of the Job Processor.

The dialog displays all registered job types for this instance and whether they are processed or not.
To register a new job type the *automatic synchronization* of the *Extended Job Processor UI* can be used. Alternatively it is also possible to register the job type manually by extending the *PowerJobs Processor.vcet.config* file.
If there is an entry for a job type, but it is not checked, the Job Processor most likely cannot find the associated job script (*.ps1* file) or it is not valid.

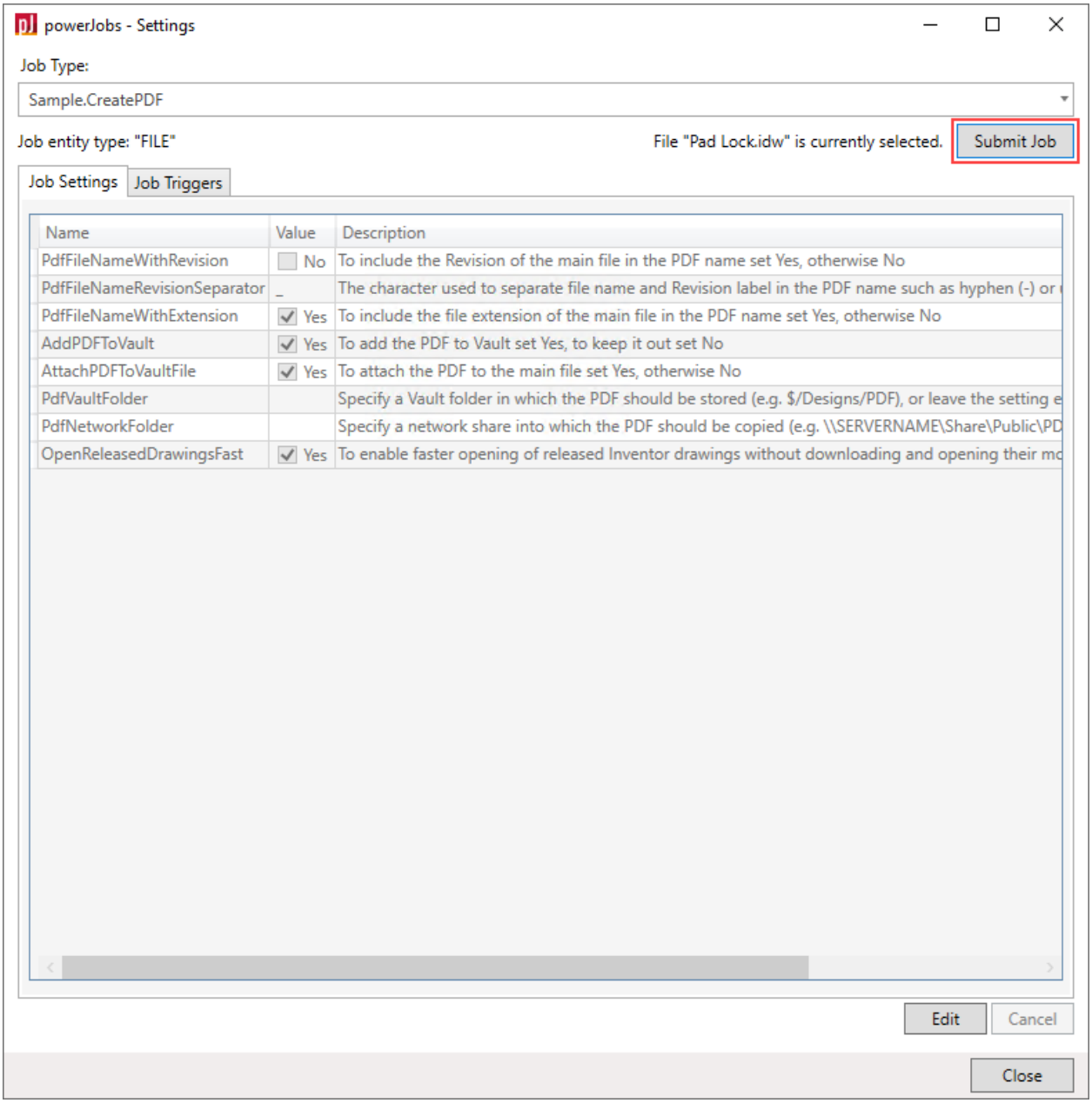
> **Warning:** Adding or removing registered job types always requires restarting the Job Processor for the changes to apply.

4.4.6 Queueing Jobs

Depending on the requirements, there are different options to queue custom job types:

powerJobs Settings Dialog

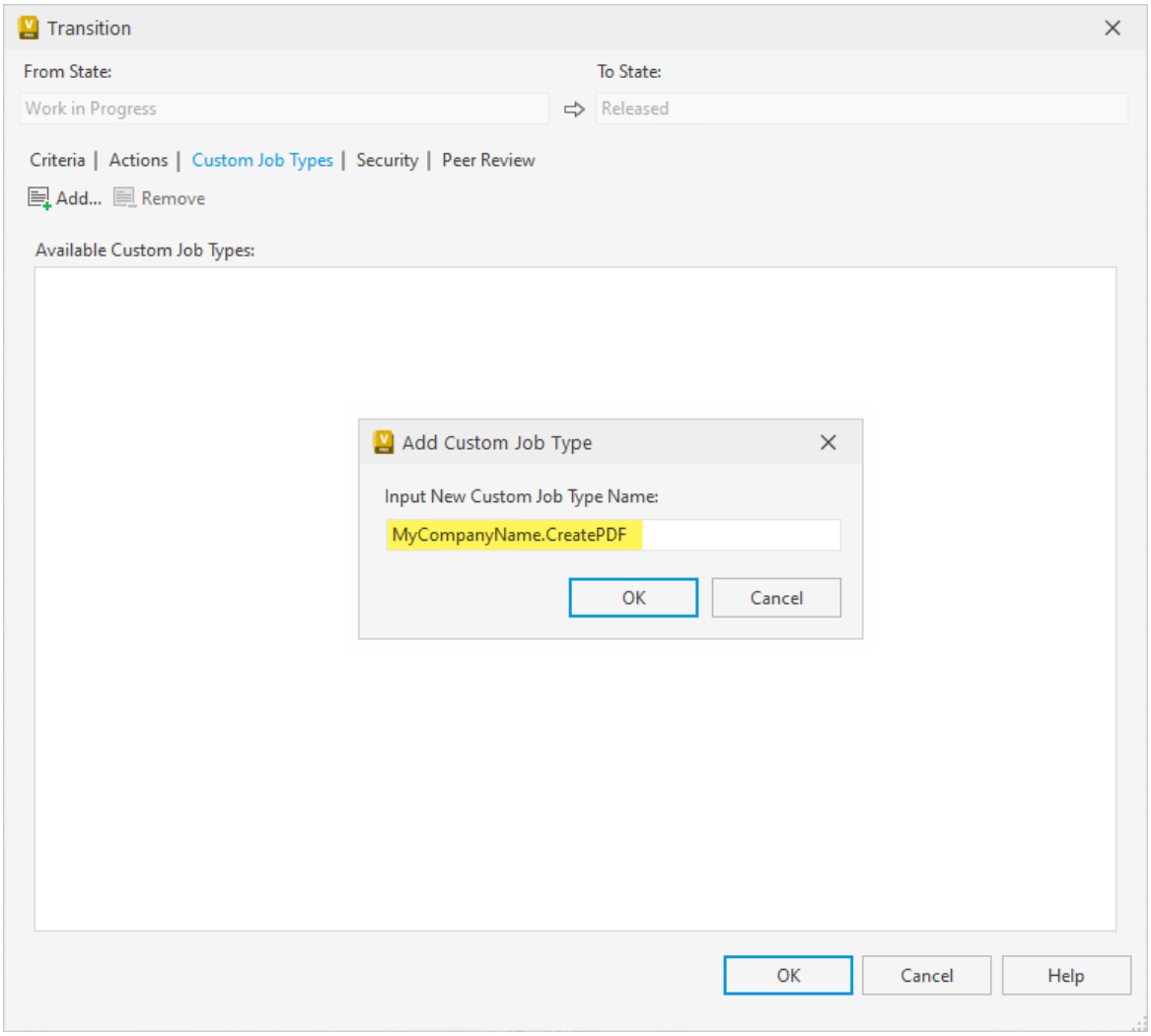
For **development purposes** the "*Submit Job*" button of the *powerJobs Settings Dialog* can be used to *queue* a job manually. To do so, select the desired object (file, item, etc…) in Vault client and open the powerJobs Settings Dialog. After clicking the button, the job gets submitted with the highest priority (1).



The powerJobs Settings Dialog allows also to queue jobs automatically based on ***lifecycle state changes*** or manually by using ***context menu items***. These features require powerJobs Client to be installed on the workstations of interest.

Associate a job with a lifecycle transition in Vault

To automate or assist workflows it is possible configure Vault to automatically queue jobs on Vault lifecycle changes. Each single Vault lifecycle state transition must be extended with the according Job Type.



Afterwards when changing the state of a Vault File, Folder, Item or Custom Object the specified job will be queued.

Lifecycle Event Editor (for Change Orders)

For Change Orders jobs can be configured to be automatically queued when they reach the desired state, by using the Lifecycle Event Editor:

1. Get and extract the application from GitHub
2. Run the executable *LifecycleEventEditor.exe*
3. Login to Vault (a user with administrative privileges is recommended)
4. Navigate to the "Change Order" tab and select the default Workflow from the dropdown menu
5. Select the desired transition (e.g. Approved → Close)

6. Click on "Actions" → "Add Job to Transition" and add the job type without extension (e.g. "MyCompany-Name.CreatePDF")
7. Click on "Actions" → "Commit Changes" to save your actions and proceed then to close the application

PowerShell customizations

Alternatively it is recommended to trigger jobs by using the Add-VaultJob cmdlet as the Priority of the created Job can be configured this way.
Jobs with higher priority will be executed first. The lower the number the higher the priority.

The following example can be used on workstations where powerEvents is installed. As in the previous examples, jobs are automatically queued at certain lifecycle transitions.
Just create your own client customization using the folowing code:

```
Register-VaultEvent -EventName UpdateFileStates_Post -Action {
        param($files, $successful)

        foreach($file in $files)
        {
                $job = Add-VaultJob -Name "MyCompanyName.CreatePDF" -Parameters @{
↪"EntityId"=$file.Id;"EntityClassId"="FILE"} -Priority 1
        }
}
```

Jobs can be triggered for Vault *Files*, *Folders*, *Items*, *CustomObjects* and *ChangeOrders* this way, by passing the required parameters.

Time triggered

Time triggered jobs can be configured so that they are executed in a specific Interval by creating a *"{Job Name}.settings"* file in the *Jobs directory*:

```
{
  "Trigger": {
    "TimeBased": "0 0 8 ? * MON,TUE,WED,THU,FRI *",
    "Vault": "Vault",
    "Priority": 10,
    "Description": "This job is triggered weekdays at 8:00 am"
  }
}
```

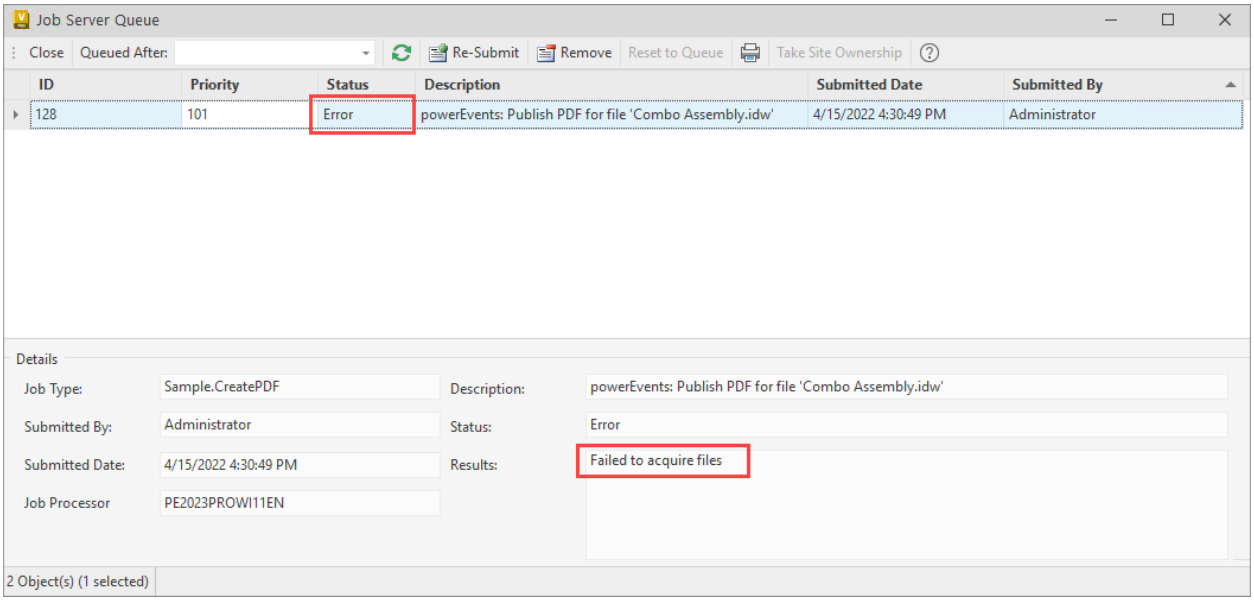
4.4.7 Errors

In job scripts powershell's throw keyword should be used to raise exceptions manually and they can be handled with *try/catch blocks*:

```
try{
        $files = $vaultConnection.FileManager.AcquireFiles($settings)
} catch{
        throw("Failed to acquire files")
}
```

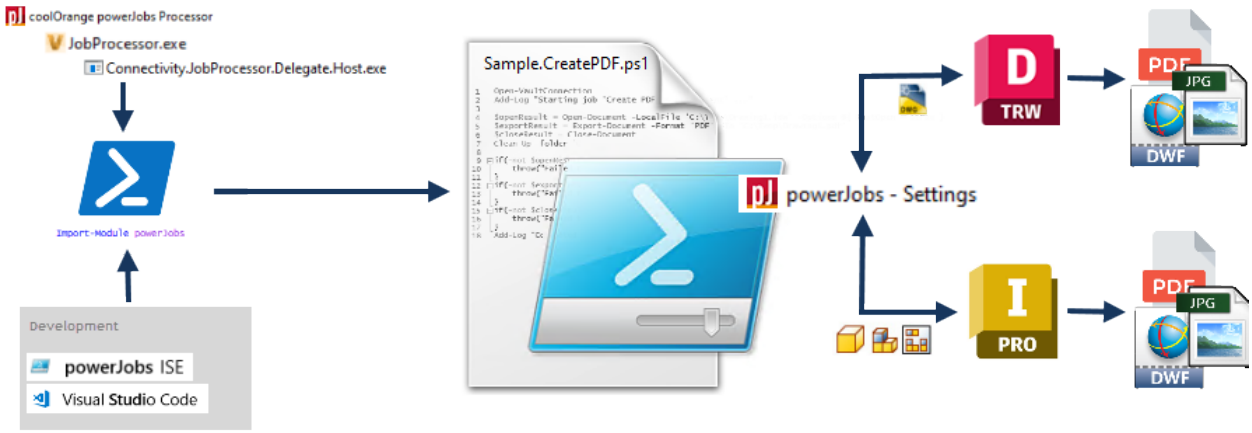
By default the script execution will terminate, because the *$ErrorActionPreference* variable is set to 'Stop'.

In order to change the PowerShell error handling behavior for all jobs globally, the variable can be changed in the *Setup_Job.ps1* script.
When an exception is thrown within a job script, the job is marked with the status **Failure** in the Job Queue.
The exception messages can be found directly in the Result of the failed jobs.



Exception details of the failed job are displayed in *Trace Window*.
Additionally more detailed information about all the Warnings and Errors that where logged during the job execution can be found in there or in the *logfile*.

4.5 Overview

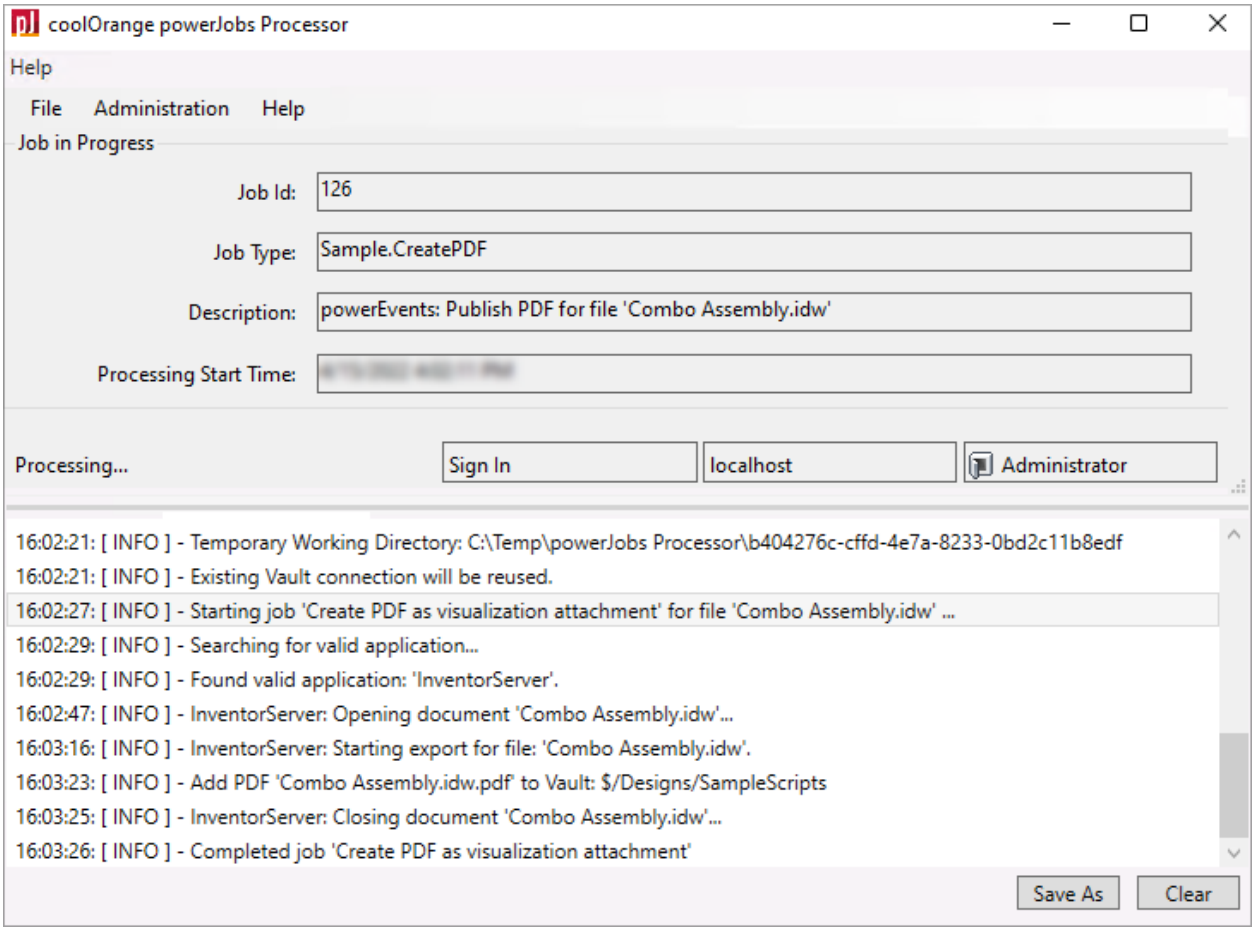


4.6 Extended JobProcessor UI

powerJobs Processor has its own executable, the powerJobs.exe. It automatically loads the Autodesk Vault JobProcessor and provides additional features.
You can still start the JobProcessor through the Autodesk Vault JobProcessor.exe but we recommend to use the powerJobs.exe.
In the powerJobs Processor Toolbar you can find the *About Dialog* and the link to the *Online Help*.

The following features are only supported using the **powerJobs.exe:**

- *Job Processor*
  - *Overview*
  - *Extended JobProcessor UI*
    - *Trace Window*
    - *Job Synchronization*
    - *Restriction Checks*
    - *Time Triggered Jobs*
      - *Setting Files*
      - *Example*



4.6.1 Trace Window

The trace window is a developer tool for the powerJobs Processor.
It gives you the possibility to log debug information without the need to open the log file every time.
What is getting logged is configurable in the *powerJobs Processor.log4net* file.

4.6.2 Job Synchronization

At every start of the application the Jobs placed in the folder "*C:\ProgramData\coolOrange\powerJobs*" are getting synchronized with the *powerJobs Processor.vcet.config* file to support the Jobs within the Job Processor.
For further information see *Registered job types*.

4.6.3 Restriction Checks

At every execution of *powerJobs.exe*, it checks if the current Windows user has enough permissions to execute and synchronize jobs properly.
If the user has not enough rights then the powerJobs Processor can not be started. In the log file will be written in detail what rights are missing on which path.

4.6.4 Time Triggered Jobs

Allow specific jobs to be triggered at certain times repeatedly. A separate settings file is used to specify when and how the job is added to the job queue.

Setting Files

In the folder *%ProgramData%\coolOrange\powerJobs\Jobs* create a text file with exactly the same name as your job, but with the file extension *.settings* instead of *.ps1*.
The settings file is written in json format .

- **TimeBased**: A cron expression that encodes the interval
- **Vault**: The name of the Vault database, the job should be triggered for
- **Priority**: Vault job priority
- **Description**: Vault job description

Syntax:

```
{
        "Trigger":
        {
                "TimeBased": <string>,
                "Vault": <string>,
                "Priority": <integer>,
                "Description": <string>
        }
}
```

Restart powerJobs Processor to make the changes effective.

Note: powerJobs Processor can trigger a job only if the same job isn't already pending in the job queue.

Example

Triggering the job *coolOrange.powerJobs.CreatePdfAsAttachment* every weekday at 8 AM for the vault "Vault" with job priority 10:
In this example the settings file has to be named *coolOrange.powerJobs.CreatePdfAsAttachment.settings*

```
{
        "Trigger":
        {
                // The cron expression for the interval. http://www.cronmaker.com/ can
↪be used to generate the desired expression.
                // every weekday at 8 AM: 0 0 8 ? * MON,TUE,WED,THU,FRI *
                "TimeBased": "0 0 8 ? * MON,TUE,WED,THU,FRI *",
                // This job will be queued for the vault with the name "Vault"
                "Vault":"Vault",
                // And this two parameters are optional and self explaining:
                "Priority":10,
                "Description":"This job is triggered weekdays at 8:00 am"
        }
}
```

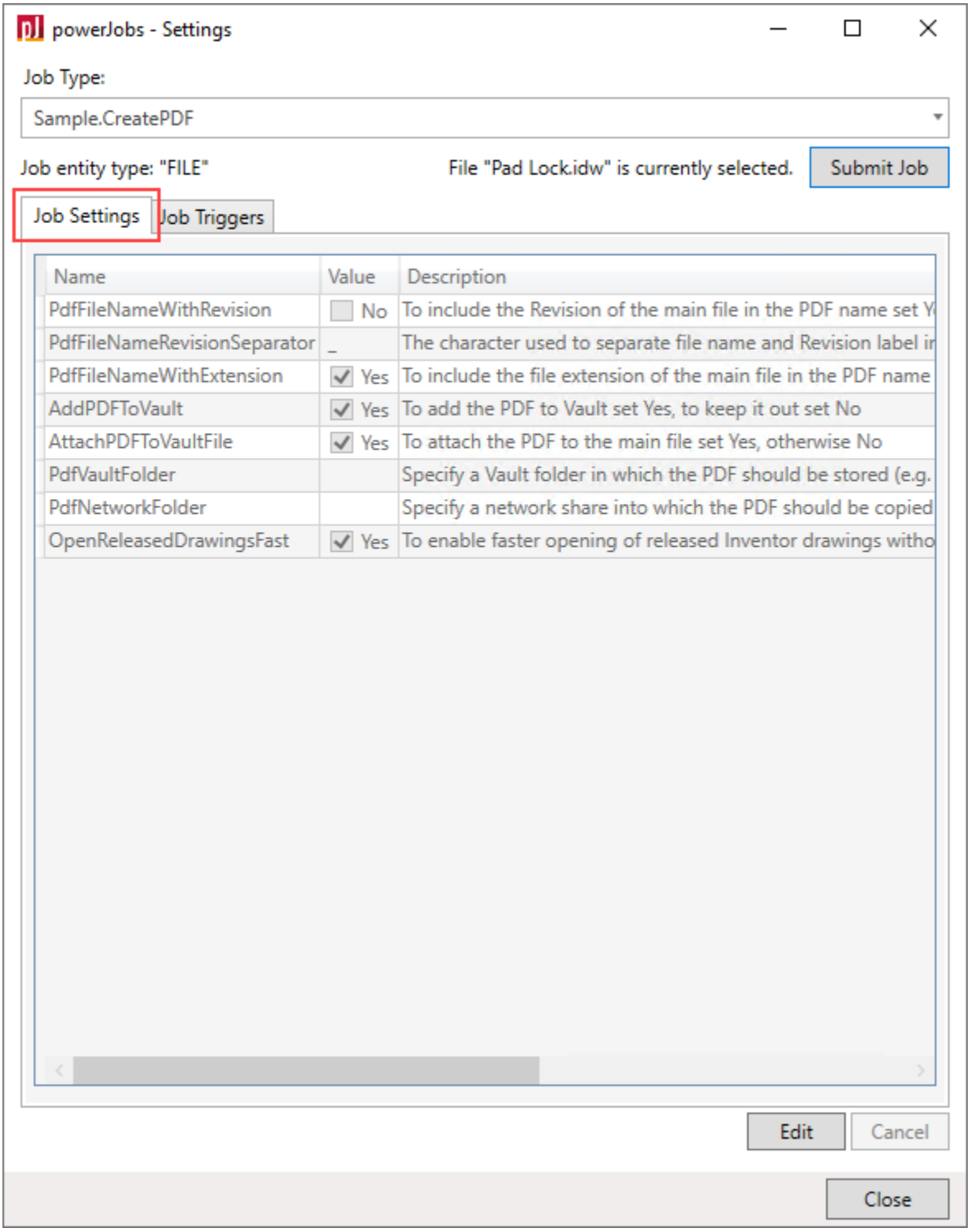
CHAPTER

FIVE

JOB CONFIGURATION

5.1 Job Settings Tab

The Job Settings Tab of the powerJobs Processor Settings Dialog displays the *Settings* for the selected job and allows to edit them. The settings are retrieved and stored directly in the according *job* file and are therefore Vault independent.



When a job that declares settings is selected, the table displays the following information for each setting:

- Name
- Settings current Value
- Description

5.1.1 Editing Settings

The values of settings can be edited by selecting a job from the dropdown list and clicking the **Edit** button.
After clicking the button, the settings in the **Value** column are editable. A setting Name is not editable.

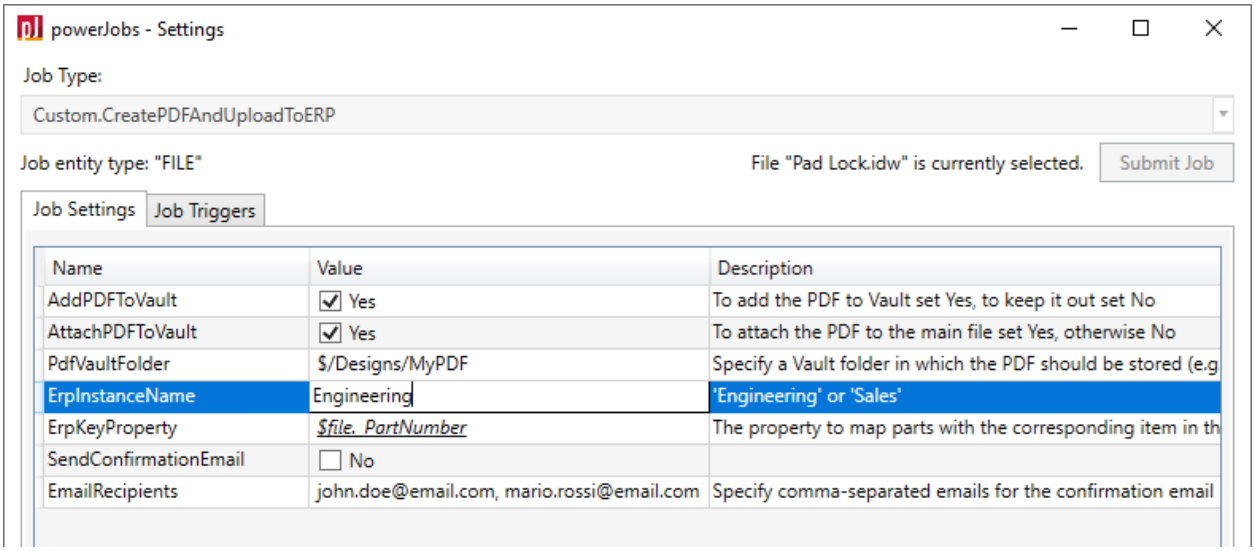
Values are displayed based on the type of the *underlying PowerShell variables*:

- **Switch** settings allow to enable or disable particular job settings by clicking the Checkbox.
- **Text** settings allow to specify setting values as string contents.
- **Powershell** settings allow experienced users to specify settings as Powershell syntax which are evaluated as the job runs. Powershell settings are highlighted as underlined and italic in the dialog.

Warning

Multi-Line **Text settings** are currently not supported by the powerJobs Settings Dialog! This includes also setting specifications in the form of PowerShell line-feeds (`` `r`n ``, `` `n ``).

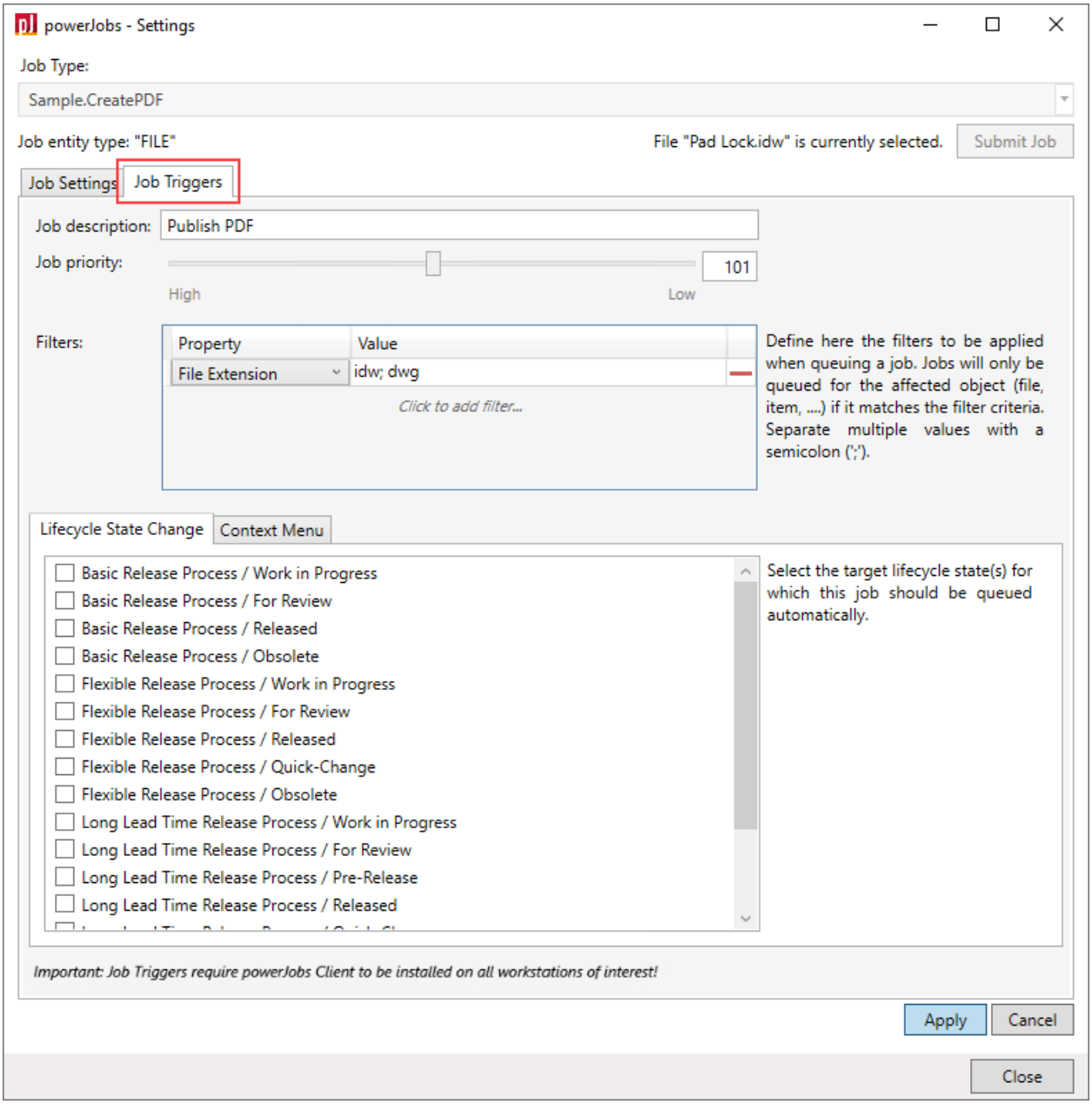
While the settings are edited, the job type can not be changed and jobs can not be submitted using the **Submit Job** button.
The settings must either be saved using the **Apply** button or discarded with the **Cancel** button. Pressing **Apply** writes the values verbatim to the *settings section* in the Job.
The **Cancel** button discards the current changes and revert all values back to the original values.



The screenshot aboves demonstrates a fictive custom job which creates a PDF and uploads the file to the company's ERP system.
The job may have been a copy of the *Sample.CreatePDF* and declares additional variables which allow to steer the behavior of the job towards the upload to the ERP system.

5.2 Job Triggers Tab

The Job Triggers Tab of the powerJobs Processor Settings Dialog displays the Triggers for the selected job and allows to configure them. The configured Triggers only apply to the **current Vault**. Job Triggers do not support replicated environments.



The input masks in the beginning of the Job Triggers tab allow the following configurations:

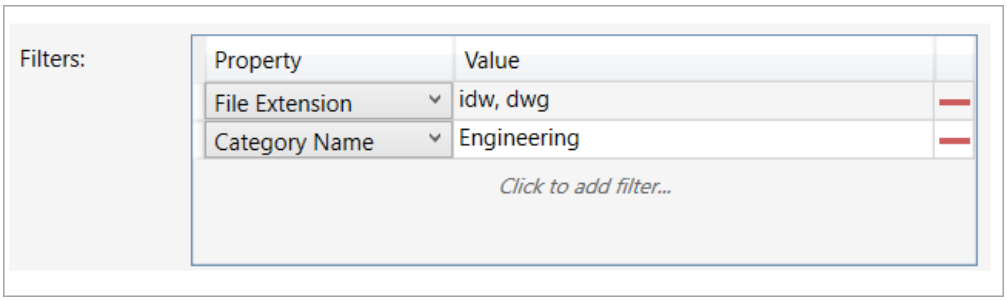
Job Description The description for the queued job. The entity name will be appended automatically. E.g. the description "Publish PDF" results in "*Publish PDF: Bosch Caster.idw*". If no description is set, a default value containing the job's name will be used.

Job Priority The priority wherewith the job is queued. The minimum priority value is 1 (highest priority) and the maximum value is 200 (lowest priority).

Filters Filters are conditions that the entity (File, Folder, Item, Change Order, Custom Entity) must meet, for a job to be queued when a configured trigger takes effect. Filters are applied on Vault properties (system- and user-defined properties). When no filters are set, the job is always queued for the entity. When multiple filters are set, all filters must match for the job to be queued. It is possible to specify one or multiple values for each filter. When multiple values are set, at least one value must match for the filter to match. **Multiple values** must be separated with **semicolons** (`;`).

Refer to the following example to queue the job only for files with the file extension `idw` **or** `dwg`, **and** which are assigned to the Category `Engineering` the following filters can be used:

- One filter for the `File Extension` vault property with the filter value `idw; dwg` to match both extensions.
- A second filter for the `Category Name` vault property with the value `Engineering`

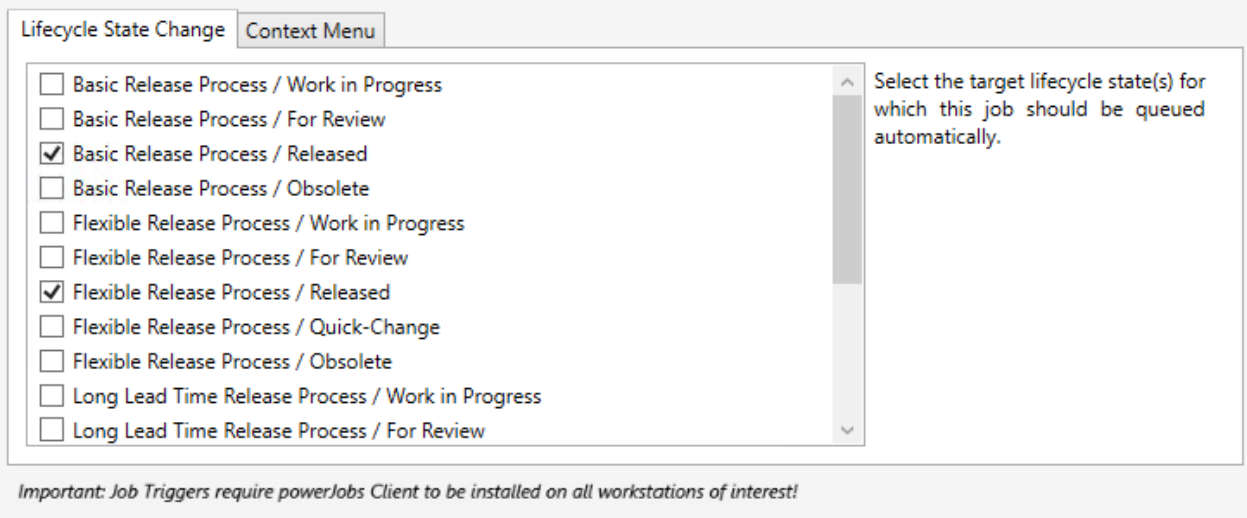


The previously mentioned settings can be considered as general settings because they **apply to all the triggers** which can be configured in the tabs beneath:

- *Lifecycle State Change Trigger* to queue the job based on a lifecylce state change in Vault
- *Context Menu Trigger* to display a context menu item in the Vault client to manually queue the job

5.2.1 Trigger job automatically on lifecycle state changes

The tab **Lifecycle State Change** displays all the **available lifecyle states** for the selected job's *entity type*.



It allows then to configure one- or multiple target states which queue the job, when a jobs entity (File, Item, etc…) reaches the according state and when all configured *filters* apply. To **disable** the Lifecycle State Change trigger **no state** can be **selected**. Lifecycle States are displayed in the format: `Lifecycle Name / State Name` and for Change Orders: `Workflow Name / State Name`.

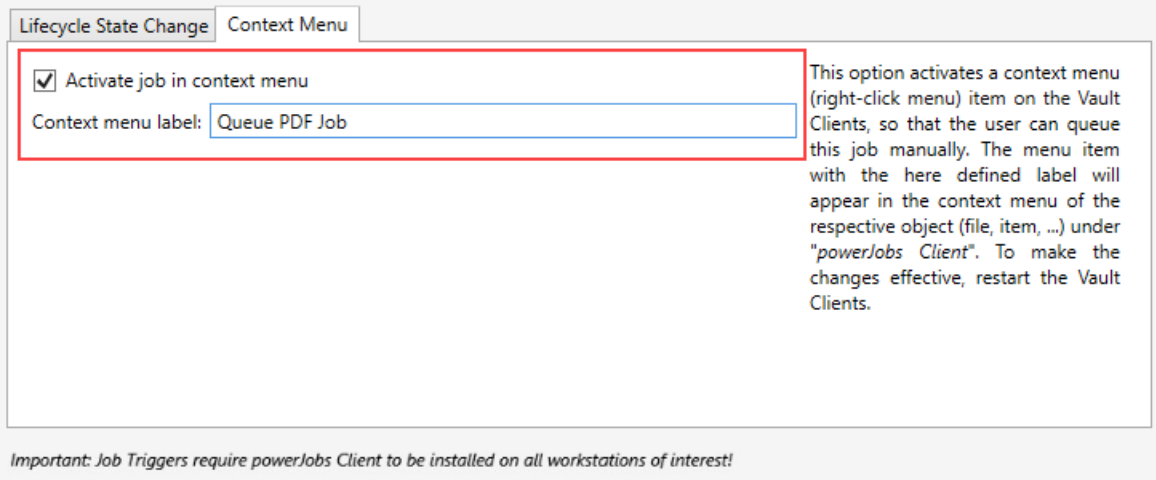
Requires powerJobs Client

This feature requires powerJobs Client to be installed on the **workstations** as it relies on the SubmitJobsOnLifecycleTransition script.

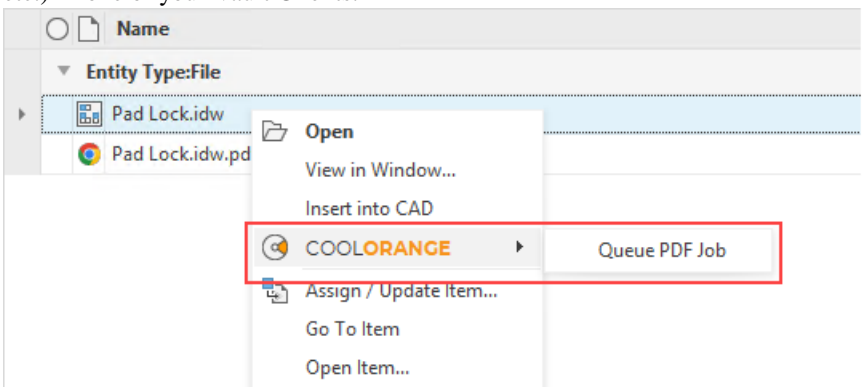
- **Minimum powerJobs Client version**: 23.0.1
- **Folder Lifecycle State Change trigger require**: 25.0.13.

5.2.2 Trigger job manually by context menu item

The tab **Context Menu Trigger** allows to configure if the currently selected job should be **queueable from the right-click menu** in Vault Clients for the according *entity type*. To enable this option, activate the checkbox in the dialog. You can set a custom name (label) for the menu item in the textbox below.



Once enabled, the menu item appears in the *"COOLORANGE"* submenu when right-clicking an entity (File, Item, etc.) in one of your Vault Clients.



Clicking the menu item will queue the job, but only if all configured *filters* match. It is also possible to select multiple elements in Vault and submit jobs for all of them at once.
If no job is queued because the filters do not apply, a messagebox will inform the user.

Changes made in the dialog become active **after restarting** the Vault Clients. This feature is currently **not supporting** custom entities!

Requires powerJobs Client

This feature requires powerJobs Client to be installed on the **workstations** as it relies on the SubmitJobsOnVaultMenuItemClick script.

- **Minimum powerJobs Client version**: 23.0.2
- **Folder Context Menu trigger require**: 25.0.13.
- **CustomObject Context Menu trigger require:** 25.0.15.

Note that prior to powerJobs Client version 25.0.14, all menu items appear in a *"powerJobs Client"* submenu.

Default Values

Some Jobs are delivered with default values for their *job triggers*. These default values are **only loaded** when the affected Job is **not already configured** in the current Vault environment.
Default values are loaded from files in the *jobs directory* with the same name as the Job script but ending in `.defaults` instead of `.ps1`. The default settings files do not contain *Lifecycle State Change Trigger*, so automatic job queueing is **disabled by default**.

Automatic clean-up of settings

On startup of the powerJobs Settings Dialog various "clean-up" operations are executed in the background in order to guarantee integrity and consistency between **already stored trigger settings** and the actual state of jobs and Vault configuration.

These actions are carried out for the following conditions:

1. when existing jobs are deleted or re-named in the *jobs directory* the related stored trigger settings are purged
2. if lifecycle definitions -or states are deleted or re-named in Vault, the according activated *Lifecycle State Change Trigger* will be removed for all jobs.
3. if Vault properties are deleted or re-named in Vault, the according activated *Filters* will be removed for all jobs.

The log file informs about the affected jobs and its removed settings.

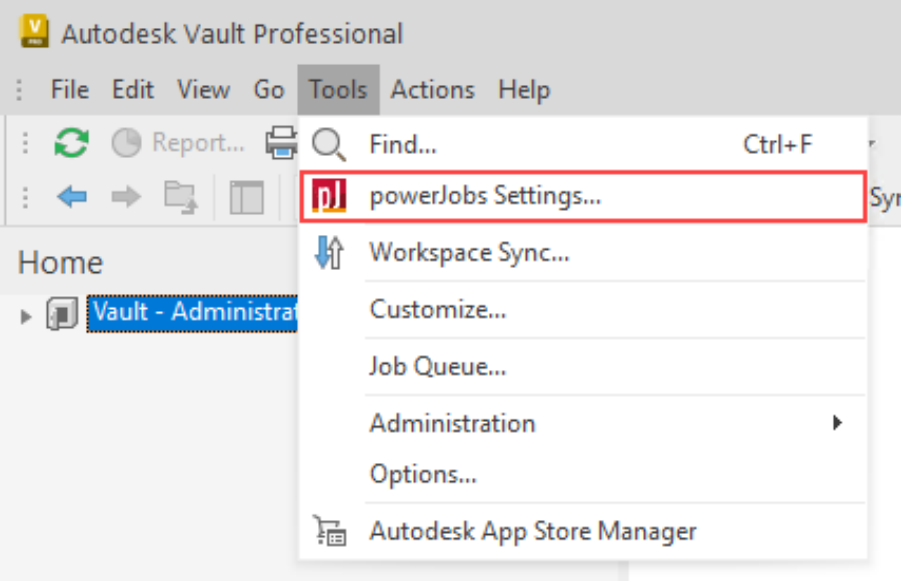
By extending the Vault Client, powerJobs Processor provides a dialog to configure various features of *jobs* directly in the Vault Client.

This includes displaying and editing *powerJobs Job Settings*, configuring smart triggers for the automatic queueing of jobs or controling whether jobs should be available in the Vault context menu.

Note: The powerJobs Settings Dialog is only available on the machine on which the powerJobs Processor is installed!

5.3 powerJobs Settings Dialog

The powerJobs Settings Dialog can be opened via the **Tools** → **powerJobs Settings...** menu entry in the Vault Client.



The powerJobs Settings Dialog supports the following features:

- *Queueing Jobs*
- Viewing and modifying *Job Settings*
- Viewing and modifying *Job Triggers*

The dialog allows configuring settings of all the jobs in the *jobs directory* on the current Job Processor machine. Newly *created jobs* in the jobs directory are displayed as soon as the dialog is closed and re-opened.

When a Vault entity (File, Item, Folder, Change Order or Custom Object) is selected in the Vault Client and the powerJobs Settings Dialog is opened, a job of the selected job type can be *submitted for testing purposes*.

5.3.1 Requirements

Once a user has **logged-in** to the Vault Client the powerJobs Settings Dialog is accessible.

On startup of the dialog a check whether the **Job Queue** is enabled on the current Vault Server is performed and it tries to enable it in case it is disabled. The logged-in user must have the following Vault permissions to perform these actions:

- *Vault Get Options* to check if the Job Queue is enabled
- *Vault Set Options* to enable the Job Queue on the Vault Server

---

Note: By default, the required Vault permissions are already assigned to most of the available roles in Vault.

To edit *Job Settings* or *Job Triggers* the following requirements must be met in addition to the previously mentioned Vault permissions:

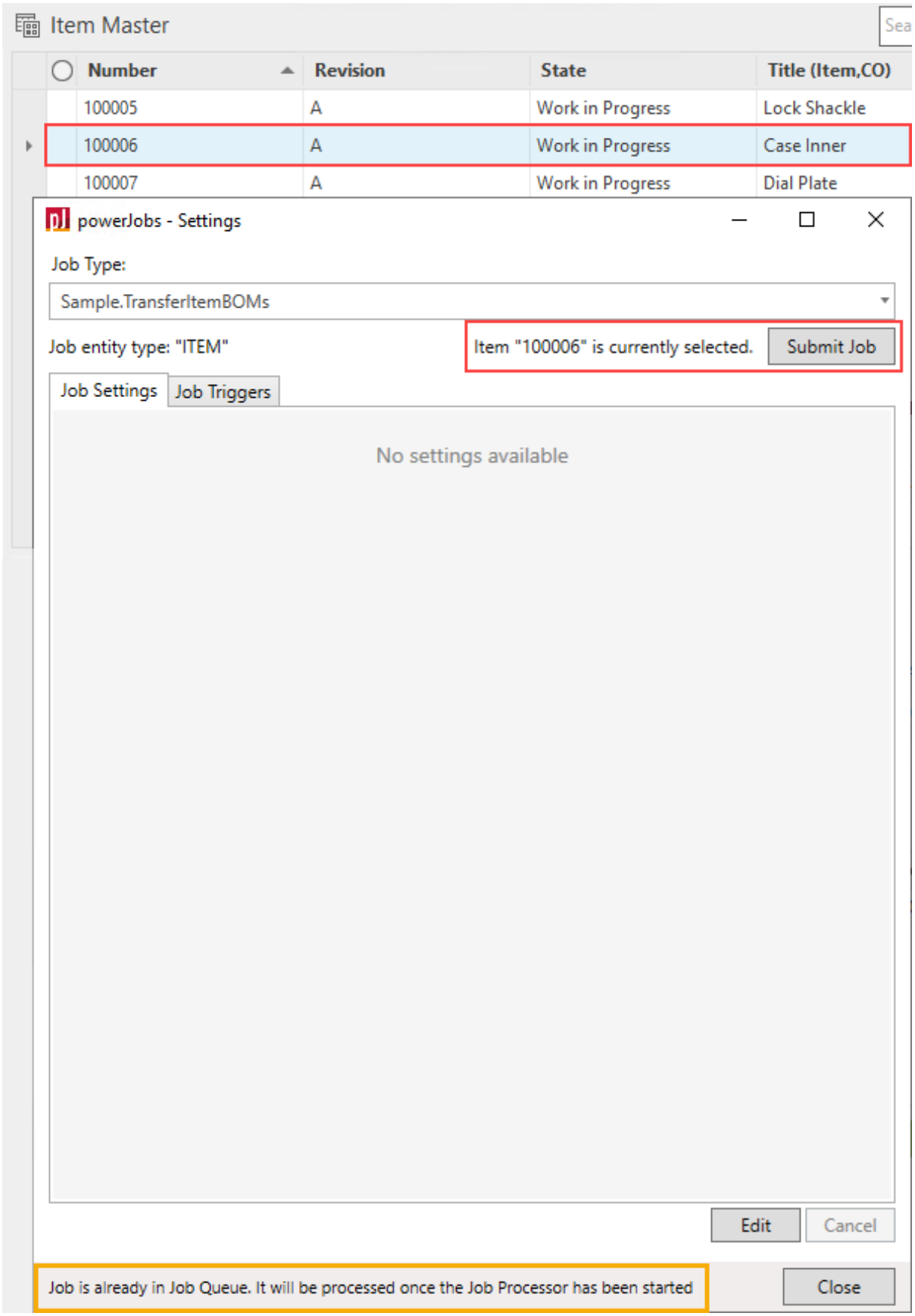
- The selected job must be have a valid *Vault entity assigned*.
  This implies that jobs designed for Items, Change Orders and Custom Objects can only be edited by the Setting Dialog within Vault Professional.
  Jobs with no entity type assigned can not be edited.

In case the requirements are not met, the status bar of the dialog provides according information.

5.3.2 Queueing Jobs

The **Submit Job** button adds a job of the selected type to the Job Queue for the currently selected entity in the Vault Client. The job will be triggered for the **exact version** of the selected Vault entity. The job can be changed by clicking the arrow on the right of the combobox and selecting a different entry.

The status bar then provides information on whether the job was successfully queued.


Item Master
Number
Revision
State
Title (Item,CO)
100005
A
Work in Progress
Lock Shackle
100006
A
Work in Progress
Case Inner
100007
A
Work in Progress
Dial Plate
powerJobs - Settings
Job Type:
Sample.TransferItemBOMs
Job entity type: "ITEM"
Item "100006" is currently selected.
Submit Job
Job Settings
Job Triggers
No settings available
Edit
Cancel
Job is already in Job Queue. It will be processed once the Job Processor has been started
Close

Jobs that are added to the Job Queue using the powerJobs Settings Dialog are queued with entity specific parameters such as the *EnitiyId* and *EntityType* of the selected elements and are queued with priority 1 which is the highest job priority. The powerJobs Settings Dialog supports triggering jobs for *Files*, *Items*, *Folders*, *Change Orders* and *Custom Objects*.

The following parameters are passed to the Job:

- `EntityId`
- `EntityClassId` (with value "FILE" for *Files*, etc.)

These parameters are used in the *Setup Job script* to automatically create the Powershell representation of the *entity* (`$File`, `$Folder`, `$Item`, `$Changeorder`, `$Customobject`).

CHAPTER

SIX

LOGGING

6.1 Logging Levels

ALL	Everything is written to the logfile
DEBUG	Debuginformation is written to the logfile
INFO	Every error, warnings and infos are written to the logfile
WARN	Every error and warnings are written to the logfile
ERROR	Every error is written to the logfile
FATAL	Only critical errors are written to the logfile
OFF	No logging

powerJobs Processor uses Apache log4net as core logging library, and additionally PostSharp Diagnostics for extended Debug logging.
By default, all the logs are stored in a logfile located in *'C:\Users\{USER}\AppData\Local\coolOrange\powerJobs Processor\Logs\powerJobs Processor.log'* and it contains Infos, Warnings and Errors.
Perhaps you can find backups of previous logfiles in this directory.
The log4net settings file is located in C:\Program Files\coolOrange\powerJobs Processor\powerJobs Processor.log4net.
Further information about log4Net Configurations can be found here.
You can change the logging behaviour of:

- the powerJobs Processor with the TraceWindow: *powerJobs.exe*
- the *JobProcessor* extension
- the powerJobs *Settings Dialog*
- the *PowerShell IDE*

6.2 When to change the logging behavior?

When you have issues or when you want to get a more detailed knowledge about what powerJobs Processor or powerVault cmdlets are doing, you can increase the *Logging Levels*.

Note: When changing the loglevel to DEBUG PostSharp Diagnostics will be enabled and will log all the function calls into the log files.
This could cause performance issues.

Additionally you can change the logfile location or integrate the logging mechanism into your administrative environment by using build in EventLogMessages etc.

6.3 powerJobs.exe

If you need to get more details of errors that happen in the powerJobs Processor application, this is the right place to change the loglevel.
Following section is used to control the logging behaviour for the powerJobs.exe:

```
<logger name="ProcessorWindow">
</logger>
```

Following *LogAppender* is used:

6.3.1 LogFile

This is the main *LogAppender* used in all the loggers. If you want to change the *Logging Levels* in the logfile, please visit following appender:

```
<appender name="FileAppender" type="log4net.Appender.RollingFileAppender">
```

In the line

```
<param name="File" value="${LOCALAPPDATA}\coolOrange\powerJobs Processor\Logs\powerJobs␣
↪Processor.log" />
```

you can configure the outputpath and name of the logfile.

Since this appender has no configured *LevelRangeFilter*, it's loggingLevel has to be configured on the loggers.
In the lines

```
<root>
        <level value="INFO" />
        <appender-ref ref="FileAppender"/>
        <appender-ref ref="ColoredConsoleAppender" />
</root>
```

you can configure the logginglevel. You could set the level to "DEBUG", than all the levels between the range Debug and Fatal will be logged.

6.4 JobProcessor Addin

By default the Autodesk JobProcessor creates logfiles located in the *'C:\Users\{USER}\AppData\Roaming\Autodesk\Autodesk Vault Job Processor'* directory when jobs are processed.
Its verbosity can be increased by modifying the configuration file in C:\Program Files\Autodesk\Vault …\Explorer\JobProcessor.exe.config.

In order to increase the *Logging Levels* for your custom Jobs, however the following section can be used to control the logging behaviour for the *JobHandler*:

```
<logger name="powerJobs.JobHandler">
        <appender-ref ref="MsgAppender"/>
</logger>
```

powerJobs Processor provides it's own *PSHost*, for which the logging can be configured in the section:

```
<logger name="coolOrange.Powershell">
        <appender-ref ref="MsgAppender"/>
</logger>
```

Following sections are used to control the logging behaviour for the powerJobs- and powerVault-Cmdlets:

```
<logger name="powerJobs.Cmdlets">
        <appender-ref ref="MsgAppender"/>
</logger>
<logger name="powerVault.Cmdlets">
        <appender-ref ref="MsgAppender"/>
</logger>
```

In addition to the *LogFiles* generated by powerJobs Processor the following *LogAppender* is available:

6.4.1 Trace Window

If you want to change the logging level or format of the logs that are displayed in the *Trace Window* please visit the *MsgSenderAppender* which is only used from the JobProcessor extension:

```
<appender name="MsgAppender" type="powerJobs.Common.MsgSenderAppender, powerJobs.Common,␣
↪Version=26.0.0.0, Culture=neutral, PublicKeyToken=b8042d5e1878fff1" >
```

In the following lines you see the Range Filter which is currently used. By changing the minimum level to DEBUG it would log debugging information to the Trace Window.

```
<filter type="log4net.Filter.LevelRangeFilter">
        <levelMin value="INFO" />
        <levelMax value="FATAL" />
</filter>
```

By changing the conversion pattern you change how the message looks like.

```
<layout type="log4net.Layout.PatternLayout">
        <conversionPattern value="[ %-5level] - %message" />
</layout>
```

6.5 Settings Dialog

Following section is used to control the logging behaviour for the *powerJobs Settings Dialog*:

```
<logger name="powerJobs.VaultExtension">
</logger>
```

Only the *LogFile* appender is used.

6.6 PowerShell IDE

When debugging or creating jobs in PowerShell IDEs, the same logging as the JobProcessor log window is logged into the PowerShell *Console Window*.
For that the *powerJobs- and powerVault-Cmdlets* loggers are getting used.
In order to customize the logging level in the console window, visit following appender:

6.6.1 ColoredConsoleAppender

ColoredConsoleAppenders are working for PowerShell IDE's that support console windows.

```
35 <appender name="ColoredConsoleAppender" type="log4net.Appender.
   ↪ManagedColoredConsoleAppender">
```

In the lines

```
61 <filter type="log4net.Filter.LevelRangeFilter">
62         <levelMin value="INFO" />
63         <levelMax value="FATAL" />
64 </filter>
```

you can configure the required *Logging Levels*. You could set the minimal filter level to "DEBUG", than all the levels between the range Debug and Fatal will be logged.
We are using a ColoredConsoleAppender, therefore you could also change the colors of the messages, depending on their log level:

```
36 <mapping>
37         <level value="DEBUG" />
38         <foreColor value="Black" />
39         <backColor value="White" />
40 </mapping>
41 <mapping>
42         <level value="INFO" />
43         <backColor value="DarkGreen" />
44 </mapping>
45 <mapping>
46         <level value="WARN" />
47         <foreColor value="Black" />
48         <backColor value="Yellow" />
49 </mapping>
50 <mapping>
51         <level value="ERROR" />
52         <backColor value="Red" />
53 </mapping>
54 <mapping>
55         <level value="FATAL" />
56         <backColor value="DarkRed" />
57 </mapping>
```

Note: When powerVault cmdlet's are getting used before importing the powerJobs Module, not all the powerVault logging can be redirected to the logger configured in the according section of the *JobProcessor addin*

CHAPTER

SEVEN

CHANGE LOGS

7.1 powerJobs Processor v26

7.1.1 v26.0.3

`05-06-2025`

Fixed

- Issues with Job Processor 2026:
  - *Trace Window* remained empty and did not display log entries from executing jobs
  - Compatibility-Issue with powerFLC Processor caused by improperly loaded nested modules (via *Import-Module*) in powerJobs-runspaces.
    As a result, invocations of coolOrange cmdlets via their module-qualified names (e.g. *powerVault\Open-VaultConnection*) failed.
  - Failing *Enter-PSHostRunspace* calls (Attach Debugger) to powerJobs-runspaces, which prevented attaching PowerShell debuggers such as Windows PowerShell ISE or Visual Studio Code
- Issues when debugging in Windows Powershell (5.X) IDEs and consoles:
  - *Open-Document* failed to start DWG TrueView 2026 (*COMException 0x80004005*), as well as 2025 and earlier versions when the cmdlet was explicitly called with parameter `-Application 'DWG TrueView'` (*0x80004005*)
  - importing the powerJobs module on Vault 2026 environments could cause a *StackOverflowException* in some situations that lead to application crashes or a "A new guard page for the stack cannot be created" system error

General

- Improved a display problem for v26, where on non-English Vault installations, the Job Processor dialog is not correctly embedded in the *PowerJobsProcessor* window but opens separately — preventing it from exiting automatically on close
- Updated powerVault to version: 26.0.3

7.1.2 v26.0.2

08-05-2025

Fixed

- Vault 2026 login issues in the PowerJobs Window that prevented the automatic queueing of *Time Triggered Jobs*. Also on newly created Job Processor 2026 environments, the started Vault JobProcessor may entered an error state due to missing login credentials, instead of prompting for user input.
- *Open-Document* calls may failed to start Inventor 2026 on some environments due to a missing *Autodesk.Inventor.Interop* assembly when debugging with Windows PowerShell

General

- Improved the assembly loading mechanism in the Job Processor 2026 to prevent compatibility issues with other add-ins and allow future-safe updates of individual dependencies,
  e.g. when jobs include latest versions of libraries, while Vault 2026 still ships with older ones.
  (In exceptional cases, the *powerJobs Processor.vcet.config* can simply be adjusted so that all assemblies of the PowerShell 7 session are loaded in the *Default AssemblyLoadContext* as usual)

7.1.3 v26.0.1

29-04-2025

Features

- Added support for *PowerShell 7*, enabling cmdlets and custom applications to seamlessly run in both Windows PowerShell and PowerShell Core

General

- Added support for Vault, Inventor and DWG TrueView 2026
- Updated powerVault to version: 26.0.1
- Removed DEPRECATED support for *Autodesk.DataManagement.Client.Framework.Vault.Currency.Entities.ChangeOrder* properties on the automatic *$changeOrder* variable, which was replaced in v22.0 with the PowerShell-optimized powerVault object.

Possible Breaking Changes with Vault Job Processor 2026

Generally, **no breaking changes** are expected when upgrading from previous versions.

However, in rare cases, custom jobs may not work with Vault Job Processor 2026!
In such cases, you'll need to modify these job scripts and modules to ensure compatibility with PowerShell 7.

For more details, Microsoft provides information on missing cmdlets and other potential differences between Windows PowerShell and PowerShell 7.4.

In addition, jobs using *custom-built applications* may not work correctly in .NET 8 environments due to potential assembly loading issues.
To avoid such problems, it's recommended to rebuild your projects targeting *.NET Standard 2.0* or *.NET 8*, ensuring that all package references are correctly resolved for the new runtime.

7.2 powerJobs Processor v25

7.2.1 v25.0.17

26-03-2025

Features

- **Settings Dialog**: Added support for configuring *Job Settings* and *Job Triggers* for jobs using *Custom Object Definition Name* (e.g. Task, Contact) as *Job Entity type*.
  (Configured *Context Menu Job Trigger* for Custom Objects require powerJobs Client version 25.0.15 or newer to be installed on the workstations.)

General

- **Settings Dialog**: Removed support for `CUSTENT`as *Job Entity type*. Instead the *Custom Object Definition Name* should be used.
- Updated powerVault to version: 25.1.14
- The *Setup_job.ps1* has been updated to retrieve folder using Get-VaultFolder

Fixed

- Compatibility-Issue where the powerGate - Settings dialog or powerGate - BOM window would not open in Vault Client when both powerJobs Processor and Vault DataStandard were installed.
- **Settings Dialog**: Issue where the *$customObject* or *$changeOrder* variables were not provided during job execution when the job was submitted using the *Submit Job button* in the Settings Dialog.
  The same issue additionally occurred for Custom Objects when the job was triggered using the *Job Triggers*.

7.2.2 v25.0.16

13-01-2025

Features

- **Settings Dialog**: Added support for configuring *Job Settings* and *Job Triggers* for jobs designed for *Vault Folders*.
  (Configured *job triggers* for Folders require powerJobs Client version 25.0.13 or newer to be installed on the workstations.)

General

- Shared third-party Libraries (e.g. log4net) are no longer installed into the GAC but are now merged to avoid conflicts with other products and plugins
- Updated powerVault to version: 25.1.12
- Updated Licensing to version: 18.4.7
- Changed color for WARN log messages in Console to make them easier to read

7.2.3 v25.0.2

13-05-2024

Features

- Updated Licensing to version: 18.4.1.
  The product can now be easily activated on new Job Processors via a single customer key - together with all other coolOrange products in your subscription.

General

- Added support for Vault, Inventor and DWG TrueView 2025
- The Windows Start Menu now contains the coolOrange-wide *Licence Manager* instead of the *powerJobs Processor License Information* shortcut
- Removed DEPRECATED support for the *Get-PowerJobs* cmdlet, the global *$powerJobs.Applications* variable and the *POWERJOBS_DLL* environment variable, all removed in *v21.0*
- Introduced a new product icon that's simpler and easy to recognize within our product portfolio

Fixed

- Issue with queued *License Jobs* where administrators might have seen outdated expiry dates for their license or trial in the Job Queue, if it was not cleared daily
- After the *license expired or the trial ended*, error details were unfortunately only displayed for the first failed job. All other blocked jobs had an empty result.
- Improved *Open-Document* stability for Inventor by enabling the `UserInteractionDisabled` flag, which addresses starting issues for Inventor 2025 and earlier
- Issues with *Export-Document* using DWG TrueView 2025 and earlier:
  - enhanced publishing stability through retries to reduce the *COMExceptions 0x80010105 (RPC_E_SERVERFAULT)* and *0x8001010A (RPC_E_SERVERCALL_RETRYLATER)*
  - instead of incorrectly returning *$true*, the cmdlet now returns a *FileNotFoundException* after internal DWG TrueView crashes (e.g. due to incorrect *-Options* like a non-existent configuration file)

7.3 powerJobs Processor v24

7.3.1 v24.0.3

29-08-2023

General

- Updated powerVault to version: 24.0.5

7.3.2 v24.0.2

`18-07-2023`

General

- Updated Licensing to version: 18.3.1

7.3.3 v24.0.1

`21-04-2023`

General

- Added support for Vault 2024
- Updated powerVault to version: 24.0.1
- Extended *Sample.CreateDWG* with *$processAutoCAD* setting, which allows to easily steer if DWG's should also be exported for AutoCAD source files
- Updated Licensing to version: 18.2.29
- End User License Agreement (EULA) has changed

7.4 powerJobs Processor v23

7.4.1 v23.0.8

`15-03-2023`

Features

- Create *DXF out of AutoCAD DWG* files (using *Inventor and InventorServer*)
  - The sample jobs *Sample.CreateDXFfromDrawing* and *Sample.CreateDWG* now use InventorServer by default to process all files and can therefore be executed for AutoCAD drawings

General

- Updated powerVault to version: 23.0.14
- *Open-Document* supports opening AutoCAD DWG files also with *Inventor and InventorServer* if they are specified in the *-Application* parameter
- Extended *Sample.CreateDXFfromDrawing* and *Sample.CreateDWG* with *$useInventorServer* setting, which allows to easily switch back to Inventor in case of problems with the job execution through InventorServer

Breaking Changes

Removed function *Test-ApplicationSupportsDocument*
Previously the function `Test-ApplicationSupportsDocument` provided by the *coolOrange.Applications.psm1* module, allowed checking if a specific *application* supports opening a document.
Since AutoCAD DWG files where only supported by the *DWG TrueView* application, the function allowed e.g. to check whether a DWG file is an Inventor DWG or not:

Previous	Now
`if (Test-ApplicationSupportsDocument -Application 'Inventor' -Document 'C:\Vault\Inventor\Pad Lock.dwg') {  Write-Host "This is an Inventor DWG file" }`	`$file = Get-VaultFile -File 'C:\Vault\Inventor\Pad Lock.dwg'if ($file._Provider -eq "Inventor DWG") {  Write-Host "This is an Inventor DWG file" }`

7.4.2 v23.0.6

14-02-2023

General

- Updated powerVault to version: 23.0.11
- The setup has been extended to provide a dependency key that is required for other product setups that depend on powerJobs Processor

Fixed

- Issue where powerVault was listed twice in Programs and Features after upgrading powerVault
- Issue where uninstalling product also uninstalled powerVault, although it was still needed by other products, e.g. powerEvents.

7.4.3 v23.0.4

30-08-2022

Fixed

- Issue in the *powerJobs Settings Dialog* where entries of the "*Job Type*" combobox disappeared when custom display scaling settings were used

7.4.4 v23.0.3

28-07-2022

Features

- New design for *powerJobs Settings Dialog*: Splitted *Job Settings* and *Job Triggers* into separate tabs for easier configuration and simplified user experience
- New functionality in the powerJobs Settings Dialog to configure ***queueing jobs from the Vault context menu***
  - Extended *default job trigger files* with configuration for context menu trigger

General

- Updated powerVault to version: 23.0.6
- Added DEPRECATED support for removed *powerJobs\Open-VaultConnection* cmdlet

Fixed

- Issue in the *powerJobs Settings Dialog* where saving *lifecycle state change trigger* configuration failed when all trigger settings where empty

- Compatibility-Issue with module of powerFLC Processor v23.0.1 (and earlier) which prevented *Open-VaultConnection* from reusing the existing Vault connection in job executions

Breaking Changes

Removed Powershell support for "Job Description" textbox in *powerJobs Settings Dialog* To set the name of the job's entity (File, Item etc…) in the job description, it is not necessary anymore to specify the according Powershell expression in the *Job description* textbox. The job's entity name will be appended automatically. To achieve the job description "*Publish PDF: Drawing.idw*":

Previous	Now
Publish PDF: $($file._Name)	Publish PDF

Removed Open-VaultConnection
Removed the `powerJobs\Open-VaultConnection` cmdlet. Instead, the underlying powerVault Open-VaultConnection cmdlet is used directly.

This means that *unnecessary* additional functionalities for jobs are no longer executed (the "filterconfig.xml" configuration for AutoCAD block attributes is no longer downloaded, local workgroup is no longer recognized in replicated Vault environments, the "JobProcessor.exe.config" file is no longer loaded in PowerShell IDE).

7.4.5 v23.0.1

`21-04-2022`

Features

- Extended *powerJobs Settings Dialog* with new *Job Triggers* section which allows to configure **automatic queueing of jobs based on Vault lifecycle state changes**.
  - Providing *default Job Triggers* for some jobs. The values are retrieved from newly delivered files (ending with `.defaults`) in the jobs directory.
- Jobs can be *associated to Vault entities* (important for the previously mentioned Job Trigger of the Settings Dialog)
- Added support for *console logs* in **PowerShell ISE**

General

- Added support for Vault 2023
- Updated powerVault to version: 23.0.1.
  Memory usage is reduced when $vaultExplorerUtil gets used in a large number of jobs (see 22.0.9).

Fixed

- Vulnerability in *Logging* configuration files by updating *log4net* to v2.0.14 (CVE-2018-1285)
- Issue with *ColoredConsoleAppender* that caused powershell remote hosts to crash when appender was logging to console

7.5 powerJobs Processor v22

7.5.1 v22.0.22

`01-02-2022`

Fixed

- Issue where the *Job Processor* produces notable overhead for each *registered* custom job on startup when activated via *serial number*
- 404 error page *"This page does not exist yet"* opens after clicking **Help** → **Online Help** button in *powerJobs Processor* window

7.5.2 v22.0.20

`16-12-2021`

General

- Assembly *powerJobs.Common* gets installed in the GAC

Fixed

- Issue with jobs which use *custom Applications* but fail to load file or assembly *'powerJobs.Common, Version=22.0.…'* after installing *updates* on Job Processor environment
- Issue with failing *Open-Document* cmdlet which incorrectly reports '*not supported file type*' when *custom Application* is implemented and `ApplicationBase.SupportedFileTypes` does not return file extensions in lower case

Breaking Changes

Custom Applications

Replace powerJobs.Common reference in Visual Studio Project: When rebuilding the Visual Studio project the build fails because the assembly **powerJobs.Common.dll** is not any more available in the directory *"C:\Program Files\coolOrange\Modules\powerJobs"*. The reference must be removed and the assembly must be *re-referenced from the GAC*. From now on the *custom Application* can also be used after *updates*, without having to rebuild the project with the same assembly version which is installed on the Job Processor environment.

Remove coolOrange.GenerateEngine reference in Visual Studio Project and change namespace: The reference to the assembly **coolorange.GenerateEngine** must be removed in your Visual Studio project. All the required interfaces and abstractions for implementing *custom Applications* were moved to the *powerJobs.Common.dll*. Usages of the **namespace** `coolOranage.GenerateEngine` must therefore be replaced with `powerJobs.Common.Applications`.

7.5.3 v22.0.19

06-12-2021

Features

- The *powerJobs Settings Dialog* displays settings based on their type:
  - **Switch** settings with yes/no (boolean) values are displayed as *checkboxes* and can therefore be easily modified
  - **Text** settings can be displayed and modified without worrying about correct PowerShell syntax (quotation, escaping etc.)
  - experienced users which specify more flexibile **Powershell** settings, those expressions are highlighted in italic and underlined
- The jobs *working directory* (located in *C:\Temp\powerJobs Processor...*) gets automatically cleaned up after every job execution.
  - Usages of the *Clean-Up* cmdlet in custom job scripts are therefore no longer necessary and can be removed.

General

- Several modifications to facilitate the usage of *Sample Jobs*:
  - Replaced *$...FileName* settings with easier configurable switches for controlling the name of exported files:
    - *$...FileNameWithExtension* settings allow to remove the Vault main file's extension
    - *$...FileNameWithRevision* and *$...FileNameRevisionSeparator* settings allow to add the Vault source file's revision label
    - Removed *$local...fileLocation* settings which were available in some job scripts
    - Removed *$hide...inVault* settings whereby exported files continue to be displayed in the Vault Client
    - Replaced *$fastOpen* settings in several jobs with easier configurable *$openReleasedDrawingsFast* switch
    - Replaced *$workingDirectory* settings with global *$workingDirectory* variable which gets automatically provided for each processed job

7.5.4 v22.0.16

24-11-2021

Features

- Extended *powerJobs Settings Dialog* to allow editing of setting values

General

- *Jobs directory* Windows Permissions changed to **Read** & **Write** for **Everyone** to allow changes to job settings

7.5.5 v22.0.13

16-11-2021

Features

- The *Submit Job* button allows to queue jobs not only for selected Vault Files but even for Folders, Items, Change Orders and Custom Objects (removed *-FileId* parameter) : - The jobs are queued for the specific version of the selected Vault entity

7.5.6 v22.0.12

11-11-2021

Features

- New *powerJobs Settings Dialog* to display *Job Scripts* and their *Settings*
- Introduced *Settings* region to expose configuration options of jobs

General

- *Sample Jobs* have been extended with *Settings* regions
- *Job Configuration*:
  - The **Create PDF** button and the **powerJobs** → **Job Dialog** menu entires have been removed from the Vault Client
    - Vault Administrators can instead start the *powerJobs Settings Dialog* via **Tools** → **powerJobs Settings...**
    - The *Submit Job* button only allows installed *powerJobs Jobs* to be queued and only for a single selected Vault File
- The Job Queue gets *automatically enabled* once the *powerJobs Settings Dialog* is opened and no longer during the Vault Log-in

Fixed

- *Export-Document* and *Sample Jobs*:
  - Issue with failing exports of sheetmetal parts to *DXF* and *DWG* on Windows locales with decimal separators other than '.' (dot)
    - Issue with corrupted *PDF* files exported from Inventor drawings, when invalid values were specified for the option `Vector_Resolution`

7.5.7 v22.0.8

25-10-2021

Features

- Vault Administrators are informed to start Inventor on newly set up Job Processor environments

General

- Improved error messages of Cmdlet *Open-Document* when *Applications* can not be started (e.g. because of license issues with Inventor, unexpected instabilities with InventorServer or DWG TrueView)
- Updated Licensing to version: 18.2.27

Fixed

- Issue that *Assert Failure* dialogues prevented the user from starting the *powerJobs Processor* on environments where the product or the corresponding License Information Tool have never been started before
- Issue with hanging jobs which are blocked by the *Open-Document* Cmdlet and an opened Registration of Inventor.exe dialog on environments were Inventor was never stared before

7.5.8 v22.0.7

`30-09-2021`

Features

- New sample Job *Sample.CreateDXF&STEPfromSheetmetal* for creating DXF and STEP files from sheet metal parts with simple configurable settings
- Extended sample Jobs *Sample.CreatePDF*, *Sample.CreateDXFfromDrawing* and *Sample.CreateSTEPfromModel* with simple configurable settings like publishing the file to a dedicated network-share

General

- Renamed Sample Job *Sample.CreateDXF* to *Sample.CreateDXFfromDrawing*
- Renamed Sample Job *Sample.CreateSTEP* to *Sample.CreateSTEPfromModel*
- Updated Licensing to version: 18.2.26

7.5.9 v22.0.3

`19-05-2021`

Features

- *powerJobs Processor* automatically logs into Vault with the saved Job Processor login credentials

Fixed

- Issue with user names which contain white spaces could not be used with *powerJobs Processor* for logging into Vault
- Issue that *Assert Failure* dialogues prevented the user from starting the *powerJobs Processor* on environments were no Job Processor login credentials has been saved previously (or after installing Vault Client updates)
- Issue with starting the *powerJobs Processor* when authenticating using *Autodesk ID* (Vault 2022)
- Issue where cryptic error messages were displayed when problems has been detected during the start of *powerJobs Processor*

7.5.10 v22.0.1

`29-04-2021`

Features

- Added support for Vault, Inventor and DWG TrueView 2022

General

- Updated Licensing to version: 18.1.24
- End User License Agreement (EULA) has changed

- Removed DEPRECATED support for removed *Add-Log* cmdlet in *19.0.7*
- Removed DEPRECATED support for removed *ErrorMessage* property on results of document related *cmdlets* in *19.0.7*
- Added DEPRECATED support for replaced *$changeOrder*

Fixed

- Issue that error messages displayed <*No File*> instead of <*ScriptBlock*> in the stacktrace when an error occurred in a script block.

Breaking Changes

Replaced $changeOrder
The *$changeOrder* object of type *Autodesk.DataManagement.Client.Framework.Vault.Currency.Entities.ChangeOrder* provided in Jobs has been replaced by the powerVault ChangeOrder object.
Members of $changeOrder used in Scripts such as `EntityIterationId` and `Number` should therefore be replaced with `Id` and `_Number`.

7.6 powerJobs Processor v21

7.6.1 v21.0.12

`21-12-2020`

General

- Updated Licensing to version: 18.1.21

Fixed

- Issue that led to an unusable machine and failing Jobs after running JobProcessor for a long time

7.6.2 v21.0.10

`24-11-2020`

Features

- *Trace Window* and *logfile* contain more details about *failed jobs* by providing the filename and line number of PowerShell errors
- Job Queue Result displays only relevant *Error messages* instead of all the logged Warnings and Errors

General

- Updated Licensing to version: 18.1.19
- Copyright notices have changed

Fixed

- Issue that the *Trace Window* stopped logging after many jobs were executed
- Issue that the "*Save As*" and the "*Clear*" button in the *Trace Window* disappeared after several log entries

7.6.3 v21.0.8

`08-10-2020`

General

- Updated Licensing to version: 18.1.18
- Added DEPRECATED support for replaced *$powerJobs.Applications*
- Added DEPRECATED support for renamed *POWERJOBS_DLL* environment variable

Fixed

- Issue that the Setup did not display the full version number

Breaking Changes

Re-registering jobs types
It is necessary to start *powerJobs.exe* at least once after upgrading from an older version to synchronize the job types for the Vault Job Processor to accept them.

Replaced $powerJobs.Applications
Property **$powerJobs.Applications** has been replaced by *$Host.Applications* and **$powerJobs** has been removed.

Renamed and changed environment variable POWERJOBS_DLL
Environment variable **POWERJOBS_DLL** has been renamed to *POWERJOBS_PROCESSORDIR* and now points to the powerJobs job handler directory

7.6.4 v21.0.7

`23-07-2020`

General

- Removed DEPRECATED support for removed *$powerJobs.Job* in *18.0.11*
- Added DEPRECATED support for removed *Get-PowerJobs*
- Moved the *Job Processor Extension* and the *Vault Client Extension* to the same *directory*

Fixed

- Issue that the Job Dialog would not open in the *Vault Client Extension*

Breaking Changes

Removed Get-PowerJobs
Removed `Get-PowerJobs` Cmdlet. Use the functions from the *coolOrange.Applications.psm1* module instead.

7.6.5 v21.0.5

08-06-2020

Fixed

- Compatibility-Issue with other coolOrange products using an older *Logging* version

7.6.6 v21.0.3

06-05-2020

General

- Renamed *powerJobs* to *powerJobs Processor*

Breaking Changes

License Registration The License has to be registered again using the same SerialNumber or ActivationFile.

Registry Keys Changed registry keys from '*HKEY_LOCAL_MACHINE\SOFTWARE\coolOrange s.r.l.\powerJobs*' to '*HKEY_LOCAL_MACHINE\SOFTWARE\coolOrange s.r.l.\powerJobs Processor*'

7.6.7 v21.0.2

25-04-2020

Features

- Added support for Vault, Inventor and DWG TrueView 2021

General

- End User License Agreement (EULA) has changed
- Updated Licensing to version: 18.1.17
- Added *powerJobs Information* shortcut to startmenu
- Removed *powerJobs Help* shortcut from startmenu as it can be accessed via powerJobs Information shortcut
- Removed Splashscreen

Fixed

- Issue that wrong file permissions of *powerjobs.vcet.config* were deployed with setup and then *powerjobs.exe* refusing to start
- Issue that changing *$ErrorActionPreference* or passing *-ErrorAction* to the *Open-VaultConnection* cmdlet had no effect

7.7 powerJobs v20

7.7.1 v20.0.8

05-11-2019

General

- Updated Licensing to version: 18.0.10

Fixed

- Issue that no *Result* is shown in Job Queue for failed jobs.

7.7.2 v20.0.6

20-08-2019

General

- Updated Licensing to version: 18.0.8

Fixed

- Issue that the *PowerJobsProcessor* does not show Windows notifications to inform about the expired license

7.7.3 v20.0.5

23-07-2019

Fixed

- Issue that JobProcessor is hanging when started without *extend UI* or without *activated License*

7.7.4 v20.0.4

29-05-2019

- Official Release

Features

- Added support for Token Licensing
- Added support for *Stand-Alone Licensing*

General

- End User License Agreement (EULA) has changed
- Updated Licensing to version: 18.0.8
- Removed “License Information” from the powerJobs Menu

7.7.5 v20.0.1-beta

28-01-2019

Features

- Added support for Vault, Inventor and DWG TrueView 2020 (BETA)

General

- Updated Licensing to version:17.0.3

Known Issues

- When using Inventor 2020 Beta1 the *Open-Document* Cmdlet causes the Inventor to be visible

7.8 powerJobs v19

7.8.1 v19.0.7

24-04-2018

- Official Release

Features

- powerJobs *cmdlets* can be used in every IDE
- support for default PowerShell Host cmdlets like Write-Host, Out-Host etc. in job scripts

General

- Updated to PowerShell 4.0
- replaced Log4PostSharp with PostSharp Diagnostics for extended Debug logging
- Re-enabled extended Debug *Logging* for Vault 2019
- Removed *powerJobs Console* shortcut from startmenu
- Assembly *coolOrange.VaultServices_[Vault Version]* gets installed into the GAC

Fixed

- Increasing memory usage when objects created in job scripts do not get cleaned up manually

Breaking Changes

Removed Add-Log
Removed 'Add-Log' Cmdlet.
Use Write-Host Cmdlet instead.

Changed return type of Open-Document, Export-Document and Close-Document
Removed `$result.ErrorMessage` property and replaced it with `$result.Error` property.
Use `$result.Error.Message` instead.

7.8.2 v19.0.3beta

24-01-2018

Features

- Added InventorServer to supported *Applications*
- Added support for Vault, Inventor and DWG TrueView 2019 (BETA)

General

- Added DEPRECATED support for removed *$powerJobs.Job* in *18.0.11*.

Known Issues

- *Open-Document* doesn't work for DWG files when using DWG TrueView 2019 (BETA), this is caused by a bug in Trueview.

> **Warning:** Extended Debug *Logging* is disabled for Vault 2019

7.9 powerJobs v18

7.9.1 v18.0.12

26-10-2017

General

- Assemblies *coolorange.licensing* and *coolorange.Utils.UI* now gets installed in the GAC

Fixed

- Job failes with error message "Could not load file or assembly licensing.Core or one of its dependencies" when using powerGate in a powerJobs job

7.9.2 v18.0.11

25-08-2017

General

- Updated powerVault to version: 18.0.17
- Removed updating comments in sample jobs when adding attachments to files

Breaking Changes

Removed $powerJobs.Job
Removed 'Job' Member of **$powerJobs** variable. Use *$job* object instead.

Fixed

- MessageBox *'Job Server is enabled'* pops up on every start of Vault Explorer when user has not enought permissions to enable the Job-Queue
- Only files with extension PS1 are recognized as Jobs by powerJobs: e.g. when renaming *.ps1* files into *.ps1_old*, the file is not registered as Job any more, can not be executed in JobProcessor
- *Open-Document*: Improved error-message when Inventor projectFile can not be modified, because other documents with different projectFile are already opened in this session

7.9.3 v18.0.6

`21-04-2017`

- Official Release

General

- Added support for Vault 2018
- Removed support for Vault 2014 & 2015R2
- Updated powerVault to version: 18.0.8
- Changed *Logging* to have a single centralized configuration file and *shared logging assembly* in GAC
- Changed registry keys to "HKLMSoftwarecoolOrange s.r.l.powerJobs": Location and Version

Fixed

- issue with jobs where *Clean-Up* cmdlet is not able to remove exported document from Temp-directory. See Add-VaultFile fix in 18.0.8
- issue with invalid error-message *"Failed to checkin the file"* when vault file is released. See Update-VaultFile fix in 18.0.8

7.10 powerJobs v17

See also:

When upgrading from previous versions read this!

> **Warning:** This version is not compatible with powerVault 18.0 or higher.

7.10.1 v17.0.31

21-10-2016

Features

- Installed "powerJobs 17.0 Logs" shortcut in start-menu section of powerJobs

Fixed

- Moved LogFile to %LOCALAPPDATA%/coolOrange/powerJobs/Logs in order that Non-Admin users have write-access to the files.
- Renamed property in RestartOptions from Inverval to *Interval*
- Option *DocumentCloseCounts* in RestartOptions had no affect when opening documents and immediately closing them again
- Open-Document automatically restarts Inventor when previous application session crashes. This means when a sample-job failes because of an Inventor-crash, the next job will not be affected
- When an application should restart because of the *RestartOptions.Interval* and at the exact same time when a document becomes opened no conflicts occure any more and the restart will be reserved for the next time

7.10.2 v17.0.27

04-08-2016

Fixed

- Issue with shared assemblies with powerVault (e.g Get-VaultFileBom was returning wrong quantities).

7.10.3 v17.0.21

09-06-2016

- Official Release

Fixed

- Support for DWG TrueView 2017

7.10.4 v17.0.11beta

23-03-2016

Features

- Added support for Vault 2017 (BETA)
- Added registry keys under "HKCUSoftwarecoolOrange s.r.l.powerJobs": Location and Version
- Applied new coolOrange standards
- Added Windows 10 support

Fixed

- Install / Upgrade with same version

7.11 powerJobs v16

7.11.1 v16.1.28

07-04-2016

Fixed

- Conflict's with multiple processes - license

7.11.2 v16.1.27

17-2-2016

Features

- Updated Licensing to support change license feature.

7.11.3 v16.1.26

03-12-2015

Fixed

- Export-Document for AutoCad DWG with special character in Layout not working

7.11.4 v16.1.25

09-10-2015

Features

- Cmdlets are logging within the jobProcessor addin
- New cmdlet: *Open-Document*
- New cmdlet: *Close-Document*
- New cmdlet: *Export-Document* (supports FastOpen on PDF generation of inventor drawing files)
- New cmdlet: *Clean-Up* - removes downloaded files
- New cmdlet: **Get-PowerJobs** (gets $powerJobs object)

General

- Redesigned GenerateEngine
- Backwards Compatibility (Vault 2016, 2015R2, 2015, 2014)
- Supporting inventor 2012->2016 for all Vault versions depending on the Vault PublishOptions
- New *Sample Jobs* from powerJobs (foreach supported export format a new sample job)
- Runs *Setup_job.ps1* for each job as preparing
- special Open-VaultConnection in powerJobs module
- *RestartManager* that handles application start/stop for timer interval and CloseDocumentCounts
- Add JobProcessorAppender to powerJobs logger

- *Console appender* for PowerShell IDEs

Warning:

- **Save-FileAs** - Removed cmdlet
- **PrepareEnvironment** - removed completely
- **Publisher** - Not supported anymore and removed!
- **$powerJobs** - object has changed
- **SysProps** - Not available any more
- Use powerVault localization feature instead

7.11.5 v16.0.62

02-07-2015

General

- available with latest poweVault (16.0.38)

Fixed

- profile.ps1 is now handled correctly on uninstall / upgrade

powerJobs Processor is an extension for the Vault Job Processor.
It comes with pre-configured, ready-to-use jobs for PDF creation and more.