

Version
3.0

Document Conversion Service

User Guide

PEERNET Inc.

Copyright © 2011 - 2024

Updated: 6/13/2024

Table of Contents

Welcome to Document Conversion Service	1
Legal Notices	2
System Requirements	4
The DCS Dashboard	6
DCS Settings	8
Watch Folder Settings	18
Desktop Conversion	22
Samples	28
Help Resources	31
Installing and Updating Document Conversion Service	33
How to Backup and Restore Configuration Files and Profiles	35
During the Installation Process	36
Using the Backup and Restore Tool	38
Manually Backup and Restore the Files	55
Using the Configuration Merge Tool	58
Installing Document Conversion Service Silently	63
Activating Document Conversion Service	67
Launching the License Wizard	68
Entering Your Serial Number	70
Manually Activating Document Conversion Service	72
Activation Status Results	77
Viewing Your Activation Status	79
Changing Your Activation Status	81
Renewing Your Annual Subscription	86
View Activation Details	92
Working With Document Conversion Service	93
The DCSAdmin Account	94
What Files Can I Convert?	98

The DCS Dashboard	104
The System Tray Icon	106
The Logging Console	107
Starting and Stopping the DCS Service	114
Editing Files with the DCS Editor	122
Configuring Third-Party Applications Used by Document Conversion Service	127
Adobe Reader for Foreign Languages	128
Configuring Flash for Adobe Reader	129
Autodesk Design Review	130
Setting the Ghostscript Version	135
Vector PDF with Office 2007	137
Microsoft Outlook	138
Animated Images and Movies with FFmpeg.exe	144
Windows Imaging Component (WIC) Add-Ons and Extensions	145
Internet Explorer	146
Outside-In AX	148
Optical Character Recognition (OCR) with Document Conversion Service	154
Converting Files with Document Conversion Service	157
The Convert File Application	160
The Drop Files Converter Desktop Application	163
Command Line Utilities	170
DCSConvertFile	172
DCSConvertFileList	182
DCSConvertFolder	191
DCSCombineFiles	203
DCSCombineFolder	213
DCSExtractResults	224
DCSCreateFileList	226
DCSLicenseDaysLeft	229
The Watch Folder Service	230
Watch Folder Service Overview	234
Starting and Stopping the Watch Folder Service	236

Configure the Watch Folder Service	237
Long Path Name Support	255
High Performance Clustering and Fail Over Conversion	257
OCR Images and Scanned PDF Files to Searchable PDF	264
Processing Outlook and EML Mail Messages and Attachments	266
Creating Done Files to Signal Completion	271
Control Sort Order on File Pickup	272
Post-Conversion Processing	273
Unique File Naming and Flat Folder Structures	279
Skipping Files with the Passthrough Converter	281
Large Volume Batch Conversion Using Clustering	283
Large Volume Batch Conversion Using Synchronous File Pickup	285
Converting With PEERNET.ConvertUtility	287
Requirements	288
Getting Started	289
C# Tutorial	290
Visual Basic .NET Tutorial	295
Using the Results Object	300
Working With PEERNET.ConvertUtility	302
Passing Custom Conversion Settings	303
Converting a Folder of Files	305
Converting a List of Files	309
Combining a List of Files	313
Combining a Folder of Files	317
Combining Select Pages Of Each File	322
Converting Files with Long Path Names	324
Controlling Parallel Document Conversion	326
Controlling the Failed Results File Location	327
Controlling the SmartInspect Logging Files	330
Waiting for Document Conversion Service to be Ready to Convert	334
Deploying Applications	336
PEERNET.ConvertUtility Namespace	340
PNConverter	342
Methods	342
PNConvertFileInfo	365

Methods	365
Properties	366
PNConversionItem	368
Methods	369
Properties	373
PNCombineItem	378
Methods	379
Properties	382
PNConversionResult	387
Properties	387
PNConversionResultError	392
Properties	392
PNConversionResultMessage	393
Properties	393
PNConversionResultOutputFile	394
Methods	394
Properties	396
PNConversionResultOutputFileRenderedPage	397
Methods	398
Properties	399
PNConversionResultPrintJob	403
Methods	404
Properties	405
PNConversionResultPrintJobPrintedPage	411
Methods	412
Properties	413
PNProfile	417
Methods	417
Enumerations	418
PNSetting	419
Methods	419
Properties	420
Enumerations	421
PNConvertResultStatus	421
PNFileSortMode	422
PNFileSortOrder	422
Setting up Client-Server Conversion	423
Setting up the Server	425

Setting up the Client	430
Setting up a Client-Server Watch Folder	434
Microsoft IIS and Document Conversion Service	435
Creating and Customizing Profiles	447
Conversion Settings	452
File Extension to Converter Mapping	457
General Converter Options	461
Built-in Converter OCR Options	465
Built-in PDF Converter Options	468
Built-in Text Converter Options	469
Built-in Cadd Converter Options	479
Built-in Image Converter Options	481
Word Converter Options	482
Excel Converter Options	493
PowerPoint Converter Options	513
Adobe Reader Options	518
Internet Explorer Options	522
Ghostscript Converter Options	530
Image Converter Options	532
OutsidesIn AX Options	536
Save	539
Devmode settings	544
Advanced File Naming	548
Image Options	554
TIFF File Format	559
PDF File Format	562
PDF Security	565
JPEG File Format	568
Processing	570
Advanced Features	579
Endorsement Options	584

Endorsement Formatting Codes	588
Watermark Stamping	592
Advanced Configuration	594
Configuring Parallel Processing	595
Document Conversion Service Startup and Shutdown	597
Document Conversion Service Printer Pool	600
Controlling the Converters	603
The Application Pool	605
Enabling and Disabling Converters	613
Custom Converter Behaviour	615
Changing Document Conversion Service's Startup Mode	618
Appendix	620
General Application Settings	621
Application Factory Settings	624
Converter Factory Settings	627

Welcome to Document Conversion Service

PEERNET Document Conversion Service is a true Windows service that comes bundled with a basic set of converters for converting the most common types of documents, a suite of command line conversion utilities, PEERNET.ConvertUtility.dll, a .NET library to convert files from your own code and a payload plug-in for more advanced needs that can be called from any programming language with COM support.

Document Conversion Service comes with several pre-built sample applications as open source projects. These samples demonstrate how to convert a multitude of document types to various image (picture) formats such as TIFF, JPEG, Adobe® PDF, PNG and others.

The Convert File sample demonstrates using the PEERNET.ConvertUtility.dll to convert files, and the Watch Folder service sample that watches a folder(s) on your system for files to convert and based on its configuration it will convert the documents to the specified format.

The Document Conversion Service is easily configured through its application configuration file to control all aspects of the conversion process, including how many documents in parallel/concurrently it will process and which applications are available to convert documents to various formats.

Legal Notices

Copyright © 2011 - 2024 by PEERNET Inc. All rights reserved.

PEERNET is a registered trademark of PEERNET Incorporated. Microsoft and Windows are registered trademarks of Microsoft Corporation. All other trademarks and registered trademarks are the properties of their respective holders.

PEERNET Inc.
1365 Lords Manor Lane
Ottawa Ontario
K4M 1K3

Information in this document is accurate up to the time of publication, but does not necessarily reflect enhancements made to PEERNET Inc.'s products, which are released without notice. The software described in this document is furnished under a license agreement. It is against the law to copy the software onto any medium, or to use the software for any purpose, except as specifically allowed in the license agreement. No part of this help system may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose other than the licensed operator's personal use, without the express written permission of PEERNET Inc.

This application and some of its associated tools and utilities use OpenSource components. You can find the source code of their open source projects along with their respective license information in the links below. We acknowledge these developers and are grateful for their contributions to open source.

[MsgReader](#)

Project Code: <https://github.com/Sicos1977/MSGReader>
Copyright © 2013-2023 Magic-Sessions
License: (MIT) <https://github.com/Sicos1977/MSGReader/blob/master/license.txt>

[MsgKit](#)

Project Code: <https://github.com/Sicos1977/MsgKit>
Copyright © 2013-2023 Magic-Sessions
License: (MIT) <https://github.com/Sicos1977/MsgKit#readme>

[AlphaFS](#)

Project Code: <https://github.com/alphaleonis/AlphaFS>
Copyright © 2008-2018 Peter Palotas, Jeffrey Jangli, Alexandr Normuradov
License: (MIT) <https://github.com/alphaleonis/AlphaFS/blob/develop/LICENSE.md>

[ImageMagick](#)

Project Code: <https://github.com/ImageMagick/ImageMagick>
Copyright © 1999 ImageMagick Studio LLC
License: <https://imagemagick.org/script/license.php>

[AvalonEdit](#)

Project Code: <https://github.com/icsharpcode/AvalonEdit>
Copyright © 2000-2014 AlphaSierraPapa for the SharpDevelop Team
License: (MIT) <https://github.com/icsharpcode/AvalonEdit/blob/master/Documentation/License.html>

Xceed Extended WPF Toolkit

Project Code: <https://github.com/xceedsoftware/wpftoolkit>

Copyright © 2007-2017 Xceed Software Inc.

License: (Microsoft Public License) <https://github.com/xceedsoftware/wpftoolkit/blob/master/license.md>

MahApps.Metro

Project Code: <https://github.com/MahApps/MahApps.Metro>

Copyright © MahApps.Metro 2011-2018

License: (MIT) <https://github.com/MahApps/MahApps.Metro/blob/develop/LICENSE>

MahApps.Metro.IconPacks

Project Code: <https://github.com/MahApps/MahApps.Metro.IconPacks>

Copyright © MahApps.Metro 2016

License: (MIT) <https://github.com/MahApps/MahApps.Metro.IconPacks/blob/dev/LICENSE>

System Requirements

Document Conversion Service is a highly scalable product with the ability to process many documents in parallel to take advantage of multi-CPU and multi-core systems available today.

Supported Platforms

Only Microsoft® 64-bit operating systems are supported. A minimum of 4GB of memory (RAM) is recommend for best performance.

- Windows Server 2022
- Windows 11
- Windows Server 2019
- Windows Server 2016
- Windows 10
- Windows Server 2012 R2
- Windows Server 2012

Limited Support for End-of-Life Platforms

These Microsoft® Windows operating systems have reached the end of their support lifecycle. Document Conversion Service will still install and run on these platforms, but new features added to Document Conversion Service may not be supported.

- Windows 8.1
- Windows Server 2008 R2
- Windows 7 SP1

Required by Document Conversion Service:

- An account with administrative privileges to use the Document Conversion Service account. This is set up during the installation process but can later be changed as needed through the service properties.
- For concurrent (parallel) document processing, you are only limited by your license and the capabilities of the computer you are running on. The performance of Document Conversion Service is directly tied to the number of CPUs and cores available as well as the configuration settings used to control the amount of resources that can be used by the service.
- To ensure the fidelity of your converted documents, some of the included converters use the application used to create your document in order to do the conversion. For these converters you will need to have installed the necessary third-party applications.

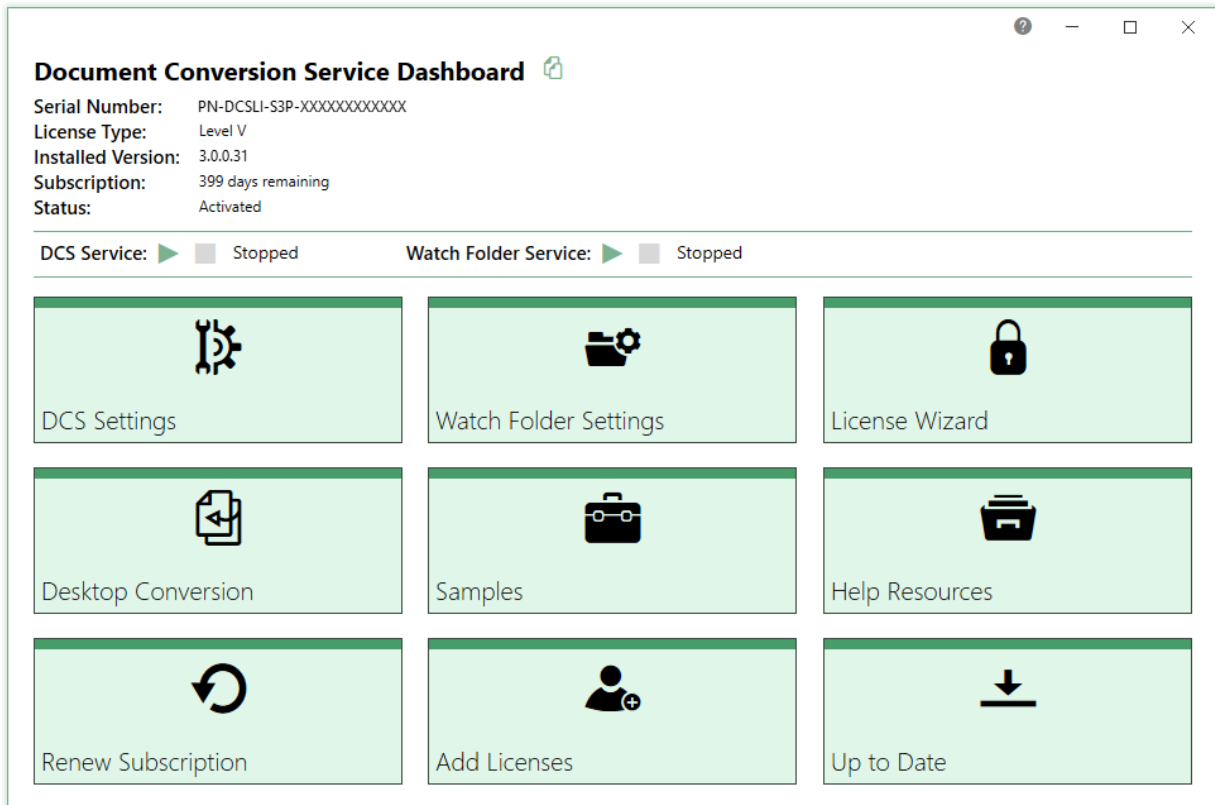
See [What Files Can I Convert?](#) for a complete list of the included converters and any required application and versions supported by each.

- The most common required applications are listed here:

- Microsoft® Office (Excel, Outlook, PowerPoint, Publisher, Visio, or Word) for Office documents
- Optional if you do not use the Builtin PDF converter: Adobe® Reader for PDF files
- Internet Explorer for HTML files
- Autodesk Design Review for Autodesk DWF files.
- Optional if you do not use the Builtin Cadd converter: Autodesk DWG TrueView installed with Autodesk Design Review for DWG files.
- for customers with licensed versions of Outside-In ActiveX Control, you are able to utilize this component as well to perform document conversions
- Optionally install the latest Ghostscript for improved Postscript and PDF file conversion
- The following file types do not need a third-party application and are built-in:
 - PDF files using the included PEERNET PDF converter.
 - Text files using the included PEERNET Text converter.
 - DWF, DWFX, PLT and GBX files using the included PEERNET Cadd converter
 - Postscript files using Ghostscript.
 - Microsoft XPS (XML Paper Specification) files.
 - Image formats including JPEG, TIFF, Windows Bitmap, ZSoft PCX and DCX, CServe Portable Network Graphics and Graphics Interchange Format
- The following printer is occasionally required when creating vector Adobe PDF files.
 - Microsoft® XPS Document Writer

The DCS Dashboard

The **Dashboard** is your hub for all things Document Conversion Service. It combines [product activation](#) and [status](#) with all tools and resources for Document Conversion Service into one place. It provides easy access to starting and stopping both Document Conversion Service and the Watch Folder Service. You can quickly find and edit the Document Conversion Service and Watch Folder Service configurations, add and edit profiles, and access the logging console and saved conversion logs. Other dashboard tiles offer fast access to sample programs, video tutorials, and other help resources.



Starting and Stopping DCS Services

At the top of the dashboard you have easy access to starting and stopping both Document Conversion Service and Watch Folder Service.

DCS Service: Stopped Watch Folder Service: Stopped

License Information

The **Dashboard** displays all status information about your current evaluation or activated license. Once you have activated your purchased copy of the software, you will find the following information in the top left-hand corner of the **Dashboard**:

- **Serial Number** - appears when product is activated.
- **License Type** - the type of license installed, such as Evaluation, Entry Level, etc.
- **Installed Version** - the current version of the product.
- **Subscription** - number of days remaining in your current subscription.
- **Status** - the current activation state.

Tools and Resources

All tools and resources from the Start menu are available through the Dashboard, divided into categories to make them easier to find.

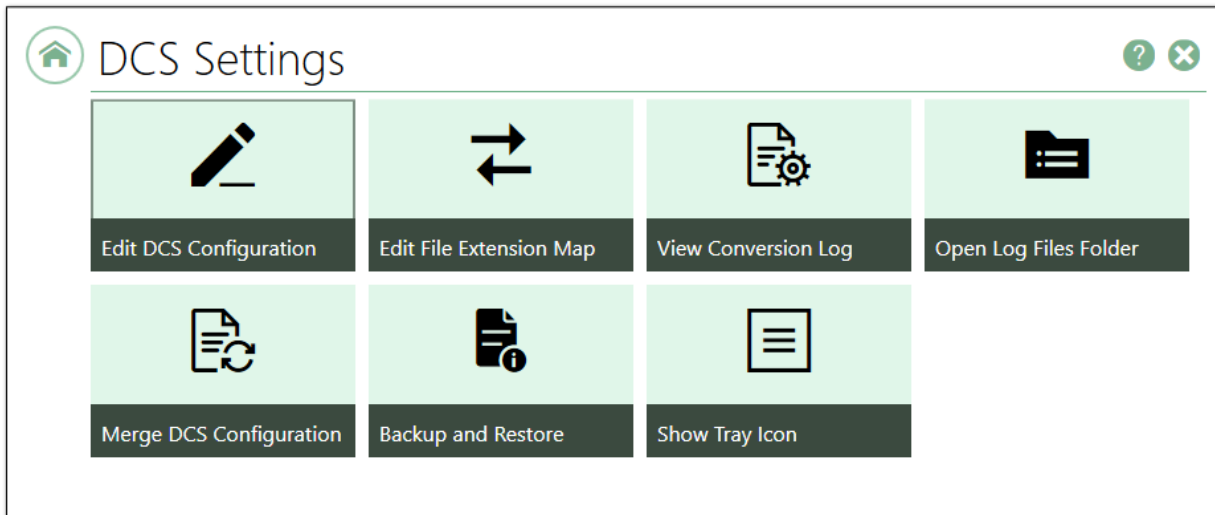
- [DCS Settings](#) - go here to edit the DCS configuration file, manage the file extensions to converting mapping file, edit conversion profiles, view the conversion log or saved logs folder, and find tools for merging, backing up, and restoring configuration files.
- [Watch Folder Settings](#) - from this tile, you can edit the Watch Folder Service configuration, view its conversion log, and open its saved logs folder.
- [Activate Product/License Wizard](#) - depending on the state of your license, the title on this tile will vary. This is where you can activate your product, change your serial number, and manage your licenses.
- [Desktop Conversion](#) - provides access to desktop conversion utilities for converting files and folders of files, and using the command line conversion tools from the command prompt.
- [Samples](#) - go here to open the Visual Studio projects for the Convert File (C#/VB) and Watch Folder Service (C#) utilities, view the PowerShell sample or open the Samples folder directly.
- [Help Resources](#) - this tile contains links to online resources, video tutorials, quick start and user guides for Document Conversion Service, the command line tools, and the NET PEERNET.ConvertUtility API for custom programming and development.

The following tiles are available in the dashboard once the product has been activated.

- **Renew Subscription** - click this tile to log into your PEERNET online account where you can renew your annual subscription.
- **Add Licenses** - select this tile to log into your PEERNET online account where you can purchase additional licenses of Document Conversion Service on the same serial number.
- **Update Available/Check for Updates** - this tile will notify you when there is an update pending for download and install, or take you to your PEERNET online account to see what downloads are available to you.

DCS Settings

The DCS settings section gathers all the tools and links you need for editing and configuring Document Conversion Service into one place.



Clicking a tile above will take you to the help for that tile.

Edit DCS Configuration


This tile is where to go to customize your DCS configuration.

You can customize the startup action for each converter and turn off converters that handle file types you do not need to convert.


Clicking this tile opens a flyout that lists each converter with its current startup option.

A converter has three modes:

- **Auto** - DCS auto-detects if this converter and any third-party application it may require are available and can be loaded. When DCS cannot load a converter set to auto, it skips it and moves to the next one.
- **On** - Document Conversion Service will always try to load the converter. If it cannot, the service will not start. This option is a good choice for file types you require to be able to be converted.
- **Off** - You can turn off converters for file types you do not need to convert.



Edit DCS Configuration



Choose which converters you want available to use when converting files.

AUTO - DCS will auto-detect if the converter is available and will load it.
ON - Set a converter ON when it is required to be available.
OFF - Turn off converters for files types you are not converting.

Choose Converters

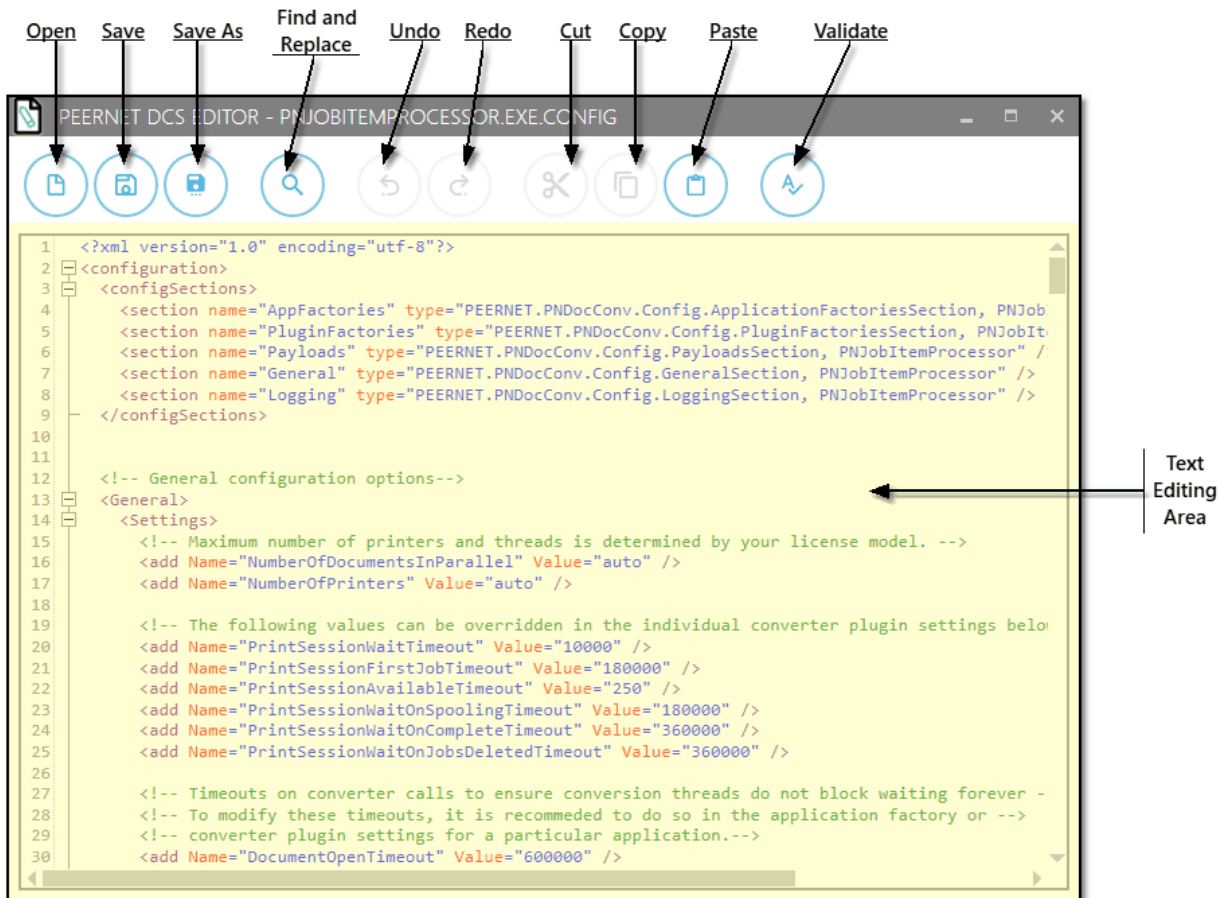
☒ Set all converters to Auto

Auto	PEERNET ArcPDF
Auto	Adobe PDF - Builtin
Auto	Image - Builtin
Auto	Text - Builtin
Auto	Cadd - Builtin
Auto	Microsoft Word
Auto	Microsoft Excel
Auto	Microsoft PowerPoint
Auto	Microsoft Outlook
Auto	Microsoft Publisher
Auto	Microsoft Visio
Auto	Microsoft XPS
Auto	Adobe Acrobat Reader
Auto	Internet Explorer
Auto	Autodesk Design Review

SAVE CHANGES GO TO ADVANCED EDITOR

After making your changes to the configuration, click the **Save Changes** button at the bottom to save them. If the DCS service is running, you will need to [stop and restart the service](#) for the changes to take effect.

If you need to edit other settings in the configuration file, use the **Go to Advanced Editor** button to save any changes and open the configuration file in the [DCS Editor](#).



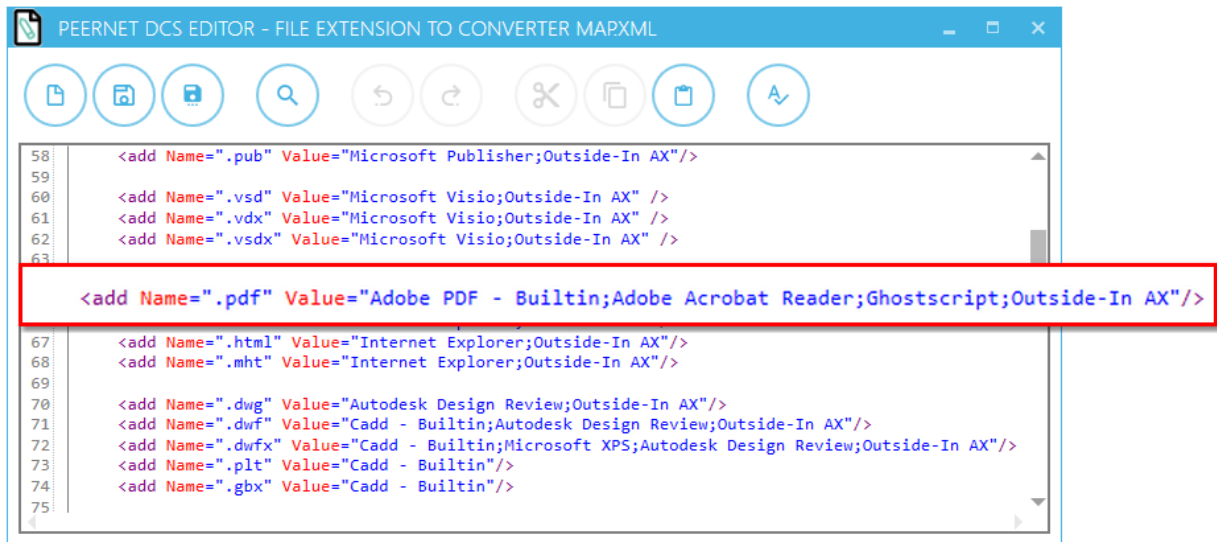
Edit File Extension Map

This option opens the File Extension Map in the [DCS Editor](#).

The file, *File Extension to Converter Map.xml*, maps each file extension to an ordered list of converters that DCS will use to convert those file types. For each extension, you can list one or more converters in order of preference.

For instance, Adobe PDF files commonly end with the file extension **.pdf**. Its mapping is this:

```
<add Name=".pdf" Value="Adobe PDF - Builtin;Adobe Acrobat Reader;Ghostscript;Outside-In AX"/>
```



Here, Adobe Acrobat Reader, Ghostscript, and finally, Outside-In AX, are listed after the built-in converter, Adobe PDF - Builtin, when converting PDF files. DCS tries first to use its built-in PDF converter to convert these files to TIFF or another output format you have chosen. If it is not running, it will try the Adobe Acrobat PDF converter, and so on through the list.

Changing the Converter Order

For each file extension, there is a list of converters to use to convert that file. The converters are in order of preference, separated by a semi-colon (;). You can switch the order of the converters or remove converters from the list if you need to. There must be at least one converter for each extension.

Adding New Extensions

The provided list of extensions to converters covers most cases. You can add new extensions, one per line, as needed.

Further Reading

See [File Extension to Converter Mapping](#).

View Conversion Log

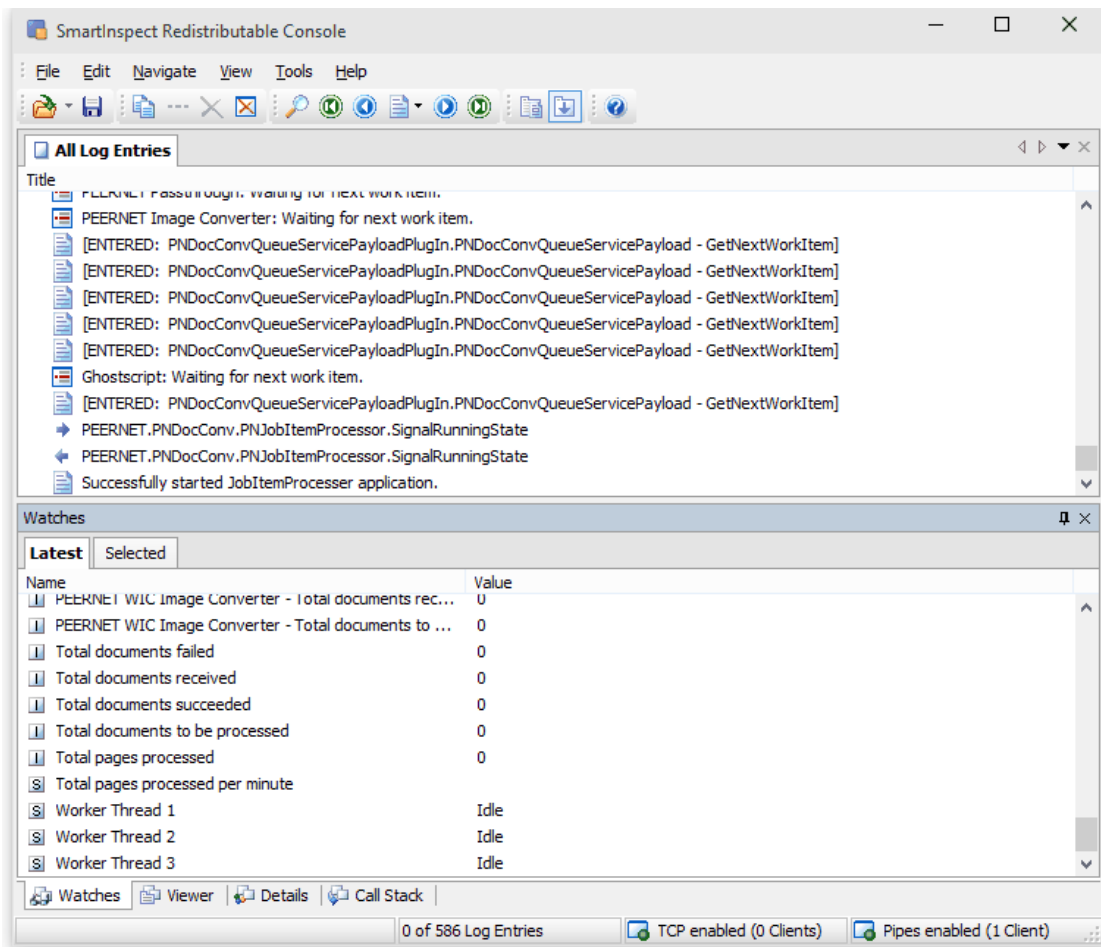
Document Conversion Service generates a conversion log while it is running. The log allows you to troubleshoot start-up issues, monitor the currently running service, and investigate conversion errors should any occur.

Clicking this tile opens the SmartInspect Redistributable Console used to view the log. While Document Conversion Service is running, live logging messages are displayed in the *All Log Entries* panel. A *Watches* tab in the bottom panel displays conversion statistics for the number of applications running and files converted. The *Viewer* and *Details* tabs window in the same section shows details of the logging message selected above.



Do Not Leave the SmartInspect Redistributable Console Open

The SmartInspect Redistributable Console is meant for short term, live logging and troubleshooting access. Do not leave the logging console open for extended periods of time, such as overnight, or it will lock and cause issues with Document Conversion Service.



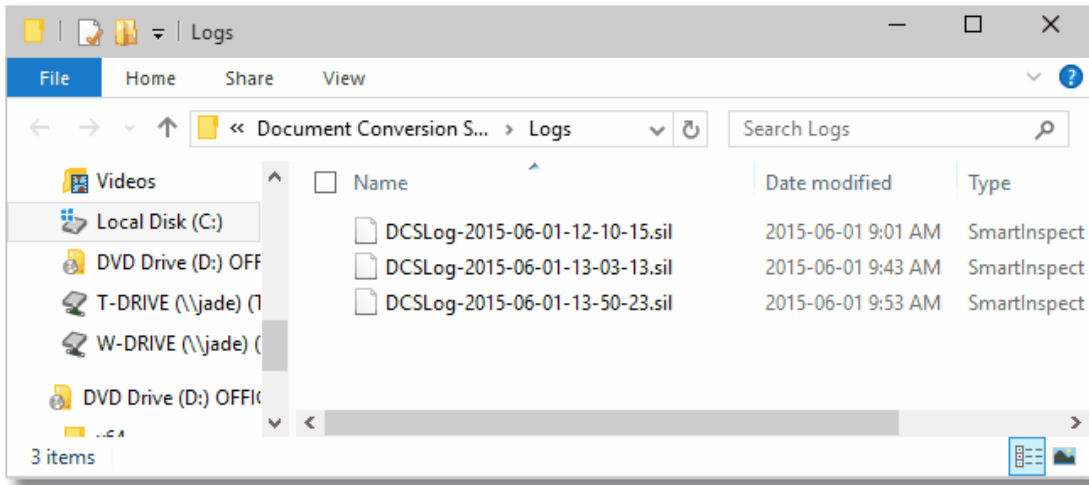
Further Reading

See [The Logging Console](#).

Open Log Files Folder

All logging messages are saved in a round-robin set of files, up to 10 in total, that are rotated based on size (250MB maximum) and by day. This allows you to keep a history of past conversions for troubleshooting.

These log files all start with the name *DCSLog* followed by a date and time. Double-clicking a log file opens that file in a logging console window.

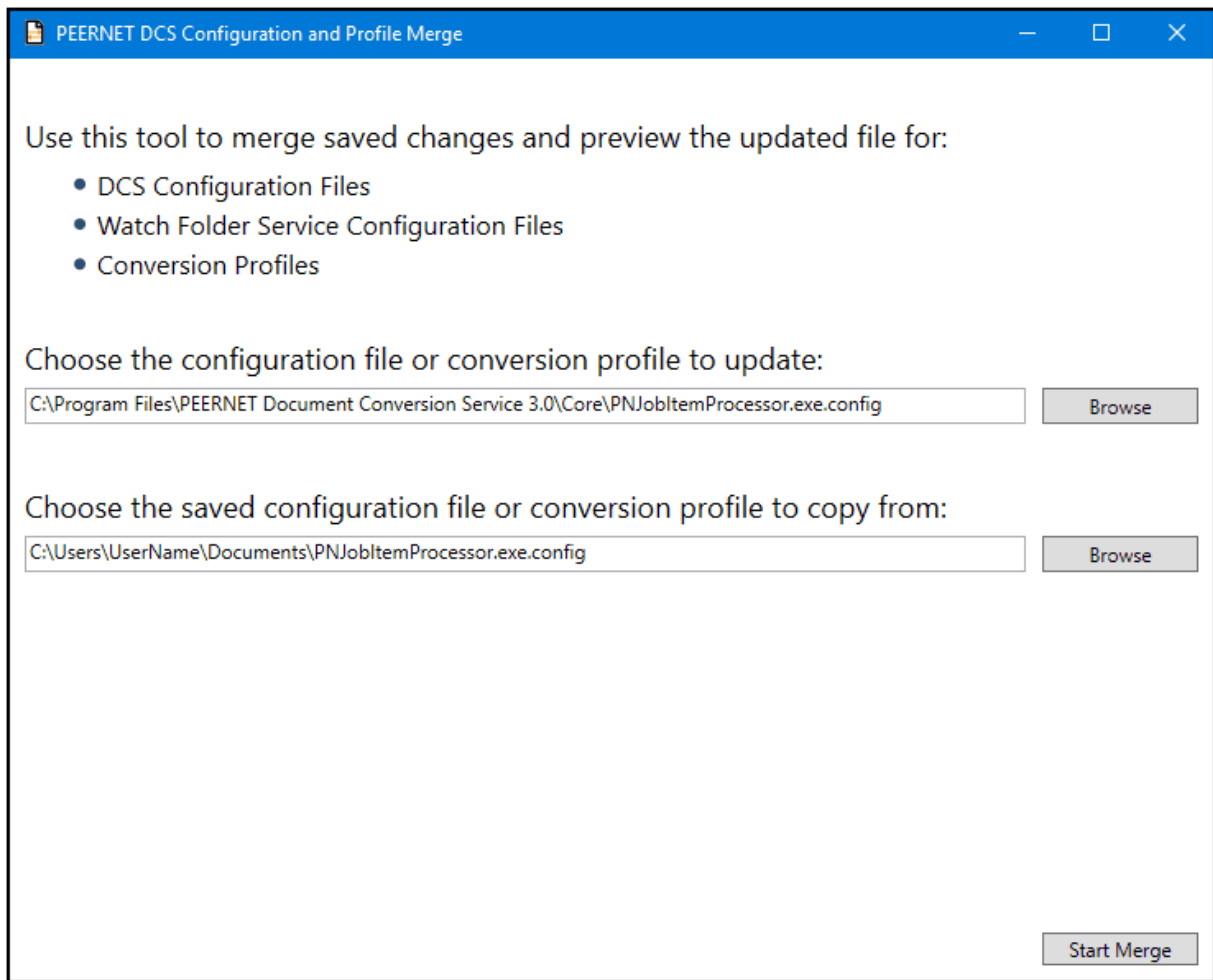


Merge DCS Configuration

This tool makes it easy to manually update the configuration files and conversion profiles used by Document Conversion Service and Watch Folder Service with saved files from a previous installation.

For a single-step backup and restore process, consider [Using the Backup and Restore Tool](#) instead of manually backing up and restore your saved configuration settings.

To merge the saved settings with the new settings, select the new configuration or profile, and then select the matching older, saved file you want to copy any settings from. Select Start Merge to begin merging the files. This tool automates that process with built-in syntax checking and editing before saving the updated files.



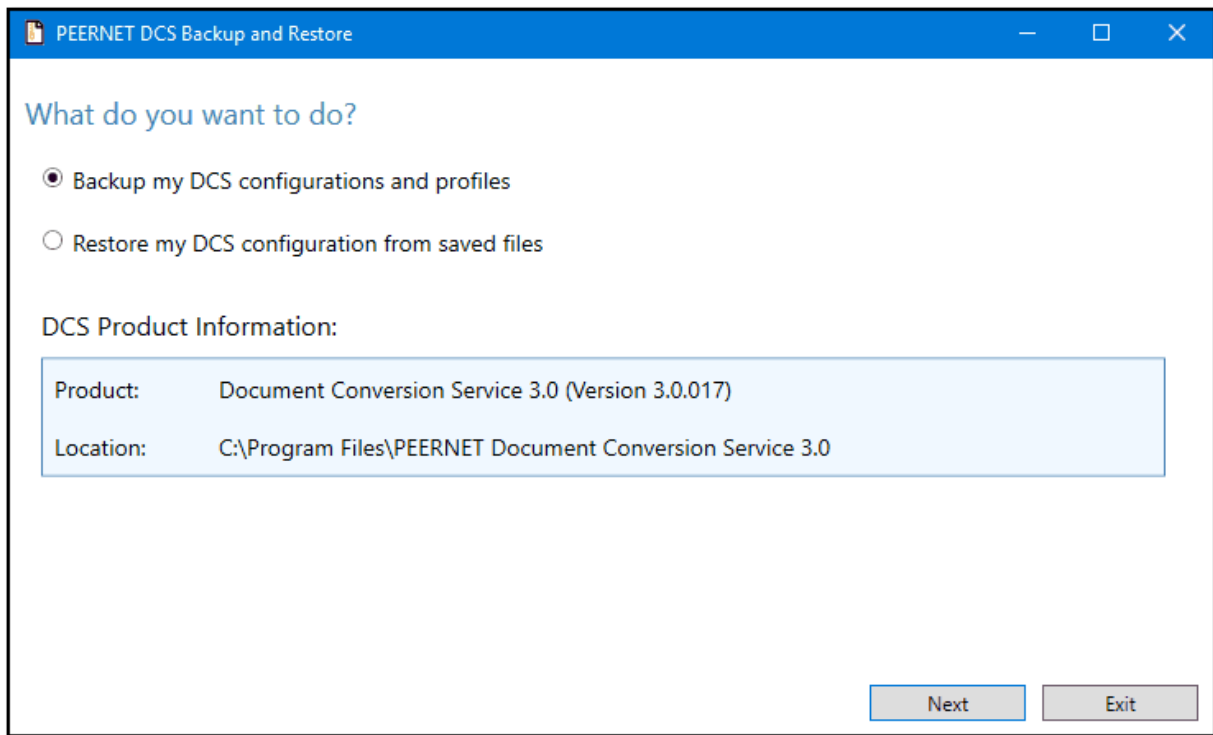
Further Reading

See [Using the Configuration Merge Tool](#).

Backup and Restore

The DCS Backup and Restore utility can backup or restore your Document Conversion Service configuration file, Watch Folder Service configuration file and any created or editing conversion profiles.

When backing up files, it creates a zip file containing the configuration files and conversion profiles you select to back up. The restore process will use the created zip file to merge your configuration files and profiles with the new ones.

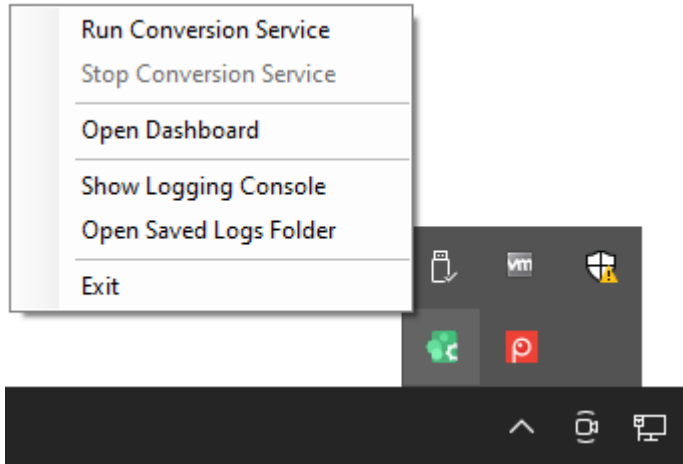


Further Reading

See [Using the Backup and Restore Tool](#).

Show Tray Icon

In addition to the dashboard, there is also a system tray application that provides quick access to starting and stopping the Document Conversion Service service, opening the DCS Dashboard, the [logging console](#) and [the saved log files folder](#).



Further Reading

See [The System Tray Icon](#)

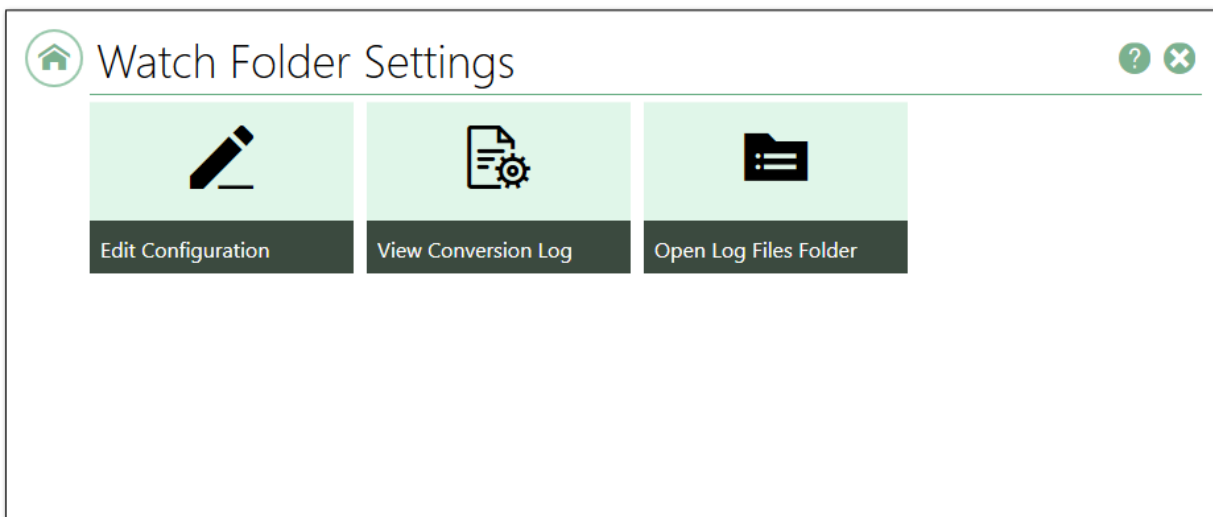
Watch Folder Settings

The Watch Folder Settings section contains links to all the tools you need for editing and configuring Watch Folder Service into one place.

The Watch Folder Service watches one or more drop folders and converts any files or folders dropped into those folders.

Each folder definition controls what file types to pick up, the type of file to create, and where to save the final file. Create custom folder definitions by copying existing ones and editing for your file locations and output.

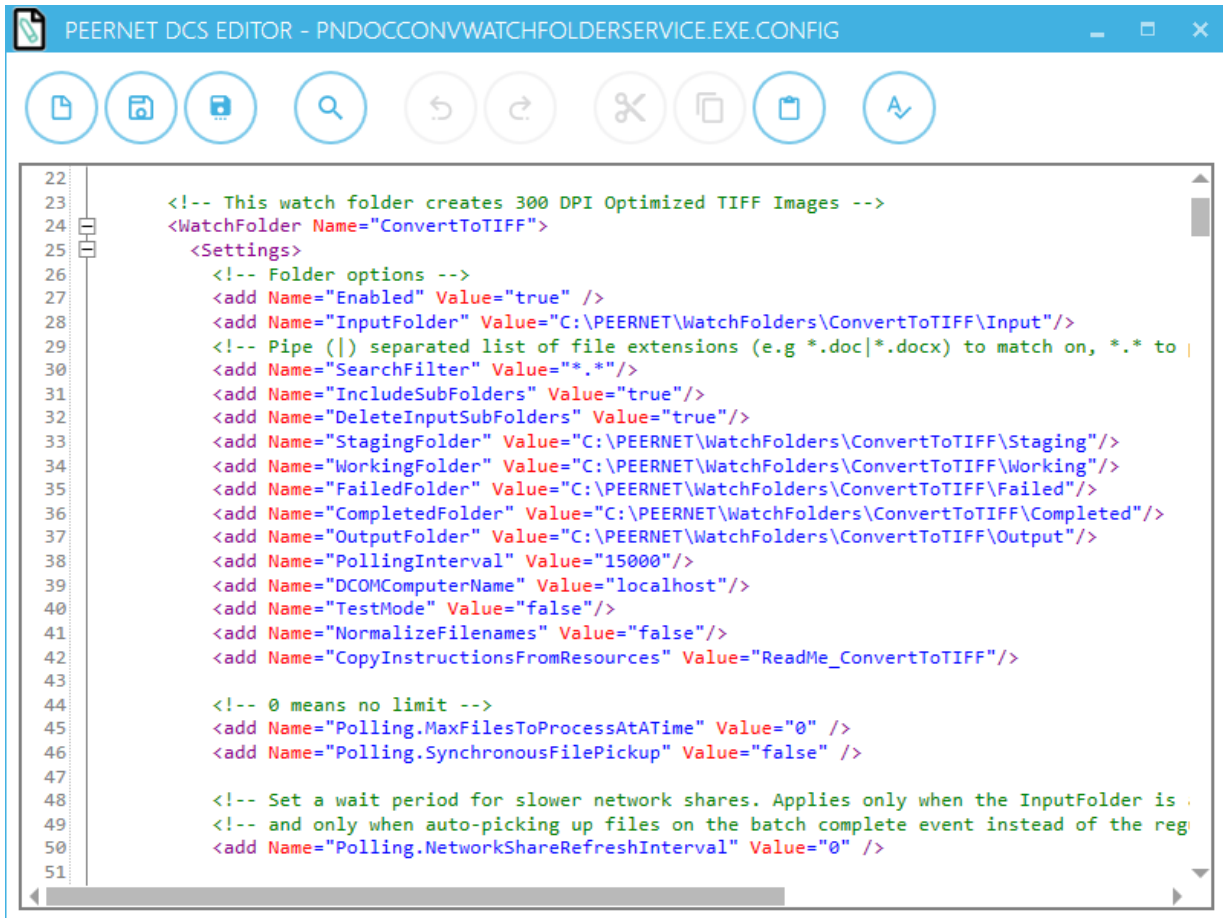
The Watch Folder Service can handle large volumes of files, high-throughput clustered file conversion, and supports processing archived Outlook messages and running post-processing commands, among many other options.



Clicking a tile above will take you to the help for that tile.

Edit Configuration

This tile opens the Watch Folder Service configuration file in the [DCS Editor](#). The configuration file is an XML-formatted file containing a WatchFolder definition for each drop folder you want to monitor.



Further Reading

See [The Watch Folder Service](#).

View Conversion Log

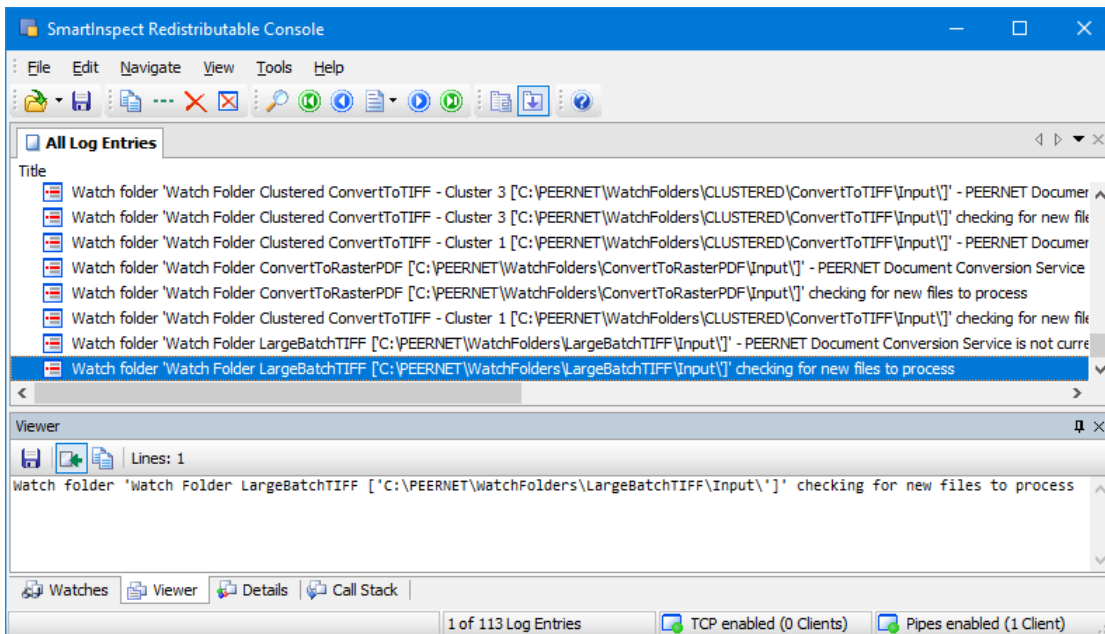
Watch Folder Service generates live logging while it is running. The log allows you to troubleshoot start-up issues, monitor the currently running service, and investigate any errors should they occur.

Clicking this tile opens the SmartInspect Redistributable Console used to view the log. When Watch Folder Service is running, live logging messages are displayed in the *All Log Entries* panel. The *Viewer* tab in the bottom panel shows the selected logging message details.



Do Not Leave the SmartInspect Redistributable Console Open

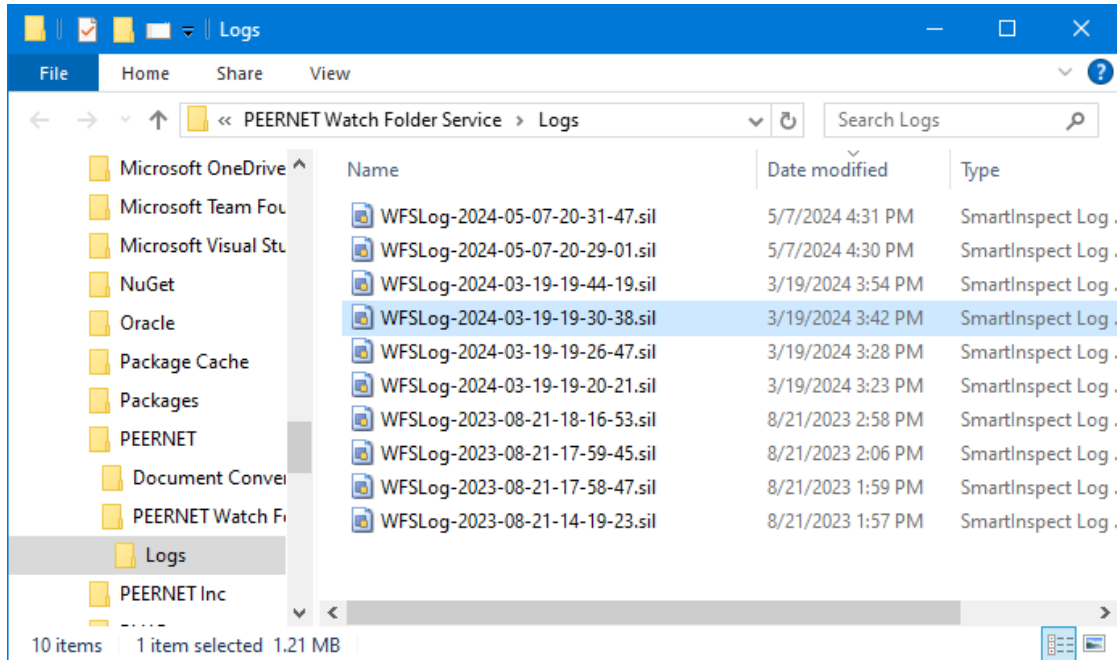
The SmartInspect Redistributable Console is meant for short term, live logging and troubleshooting access. Do not leave the logging console open for extended periods of time, such as overnight, or it will lock and cause issues.



Open Log Files Folder

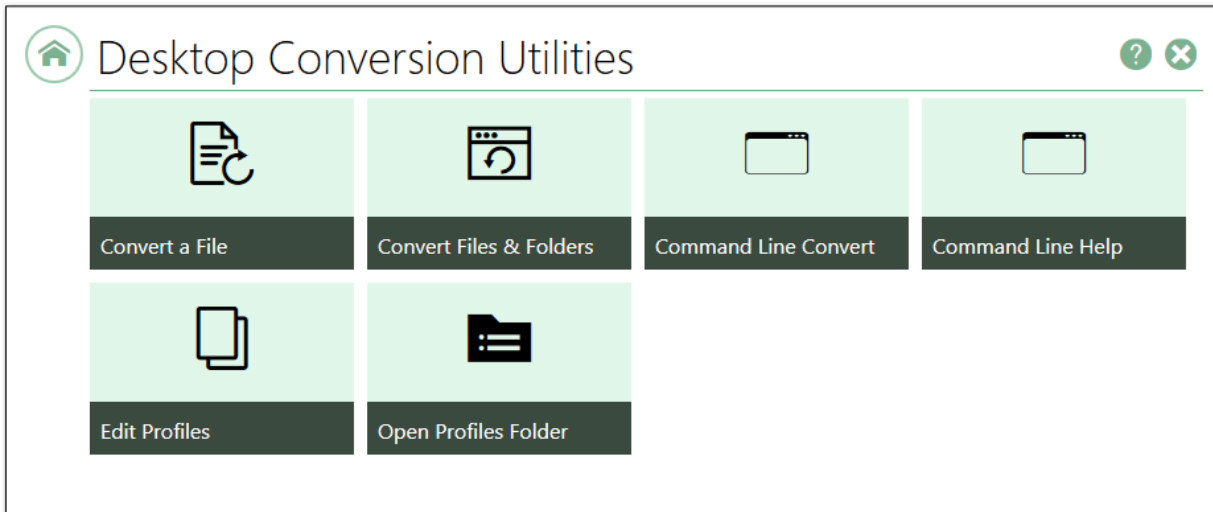
All logging messages are saved in a round-robin set of files, up to 10 in total, that are rotated based on size (250MB maximum) and by day. This allows you to keep a history for troubleshooting.

These log files all start with the name *WFSLog* followed by a date and time. Double-clicking a log file opens that file in a logging console window.



Desktop Conversion

The Desktop Conversion section contains links to all the desktop file conversion applications and the command line tools. It also has links to the profile editing tools for editing and creating the conversion profiles used by both the applications and command line tools.



Clicking a tile above will take you to the help for that tile.

Convert a File

The Convert File application converts a single file at a time. The profile chosen from a drop list of profiles determines the type of file created. This application includes advanced options to convert the file remotely on another computer (client-server document conversion).

Sample code for this application is part of the Document Conversion Service install.

The screenshot shows the 'PEERNET Convert File Sample' application window. It features several input fields and buttons for configuring a file conversion. The 'File to Convert' field is set to 'C:\Users\Michelle\Documents\factsheet.pdf' with a 'Browse...' button. The 'Output File Name' field is 'factsheet'. The 'Save in this Folder' field is 'C:\Users\Michelle\Documents\Documents' with a 'Browse...' button. The 'Convert to Type' dropdown is set to 'TIFF 200dpi OptimizedColor', and there is an unchecked 'Overwrite existing files' checkbox. A 'Remote Conversion Settings' section contains a checkbox for 'Conversion Service is running on this remote computer:' (unchecked), a text field with 'MAM-VM-WIN10-64', and two radio buttons: 'Use default shared folder location \\MAM-VM-WIN10-64\DCSREMOTE' (selected) and 'Use custom share location (\\<machine>\DCSREMOTE)'. Below these is an empty text field and a 'Browse...' button, with a note: 'Both the client and the server will need access to this folder.' At the bottom right are 'Convert File' and 'Exit' buttons. A 'Conversion Results:' label is above a large empty text area at the bottom.

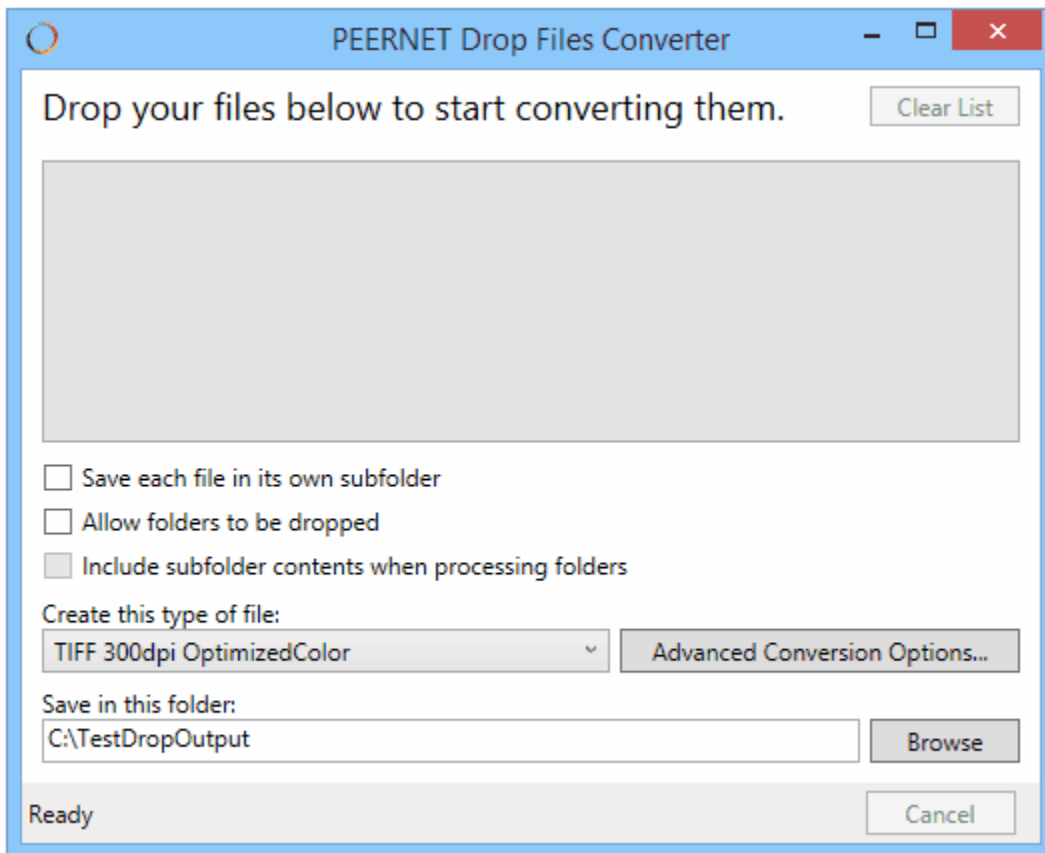
Further Reading

See [The Convert File Application](#).

Convert Files and Folders

Drag and drop files and, optionally, folders onto this application to convert them. Choose a profile from the list to determine the type of file to create, and select a directory to store the new files. The conversion starts when you drop a file or a collection of files onto the light gray drop area.

The Remote Conversion section allows to convert the file remotely on another computer (client-server document conversion).



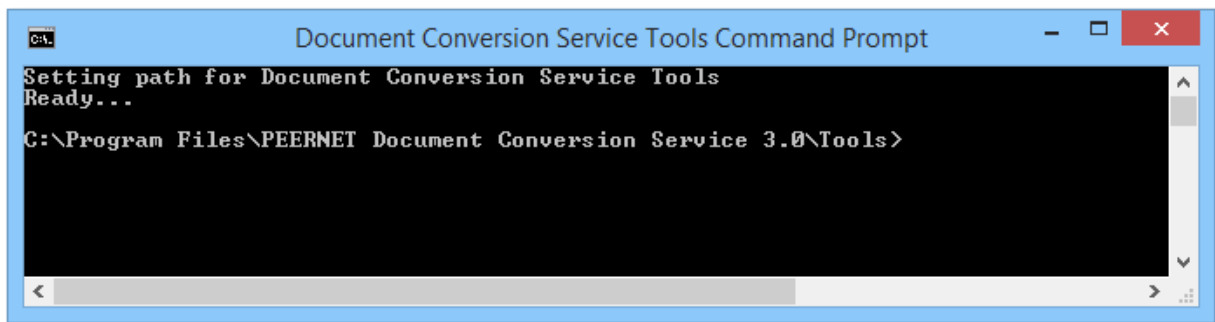
Further Reading

See [The Drop Files Converter Desktop Application](#).

Command Line Convert

Document Conversion Service includes several command line utilities for converting files and folders. Call these utilities from the DCS command window, scheduled tasks, from batch files or any program that can call an external program.

Clicking this tile opens the DCS command window. Run, test and experiment with the command line tools here.



Further Reading

See [Command Line Utilities](#).

Command Line Help

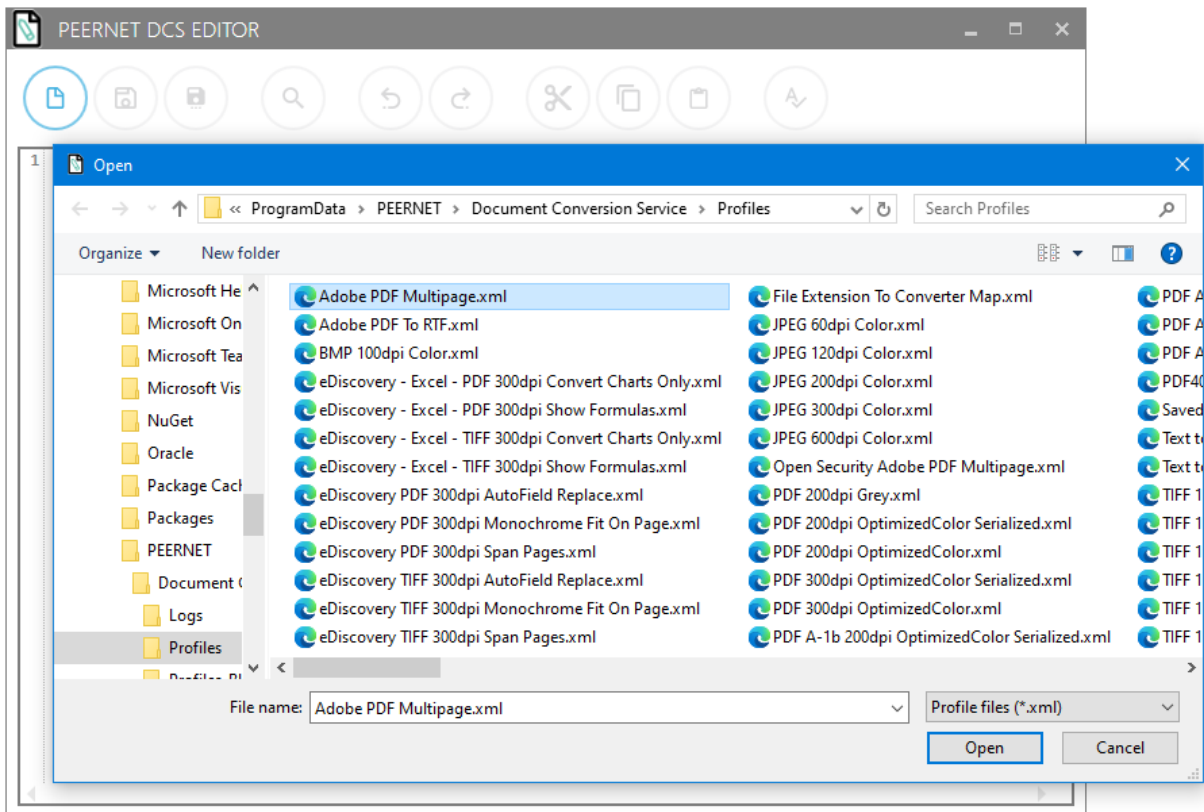
This tile opens the latest online Command Line Tools quick start guide. Learn about each command line utility, its options and see sample code you can use to get started.

Edit Profiles

Profiles are XML-formatted files containing a list of settings used when converting files. Used by the PEERNET.ConvertUtility.dll, the command line tools and the PNDocConvQueueServiceLib, they describe the type of output to create. These same settings are also used in the Watch Folder Service when setting up the drop folders.

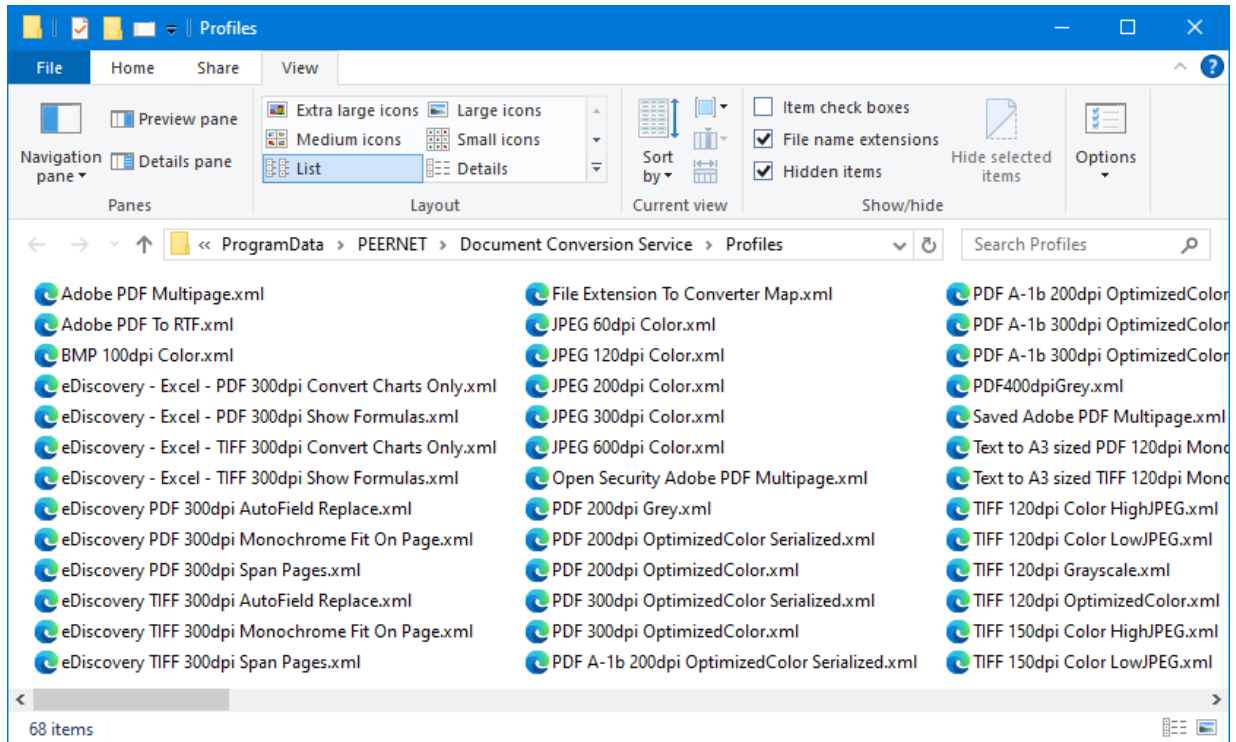
Document Conversion Service includes many sample profiles for creating TIFF, PDF and other file types. You can use these samples as a base to edit and create your own profiles with custom settings.

Click this tile to open, edit and create new profiles using the DCS Editor.



Open Profiles Folder

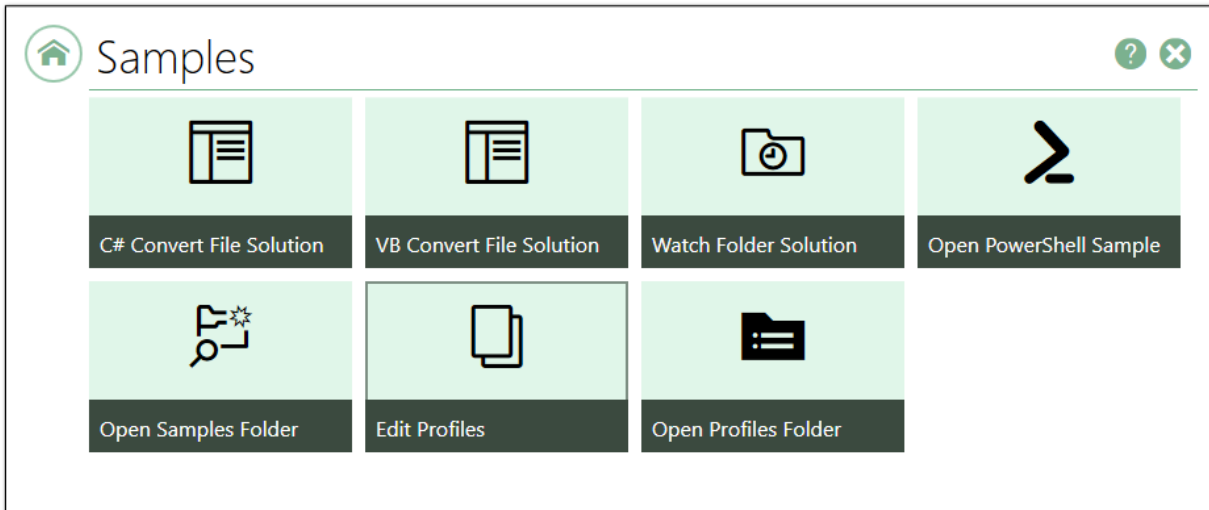
This tile opens the folder containing the sample profiles included with Document Conversion Service.



Samples

See how to integrate Document Conversion Service file conversion into your programs with our included samples, Watch Folder Service, and the Convert File desktop application.

These samples can be used on their own or as a starting point for integrating Document Conversion Service into your applications.



Clicking a tile above will take you to the help for that tile.

C# Convert File Solution

This C# sample demonstrates using the .NET library, [PEER.NET.ConvertUtility.dll](#), to send a file to Document Conversion Service for conversion to TIFF, PDF, or another format. It also shows how the PEER.NET.ConvertUtility.dll uses conversion profiles to say what output file type to create.

VB Convert File Solution

This sample program is the Visual Basic (VB) version of the C# Convert File sample above. Its only difference is the coding language used to write the sample program.

Watch Folder Solution

Our Watch Folder Service watches multiple drop folders. Each drop folder is configured to convert any file, or folder of files, dropped into that folder into a PDF, TIFF, or other file type. When you drop a file or folder of files into the folder, the service sends the files to Document Conversion Service.

Document Conversion Service converts the file using the output format set up for that folder. The converted file is returned and stored on the output location for that watch folder.

This advanced C#.NET sample demonstrates using the PNDocConvQueueServiceLib COM object from a service in a multi-threaded environment.

Open PowerShell Sample

Clicking this tile opens the included PowerShell sample in the Windows Powershell IDE. This PowerShell script demonstrates how to set conversion options and convert a file using the PEERNET.ConvertUtility.dll. After conversion, it shows how to read the returned results for the created file list or conversion errors.

Open Samples Folder

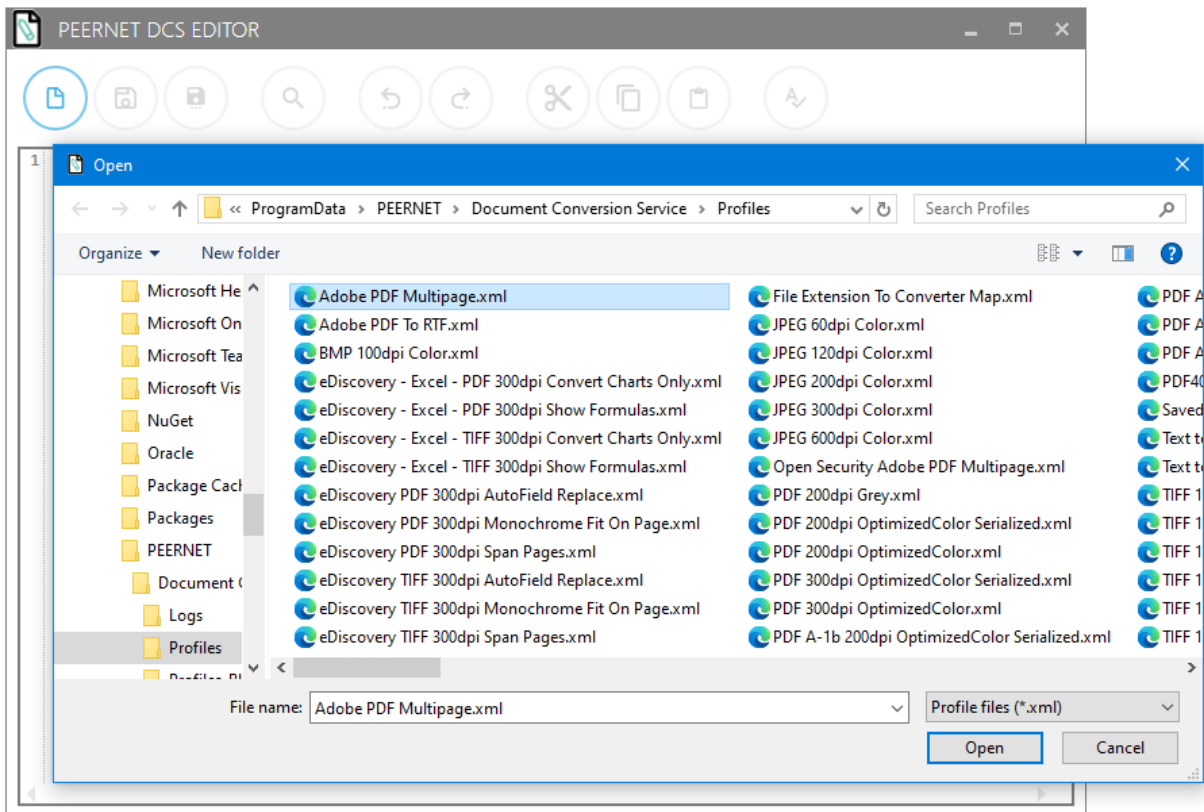
This tile opens the Samples folder containing all sample programs, scripts and solutions.

Edit Profiles

Profiles are XML-formatted files containing a list of settings used when converting files. Used by the PEERNET.ConvertUtility.dll, the command line tools and the PNDocConvQueueServiceLib, they describe the type of output to create. These same settings are also used in the Watch Folder Service when setting up the drop folders.

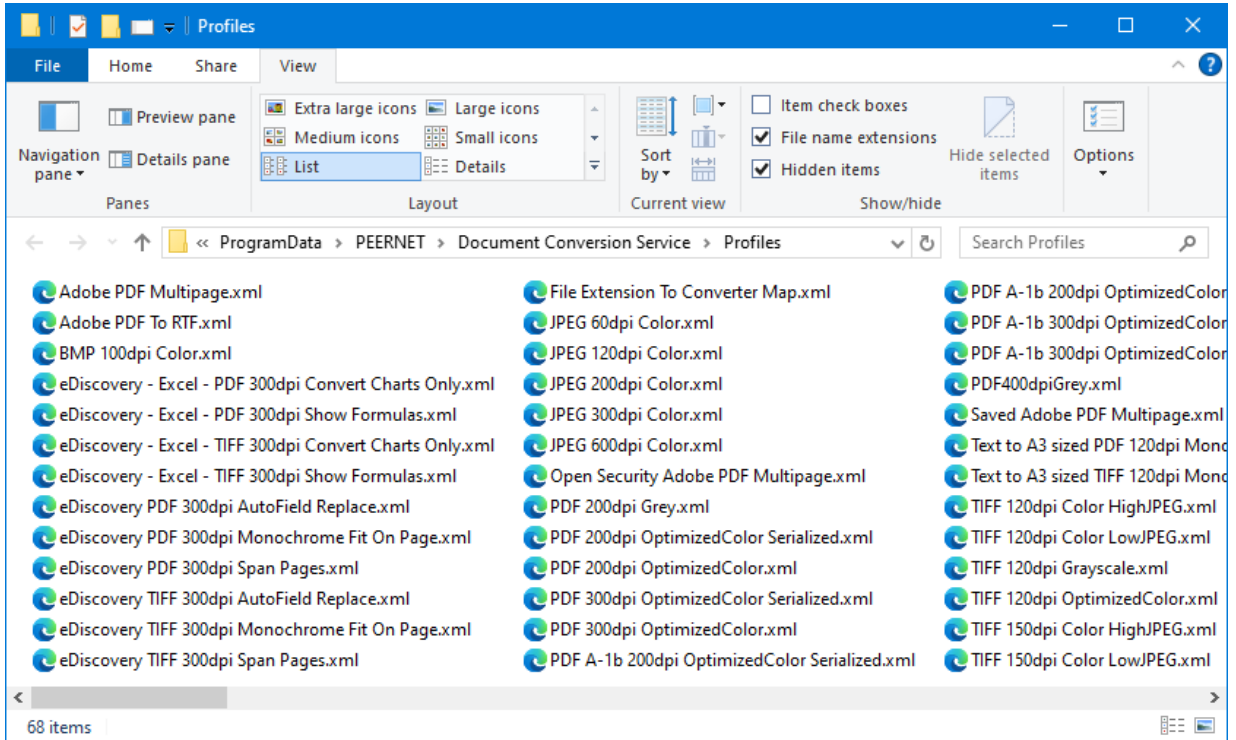
Document Conversion Service includes many sample profiles for creating TIFF, PDF and other file types. You can use these samples as a base to edit and create your own profiles with custom settings.

Click this tile to open, edit and create new profiles using the DCS Editor.



Open Profiles Folder

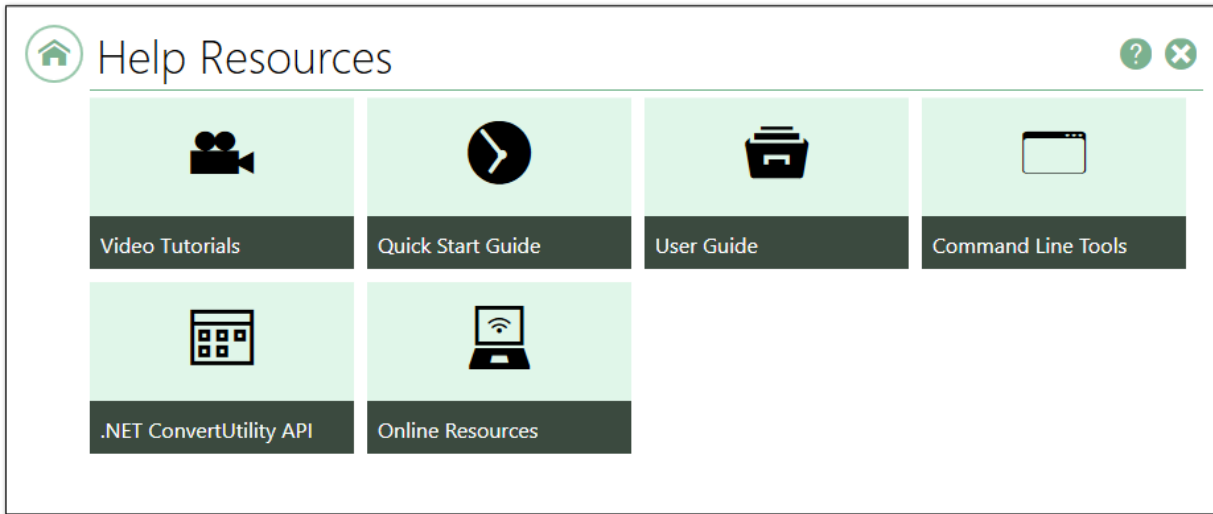
This tile opens the folder containing the sample profiles included with Document Conversion Service.



Help Resources

The Help Resources section contains convenient links to all our online help resources for **Document Conversion Service**.

Our video tutorials and quickstart guides will help you install, configure, and run Document Conversion Service and Watch Folder Service. For programmers, our [command line tools](#) and [.NET ConvertUtility API](#) have a complete programmer's reference.



Clicking a tile above will take you to the help for that tile.

Video Tutorials

Learn more about Document Conversion Service with our **Getting Started** video series, or go further and customize our software to your needs with our advanced tutorials. Our advanced videos cover topics from monitoring multiple input folders for files to convert to how to set up high-throughput clustered conversion using multiple computers.

Quick Start Guide

This short guide covers the basics of how to start and stop Document Conversion Service, how to convert a file, and what third-party applications you may need to have installed.

User Guide

The user guide explains all things Document Conversion Service. Whether you are a beginner or an advanced user, this document contains everything you need to get started and all the information you need for any advanced customization.

It covers installing, upgrading the product, and backing up custom configurations to restore later.

Learn how Document Conversion Service can convert files with sections for desktop conversion, monitored Watch Folders, command line utilities, and a programmable API.

Command Line Tools

Discover our collection of command line tools for converting and combining files and folders of files. Our command line tools can be used from any command prompt, as scheduled tasks, in batch files, or called from any programming language such as C#, VB, C++, Java, and PowerShell.

.NET ConvertUtility API

Add document conversion into your own .NET programs with our PEER.NET.ConvertUtility.dll included with Document Conversion Service. Easily convert files, folders, dynamic lists of files, or combine multiple files into a single TIFF or PDF.

Online Resources

All our quick start, user guide, and programmer's reference documentation are here.

Installing and Updating Document Conversion Service

This topic covers the following tasks:

- [Download and Install Document Conversion Service](#)
- [Backup and Restore Settings When Upgrading](#)
- [Installing Document Conversion Service Silently](#)

Installing Document Conversion Service

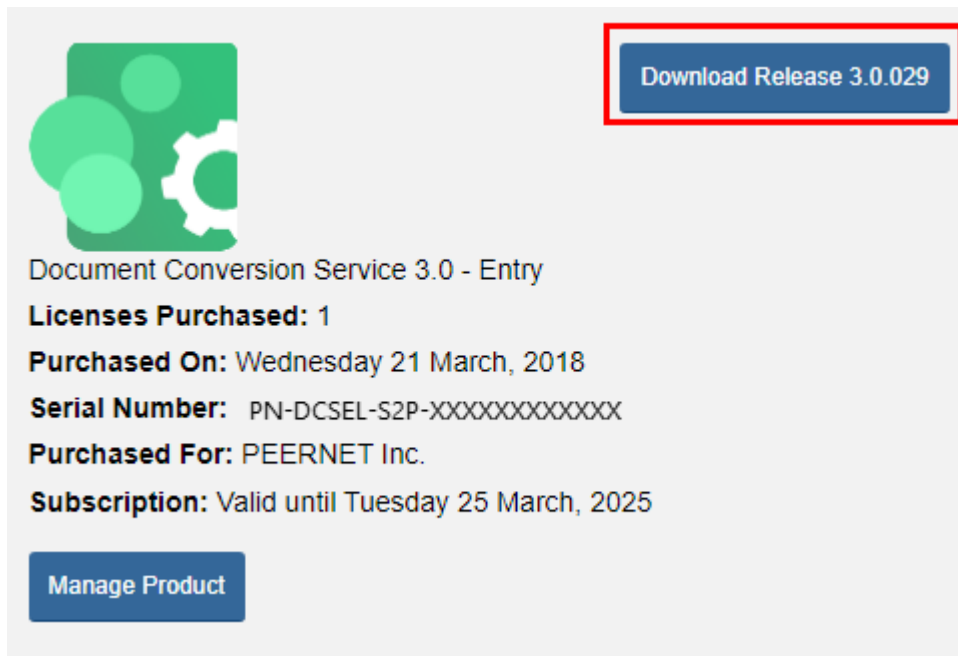
The first step to installing Document Conversion Service is to download the latest version of the software which is available through your online account. If you installed a trial version of Document Conversion Service before purchasing, you still need to download and install your purchased copy of the software.

Any modifications made to configuration files for Document Conversion Service, Watch Folder Service and any edited conversion profiles are backed up and restored as part of the installation process. See below on how to create your backup and restore your settings [During the Installation Process](#).

Log into your [PEERNET online account](#) and find your recently purchased software in the **My Products** list. Select the **Download Release** button to download the latest release of the software. Your Download Release button may have different text, depending on when you purchased and the latest available release version.

Once you have downloaded the latest release, double-click the pndscsetup_3.0.###.exe to run the install.

If you are upgrading from a previous version and need to keep modifications made to configuration files for Document Conversion Service, Watch Folder Service and any edited conversion profiles, these are backed up and restored as part of the installation process. Learn how to create your backup and restore your settings [During the Installation Process](#).



Download Release 3.0.029

Document Conversion Service 3.0 - Entry

Licenses Purchased: 1

Purchased On: Wednesday 21 March, 2018

Serial Number: PN-DCSEL-S2P-XXXXXXXXXXXXXX

Purchased For: PEERNET Inc.

Subscription: Valid until Tuesday 25 March, 2025

Manage Product

Upgrading from a Previous Version

When upgrading Document Conversion Service to a new version, (i.e. upgrading from version 2.0 to 3.0), or updating within the current version (i.e. updates from version 3.0.001 to 3.0.002), these updates and upgrades are done in-place and will set all application settings back to the installation defaults.

If you have made any changes to the settings they will need to be reapplied. This includes changes to the following:

- The Document Conversion Service configuration file
- The Watch Folder Service configuration file
- Any changes you made to the default conversion profiles, or any new conversion profiles you created. Conversion profiles are used by the command line tools, the PEERNET.ConvertUtility.dll and the Convert File sample programs.

The section, [How to Backup and Restore Configuration Files and Profiles](#) shows how to backup and restore these files when updating or upgrading the software.

Installing Document Conversion Service Silently

For users who need to push the product using software management tools, Document Conversion Service can be installed silently with no user interaction. See the section [Installing Document Conversion Service Silently](#) for the command line parameters and steps.

How to Backup and Restore Configuration Files and Profiles

The configuration files for Document Conversion Service, Watch Folder Service and any edited conversion profiles need to be backed up and restored when:

- you are upgrading to a newer version of Document Conversion Service, i.e updating from version 2.0 to 3.0
- you are installing a minor version update of Document Conversion Service, i.e. updating from version 3.0.001 to 3.0.002
- you are moving the current Document Conversion Service install to a new server
- you want to keep a backup copy saved for future reference

Updates and upgrades are done in-place and will set all application settings back to the installation defaults. If you have made any changes to the configuration files or profiles, they will need to be saved and the changes reapplied. There are a few ways to do this.

- [During the Installation Process](#) - Beginning with Document Conversion Service 3.0.017, when upgrading a previous version, the install program includes options to create a backup zip file containing the Document Conversion Service configuration file, the Watch Folder configuration file and the conversion profiles. At the end of the upgrade process, the contents of the backup zip file can be restored using the new DCS Backup and Restore utility
- [Using the Backup and Restore Tool](#) - This tool can be used to create a backup as well as restore a backup created with this tool or by the installation program. For users prior to version 3.0.017, a stand-alone copy of the DCS Backup and Restore utility can be downloaded by contacting [PEERNET Support](#)
- [Manually Backup or Restore the Files](#) - the configuration files and profiles can always be backed up and restored manually.
- [Using the Configuration Merge Tool](#) - merge manually saved DCS and Watch Folder configuration files or conversion profiles through a visual GUI with preview, syntax highlighting and error reporting.

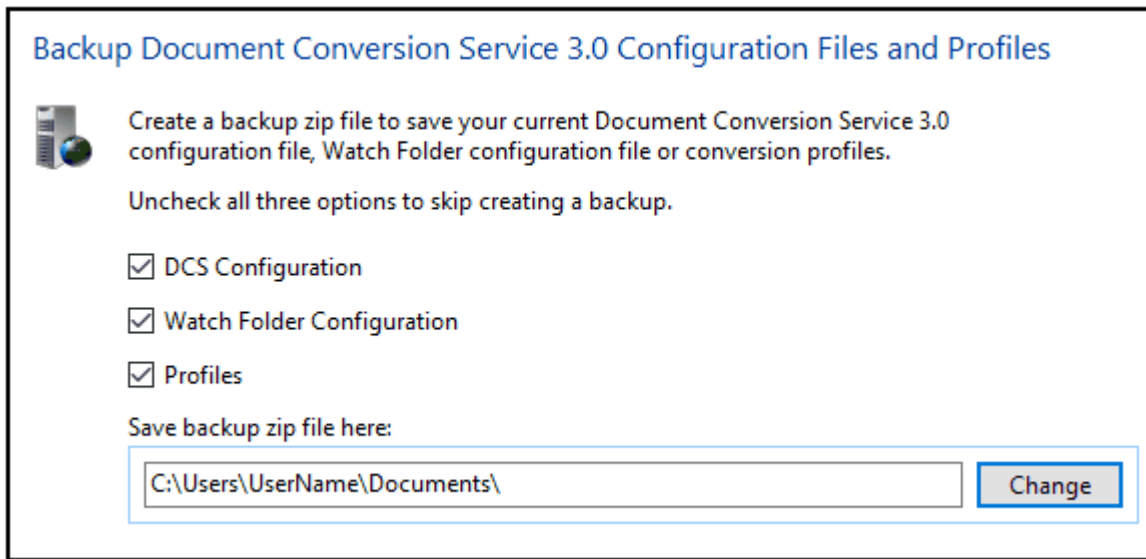
During the Installation Process

Beginning with Document Conversion Service 3.0.017, updating or removing an existing installation now includes an option to create a backup zip file containing the Document Conversion Service configuration file, the Watch Folder Service configuration file and the conversion profiles.

Create a Backup During Installation

This option is shown when running the install for Document Conversion Service 3.0.017 or later when upgrading over an existing installation, or when running the install to remove the current installation.

To skip creating a backup file, uncheck all three options.



Backup Document Conversion Service 3.0 Configuration Files and Profiles

Create a backup zip file to save your current Document Conversion Service 3.0 configuration file, Watch Folder configuration file or conversion profiles.

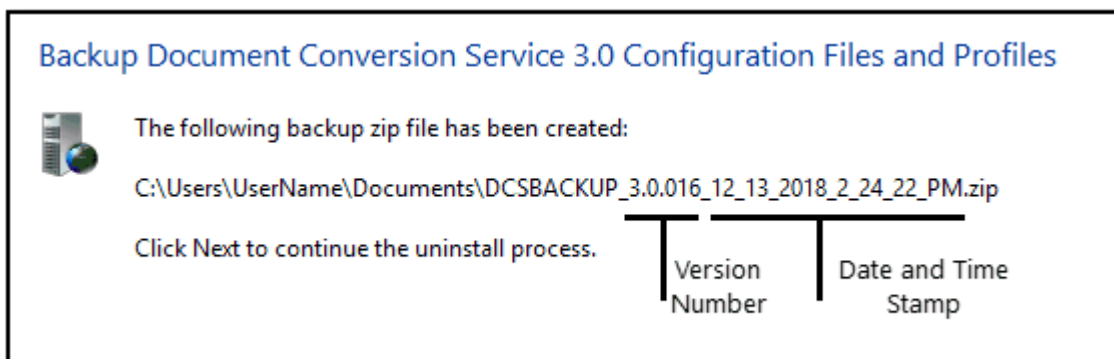
Uncheck all three options to skip creating a backup.

- ☒ DCS Configuration
- ☒ Watch Folder Configuration
- ☒ Profiles

Save backup zip file here:

C:\Users\UserName\Documents\ Change

The file is created in the Documents folder by default but can be stored in a different location if desired. The backup file is named using the currently installed version number and the current date and time. From this point, click Next to continue with the installation.



Backup Document Conversion Service 3.0 Configuration Files and Profiles

The following backup zip file has been created:

C:\Users\UserName\Documents\DCSBACKUP_3.0.016_12_13_2018_2_24_22_PM.zip

Click Next to continue the uninstall process.

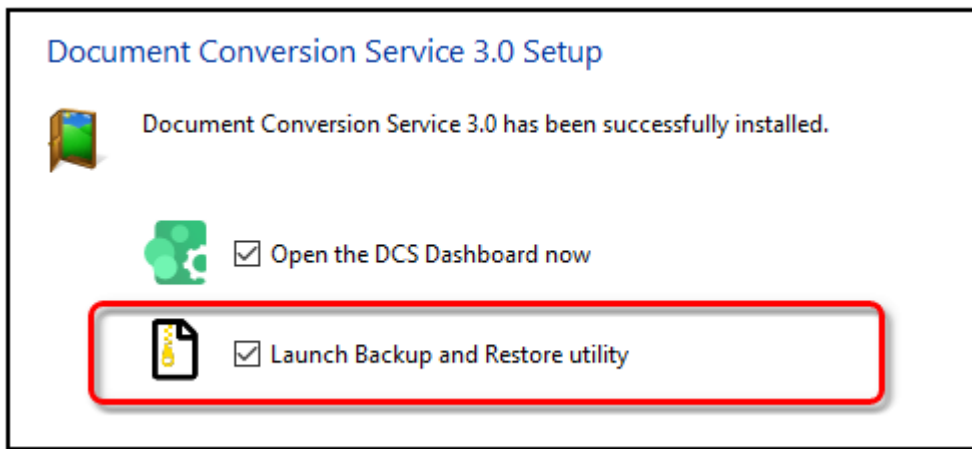
Version Number	Date and Time Stamp

Restore a Backup at the End of Installation

When upgrading older versions, the install is preset to launch the DCS Backup and Restore utility at the end of the installation process if you have created a backup zip file in an earlier step of the install. When launched, the DCS Backup and Restore tool will automatically open the backup zip file created as part of the installation upgrade process.

If you are installing Document Conversion Service for the first time on a new machine, the option to restore any saved backup zip will not be enabled by default. If you have a backup zip file from an earlier install or another computer, check this option to run the utility and load your saved file.

See the next section, [Using the Backup and Restore Tool](#) for steps on restoring your files.



Using the Backup and Restore Tool

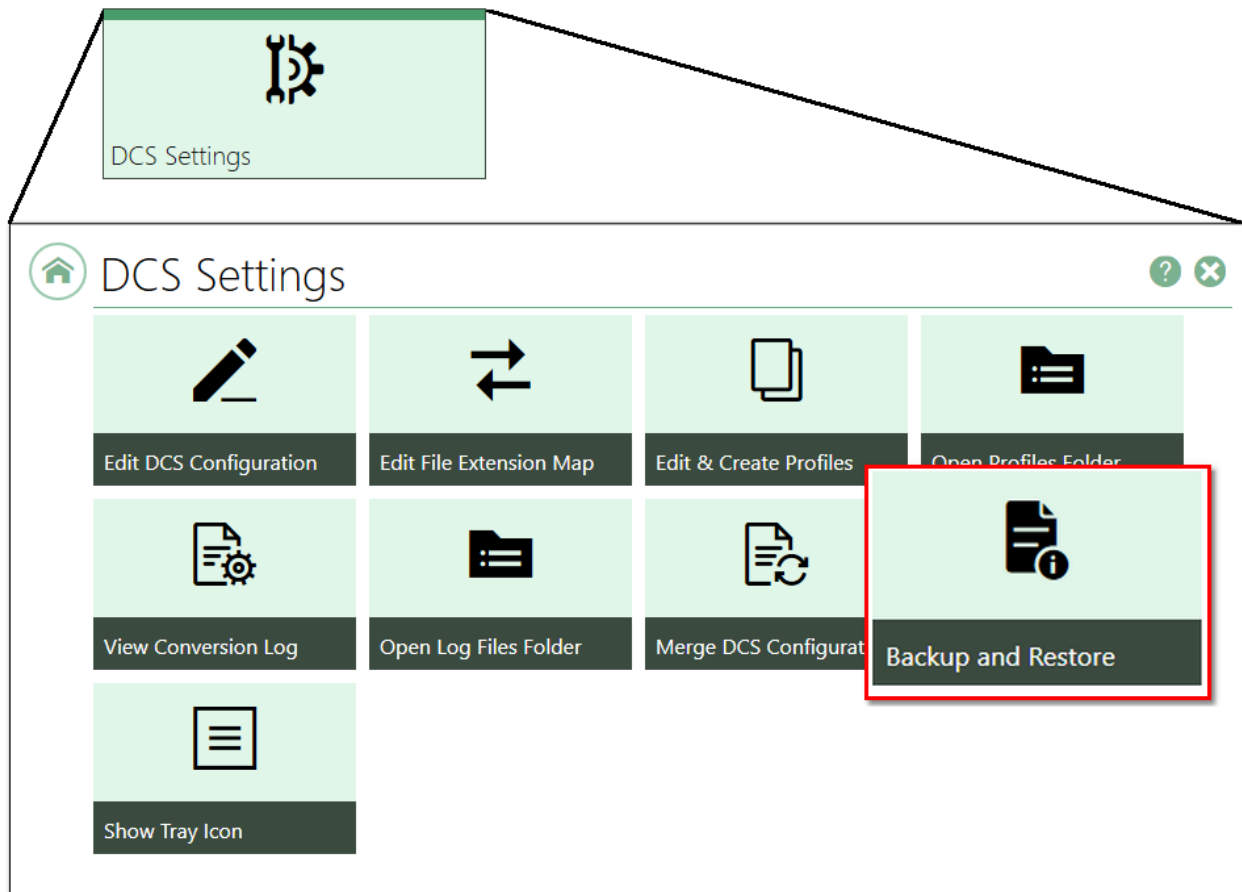
The DCS Backup and Restore utility can backup or restore your Document Conversion Service configuration file, Watch Folder Service configuration file and any created or editing conversion profiles.

Available starting with Document Conversion Service 3.0.017, users running earlier 3.0 versions can get access to the tool as a stand-alone executable by contacting [PEERNET Support](#). This tool does not work with version 2.0 or earlier.

When backing up files, it creates a zip file containing the configuration files and conversion profiles you select to back up. The restore process will use the created zip file to merge the your configuration files with the new ones, and update the conversion profiles collection with the saved ones.

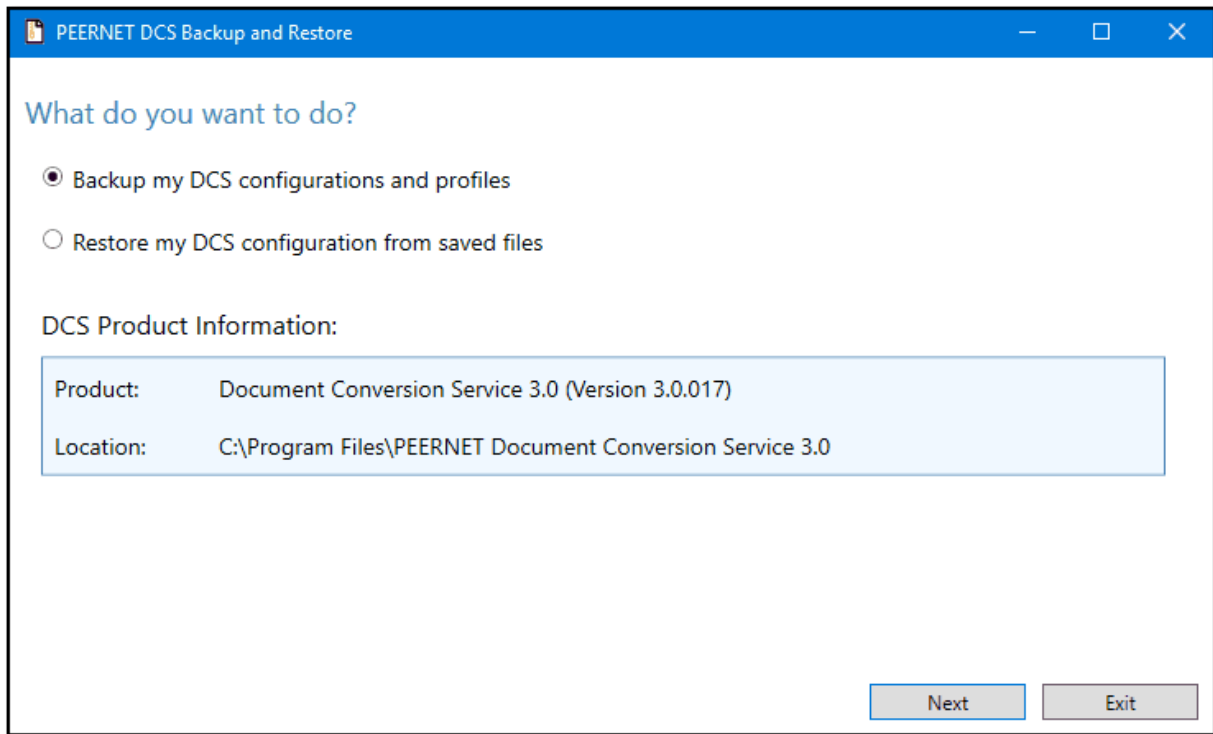
To launch the DCS Backup and Restore tool, open the DCS Dashboard. Click the **DCS Settings** tile, then the **Backup and Restore** tile.

This tool can also be accessed from the Start menu. Go to **Start - Document Conversion Service 3.0 – DCS Backup and Restore**.



The tool detects what installed version of Document Conversion Service you are running and allows you to choose to:

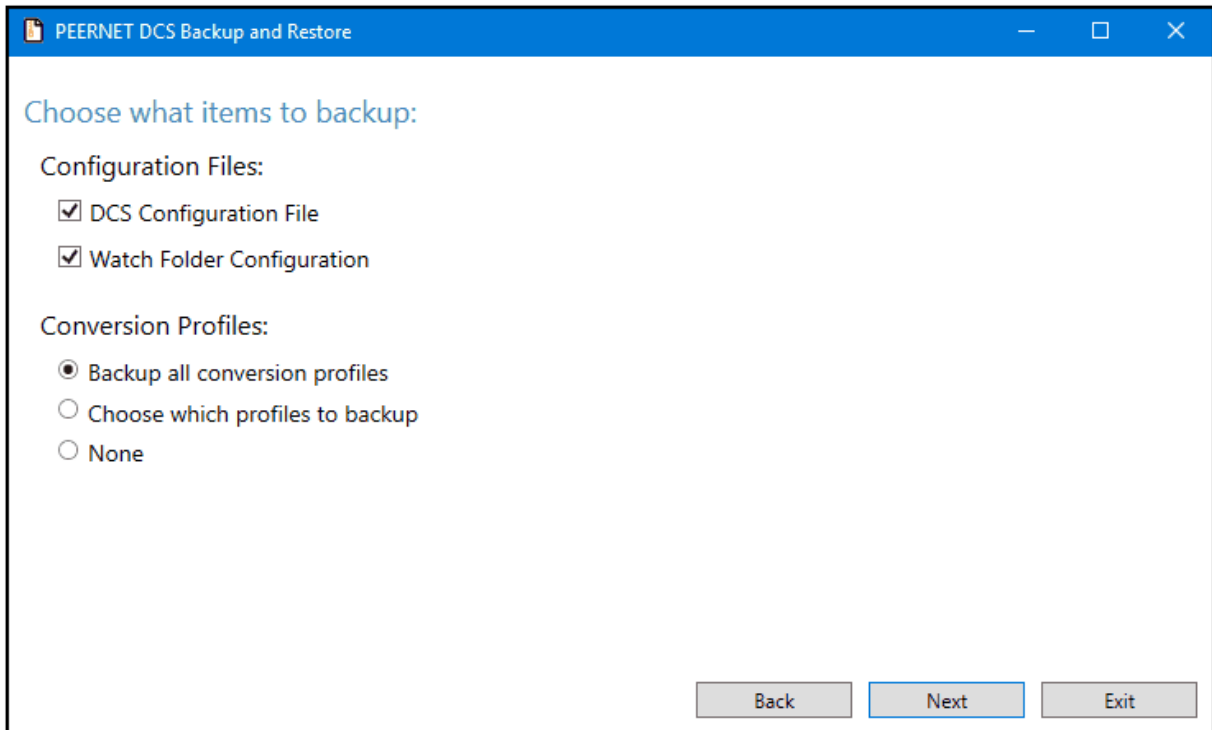
- [backup your DCS configurations and profiles](#)
- [restore your DCS configuration from a saved backup](#)



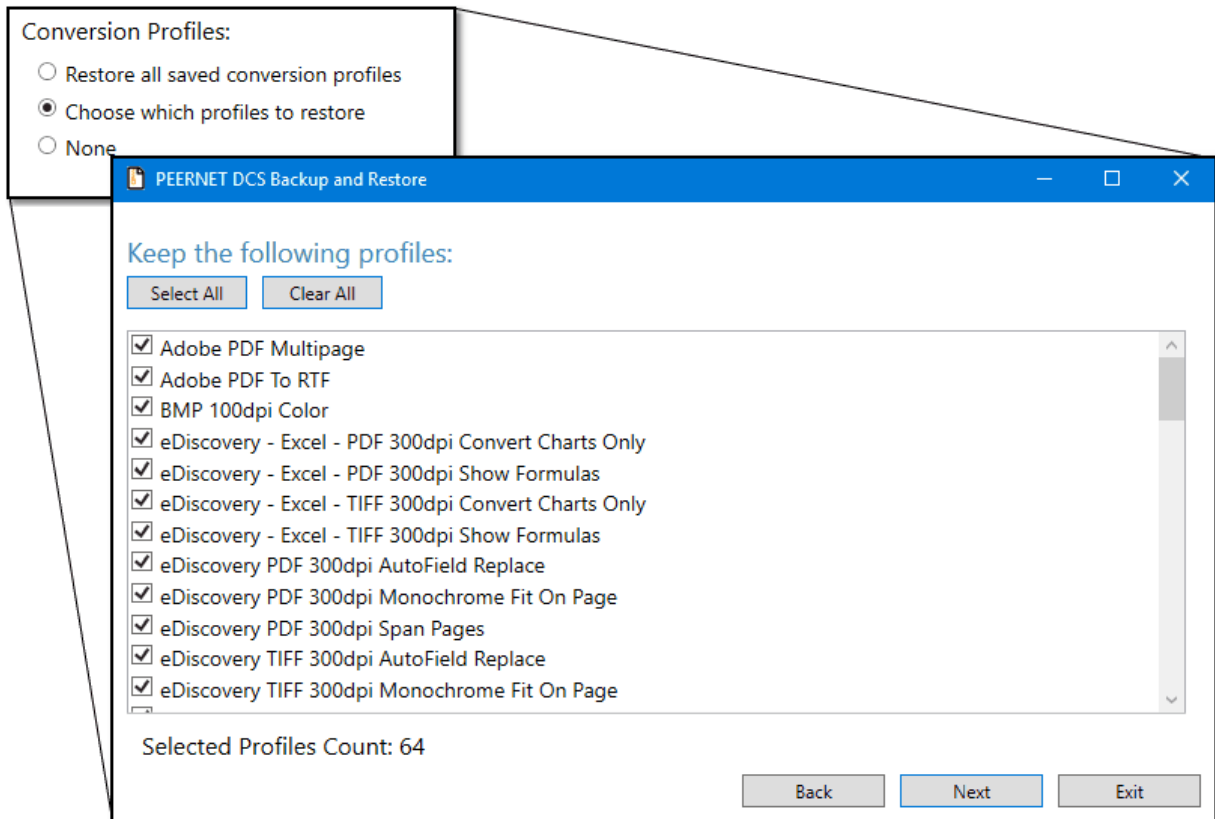
Backing up Your Configuration Files and Profiles

To backup your configuration files and conversion profiles, select **Backup my DCS configurations and profiles**.

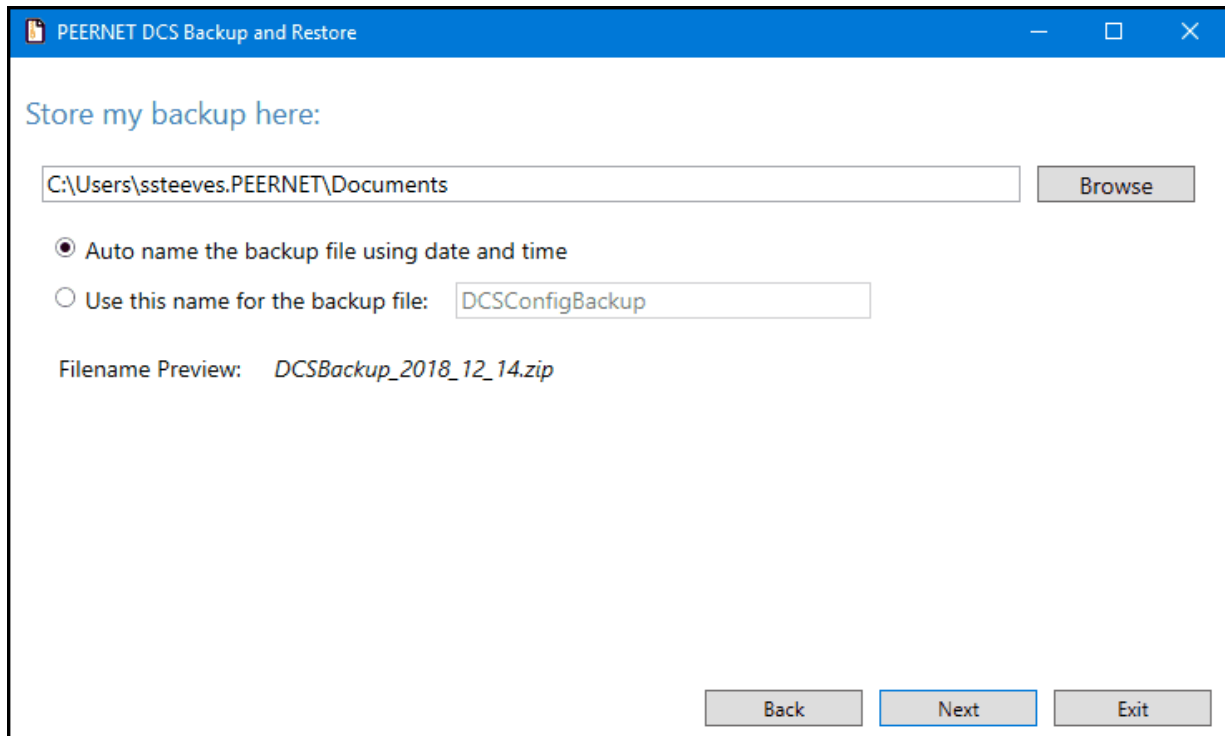
The first step is to choose which items to back up. The default is to back up both the DCS configuration file and Watch Folder configuration file as well as all conversion profiles. If you are unsure which of those files you may have modified, leave everything checked. If you know you have only made changes to specific files, you can choose to only save those files.



If you know you have only modified and/or added certain conversion profiles, use the **Choose which profiles to backup** option to only backup certain profiles.

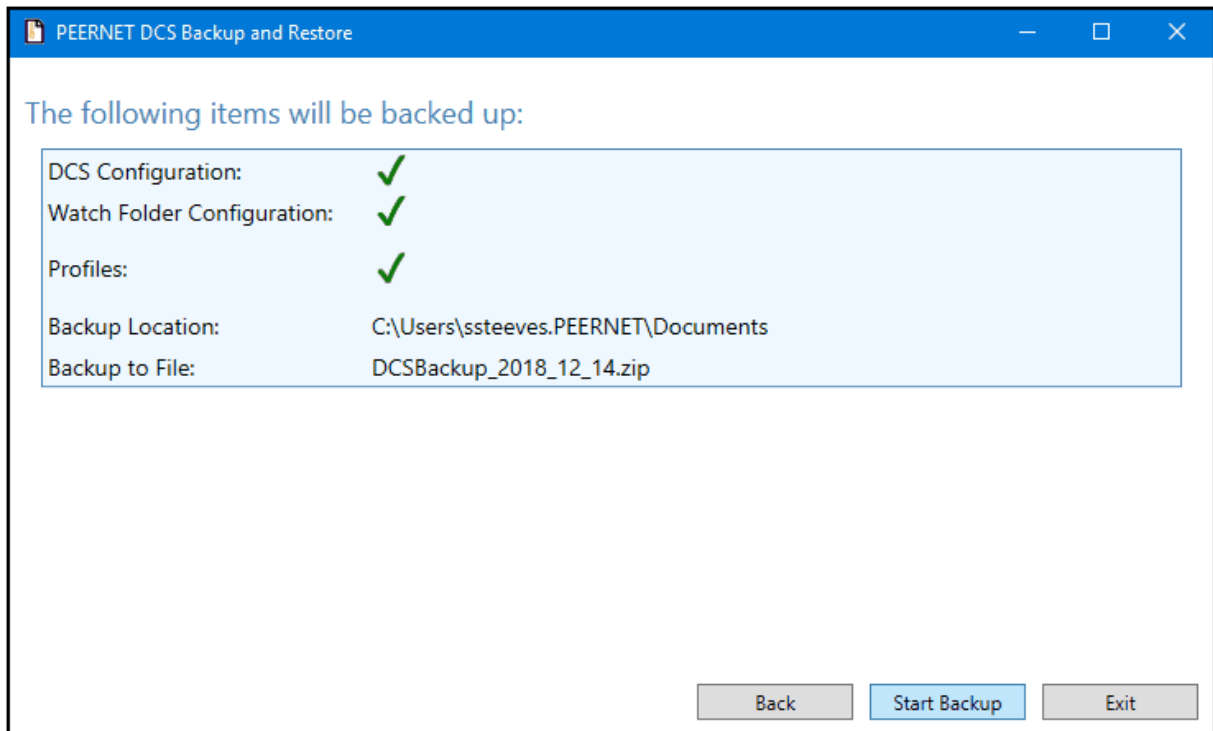


Next, choose where to save and how to name the backup zip file. You can let the tool auto-name the file and save it in the Documents folder, or provide your own name and location.



The screenshot shows a Windows-style dialog box titled "PEERNET DCS Backup and Restore". The main heading is "Store my backup here:". Below this, there is a text input field containing the path "C:\Users\ssteeves.PEERNET\Documents" and a "Browse" button to its right. Underneath, there are two radio button options: "Auto name the backup file using date and time" (which is selected) and "Use this name for the backup file:". The second option has a text input field next to it containing "DCSConfigBackup". Below these options, a "Filename Preview:" label is followed by the text "DCSBackup_2018_12_14.zip". At the bottom right of the dialog, there are three buttons: "Back", "Next" (which is highlighted with a blue border), and "Exit".

Review your selections before starting the backup. Any included item will have a green check mark. The backup location and file name are also shown. Click **Start Backup** to save the backup zip file.

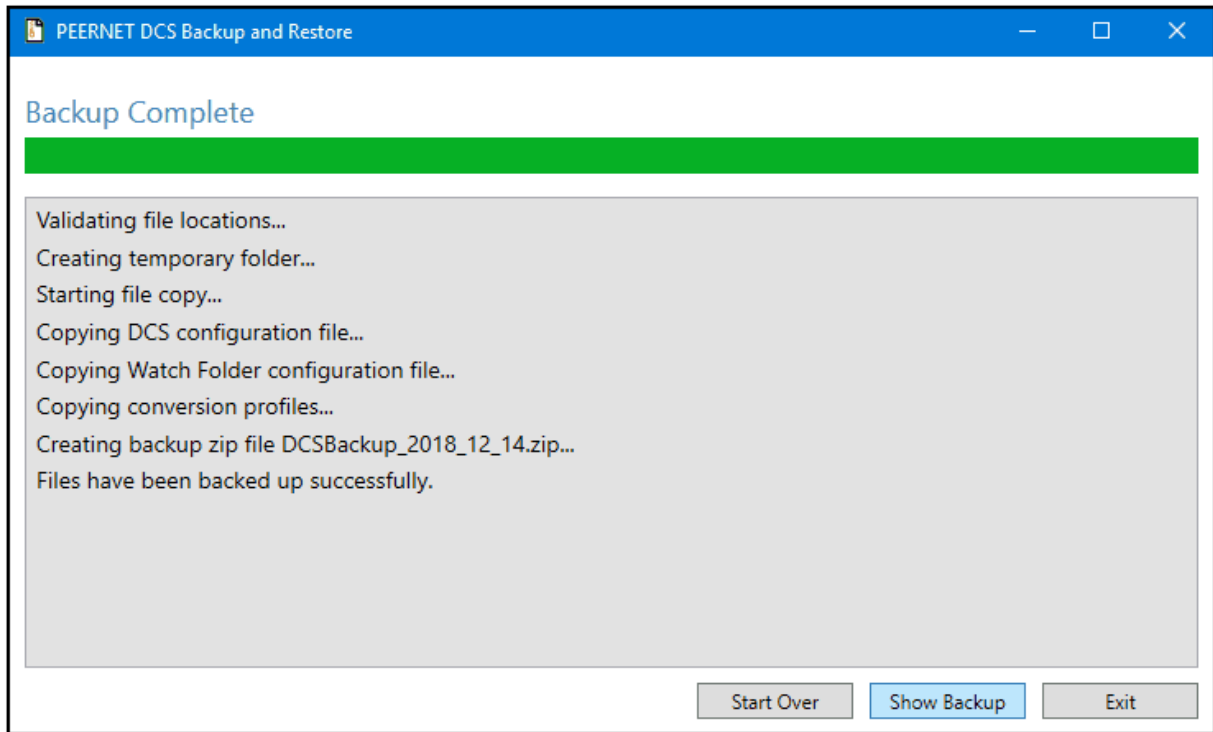


The Backup screen shows the progress as the backup file is created.

When complete, you can view your new backup file using the **Show Backup** button. It will open a Windows Explorer window with the file highlighted.

If you missed a file, you can create a new backup using the **Start Over** button.

Close the tool by clicking **Exit**.

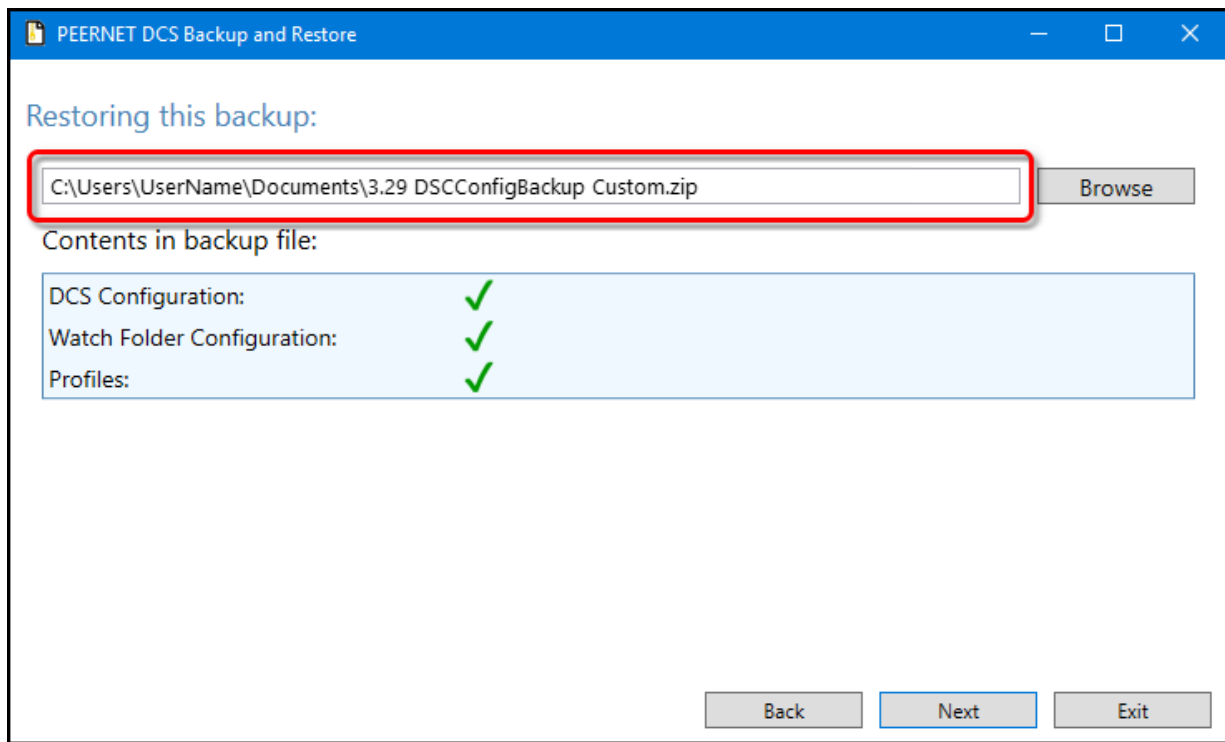


Restoring your Configuration Files and Profiles

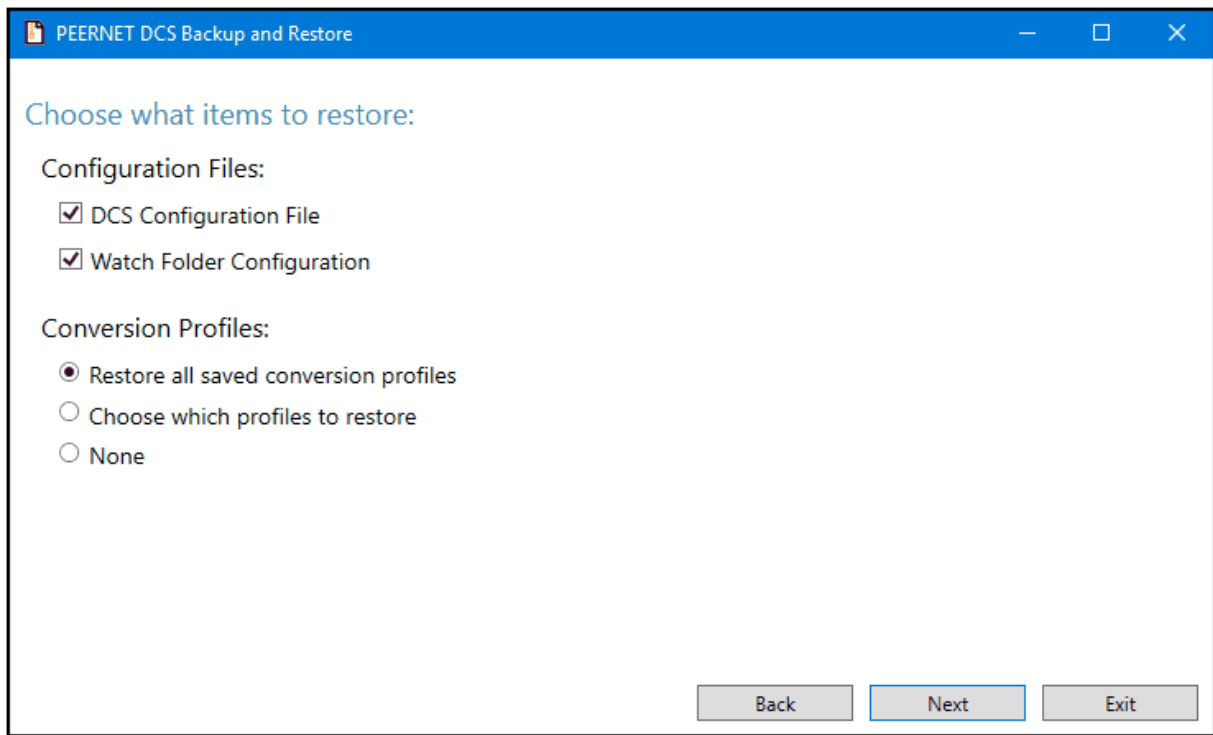
If you are running the DCS Backup and Restore utility from the installation program, and have created a backup zip file as part of the install process, the tool will open with that file already loaded.

If you are running this tool manually, such as when migrating the settings to a new computer, choose **Restore my DCS configuration from saved files** to start a restore. Use the **Browse** button to find and load your backup zip file.

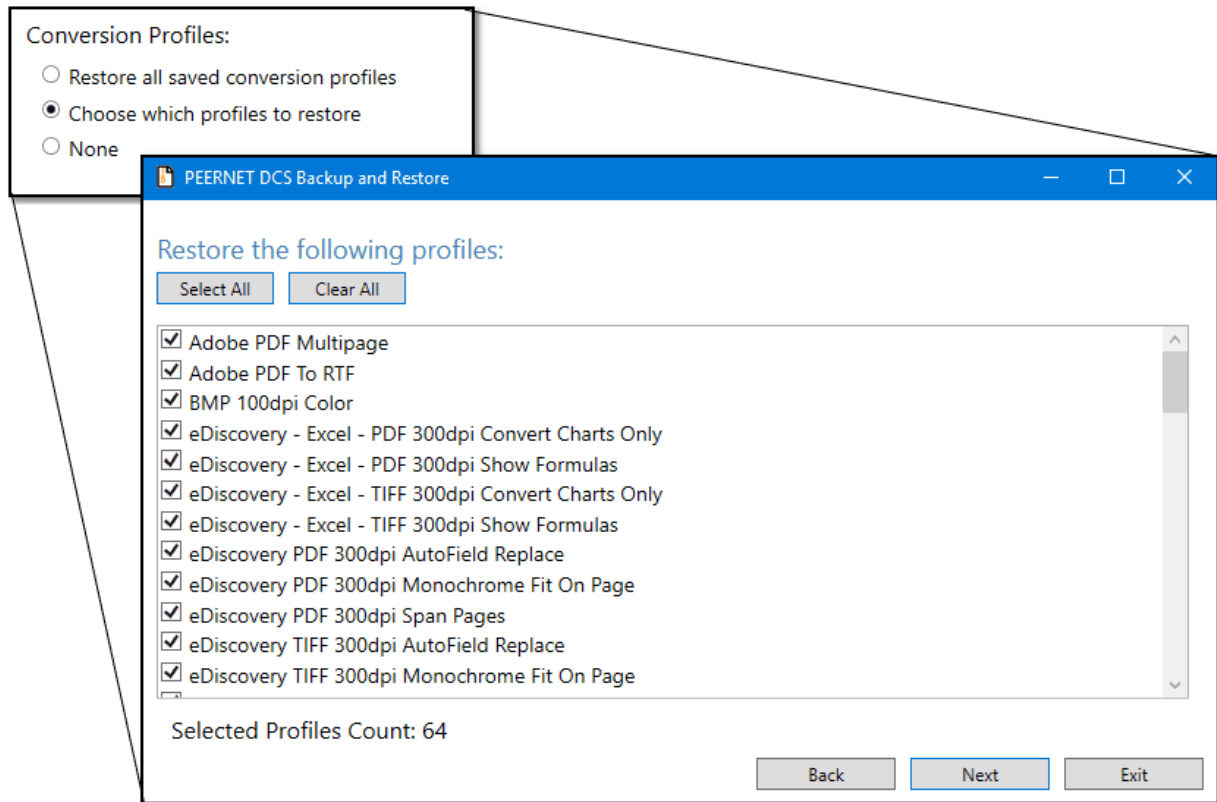
The file will be validated and show a summary of which files are contained in the backup zip file. Click **Next**.



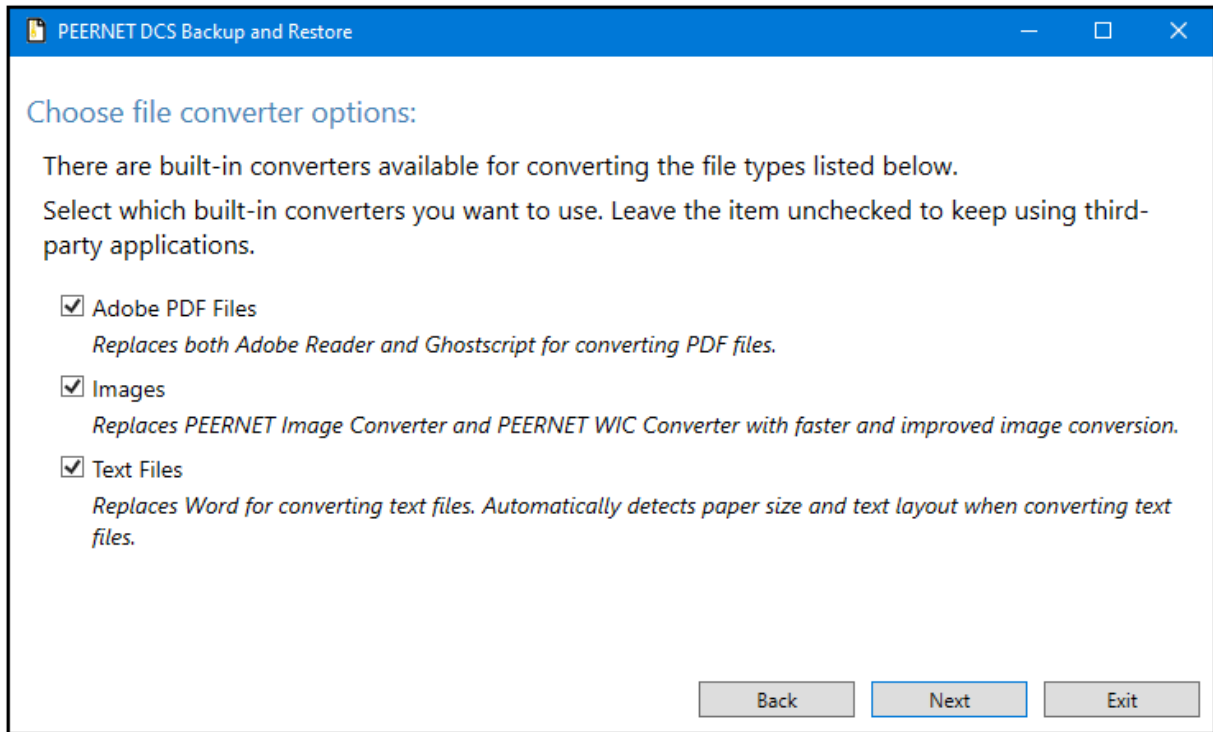
The next step is to choose which items to restore. Items that are not in the backup zip file will be disabled and cannot be selected. For instance, if the backup zip file only contained a DCS configuration file and conversion profiles, the **Watch Folder Configuration** option would be disabled.



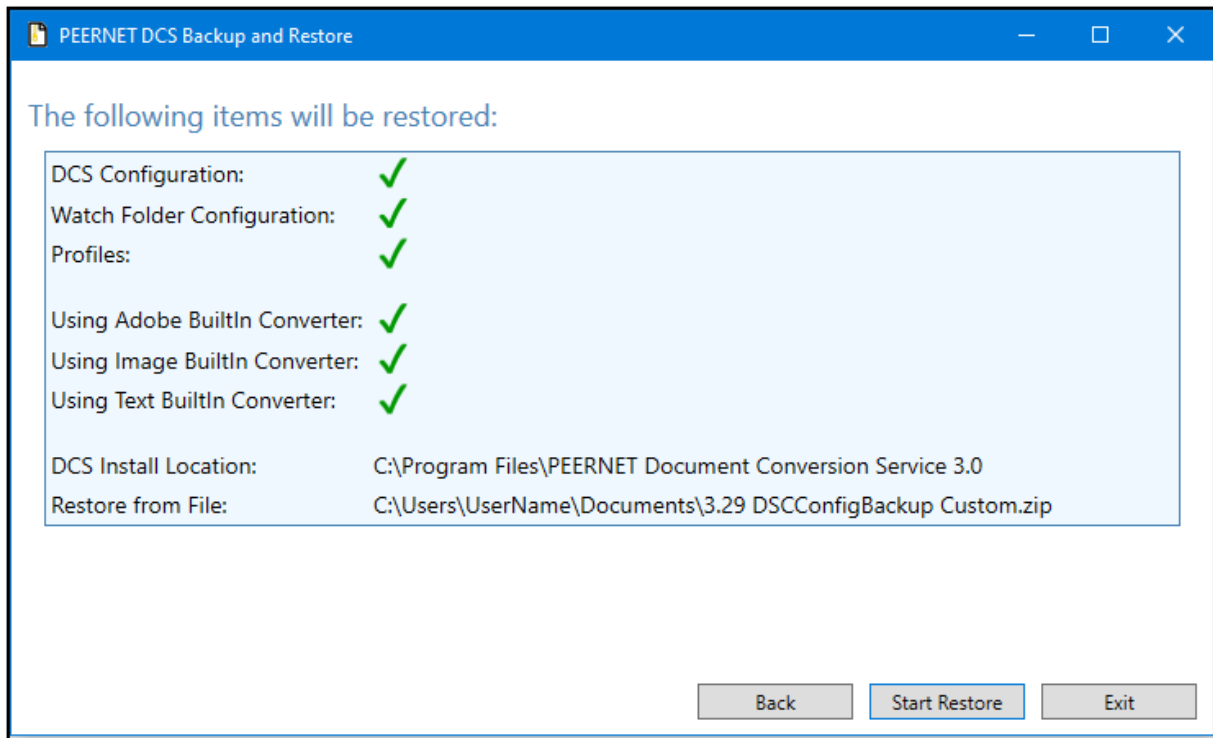
If there are conversion profiles in the backup, and you don't want to restore all of them, use the **Choose which profiles to restore** option to only restore certain profiles.



Starting with Document Conversion Service 3.0.031, new built-in converters for Adobe PDF files, text documents, and images files are available. These converters do not require any third-party applications to be installed, and offer improvements for conversion speed and support for converting new file types. Running a restore in previous versions of Document Conversion Service will not show this option.



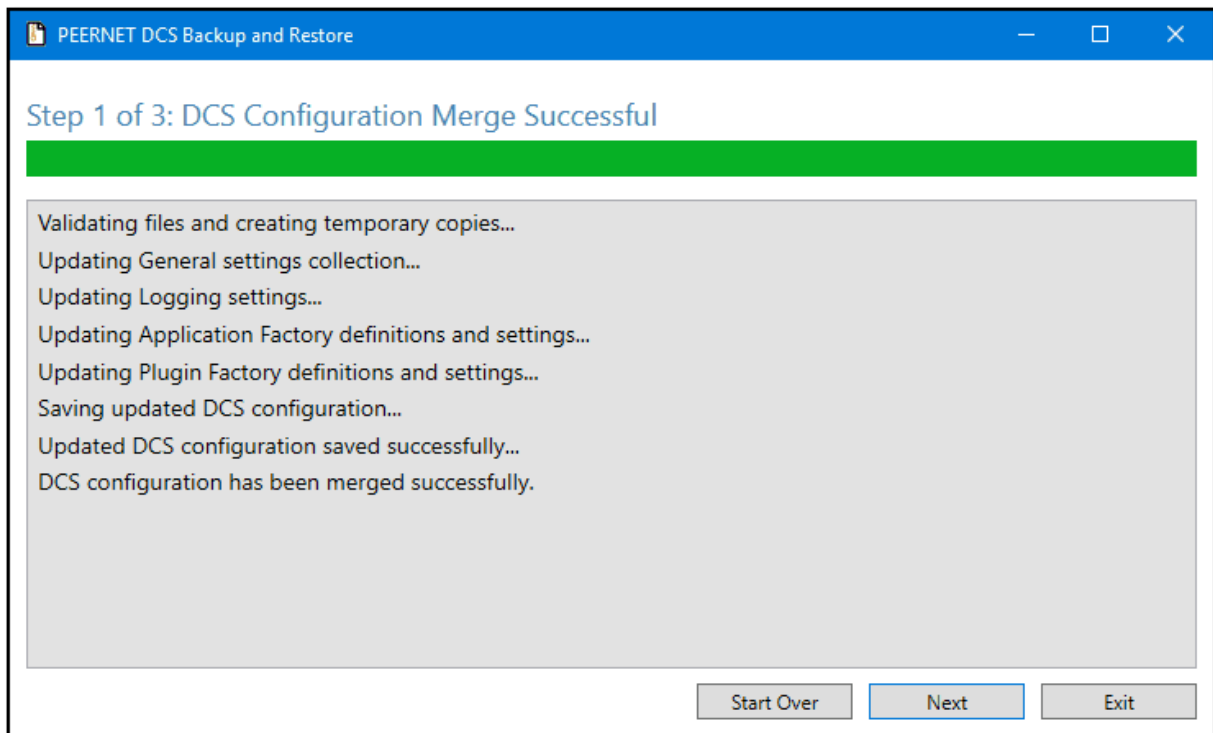
Review your selections before starting the backup. Any included item will have a green check mark. Click **Start Restore** to begin restoring the selected files from the chosen backup zip file.



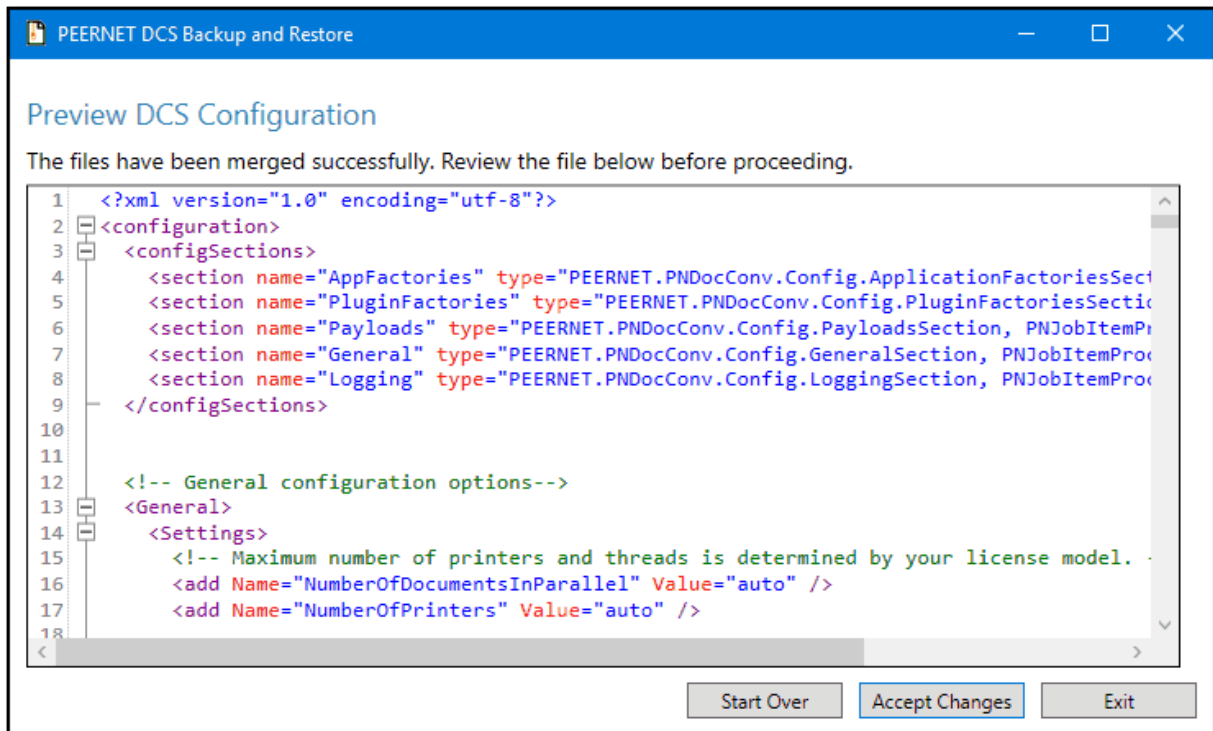
The first step of the restore is to merge any changed or added settings in the saved DCS configuration file with the new file just installed. This allows you to maintain any of your changes to this file while also gaining access to any new features that were added. This step is skipped if the DCS configuration file was not selected to be restored.

The results of the merge, whether successful or with errors, are listed on the screen.

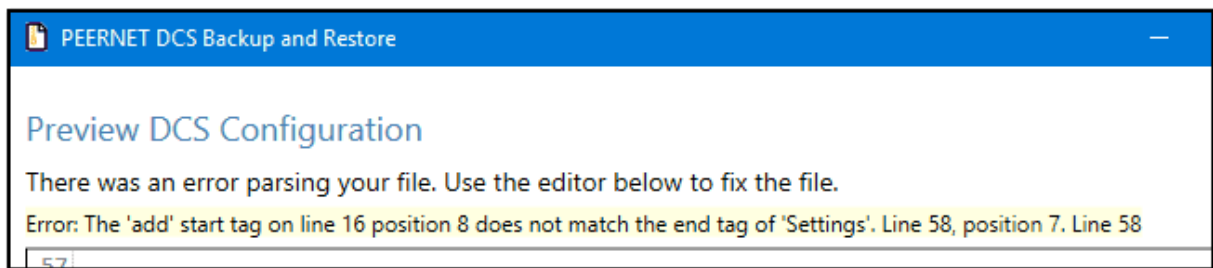
Click the **Next** button to review the successfully merged file, or to edit the file if any errors occurred. The merged file is not copied to the Document Conversion Service folder until later, in Step 3.



Preview and edit your merged DCS configuration file from this screen. Any changes made in the editor are validated before moving to the next step. Click **Accept Changes** when you are done.

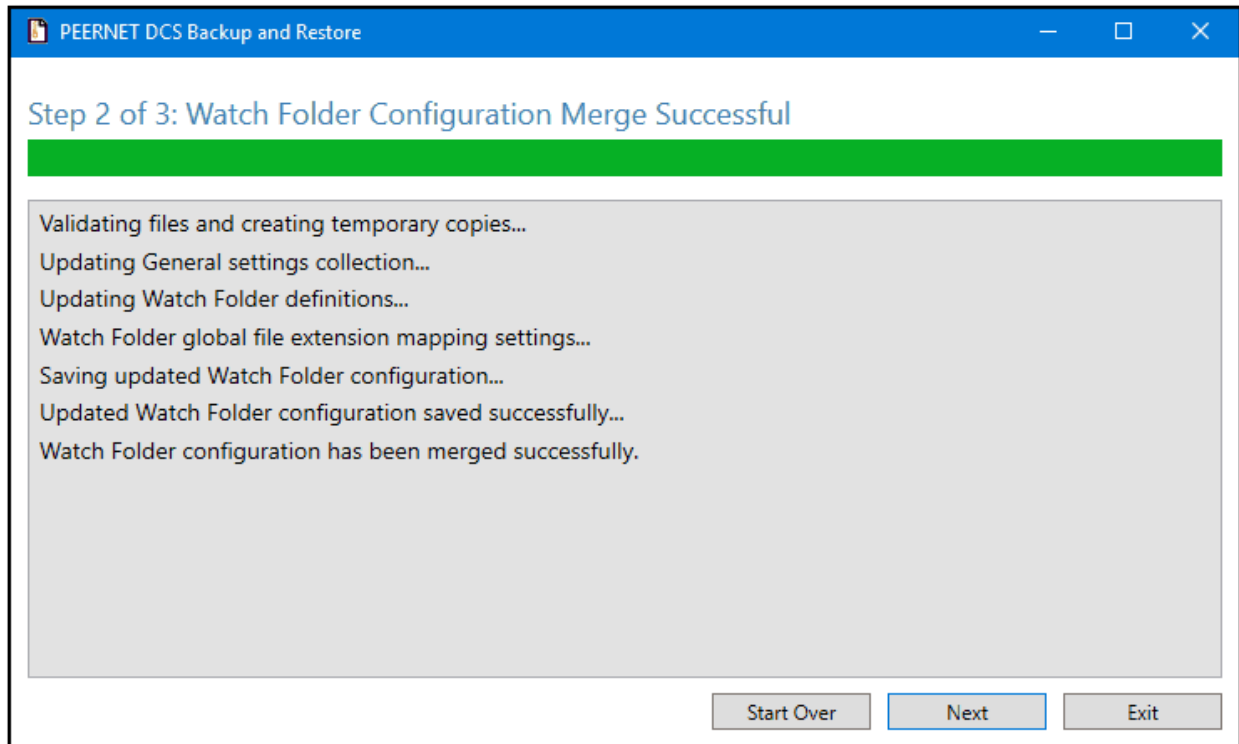


If any errors were found, information about the error is displayed at the top of the screen. This allows you to find and fix syntax and other errors easily.

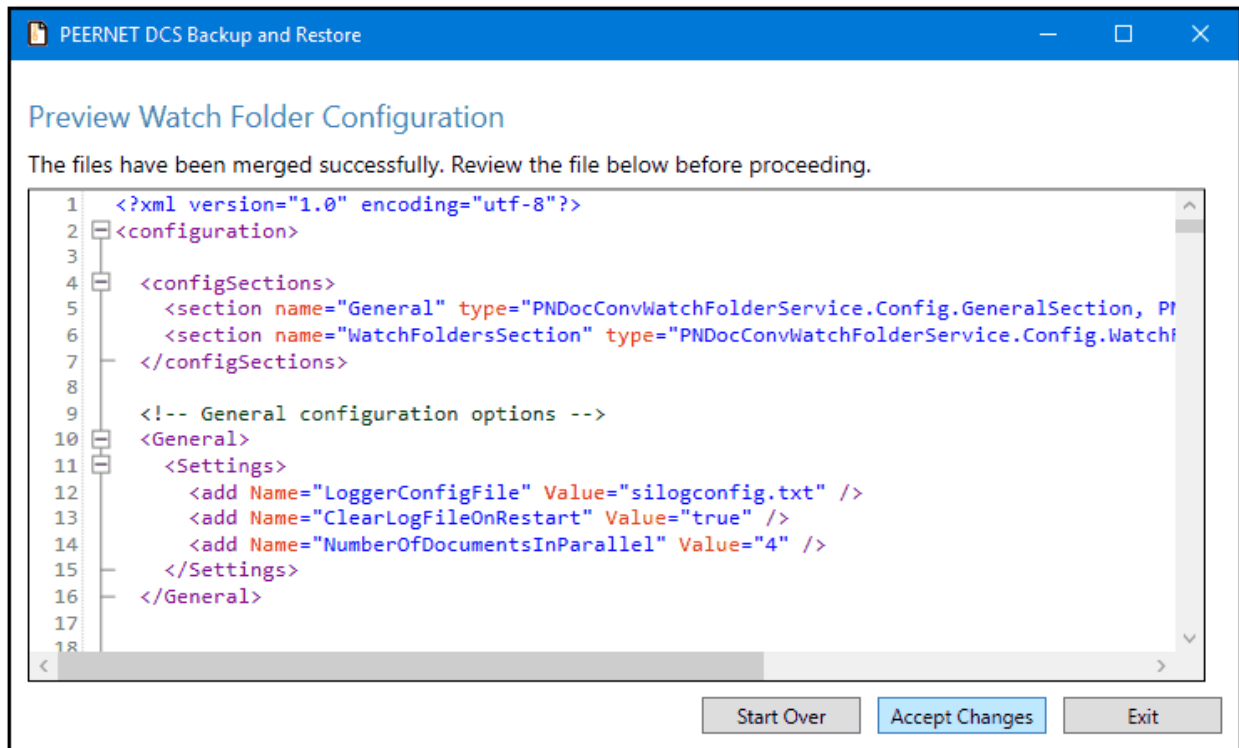


Next, the same steps are done using the Watch Folder configuration file, if this file was selected to be restored. As with the DCS configuration file, the results of the merge, whether successful or with errors, are listed on the screen.

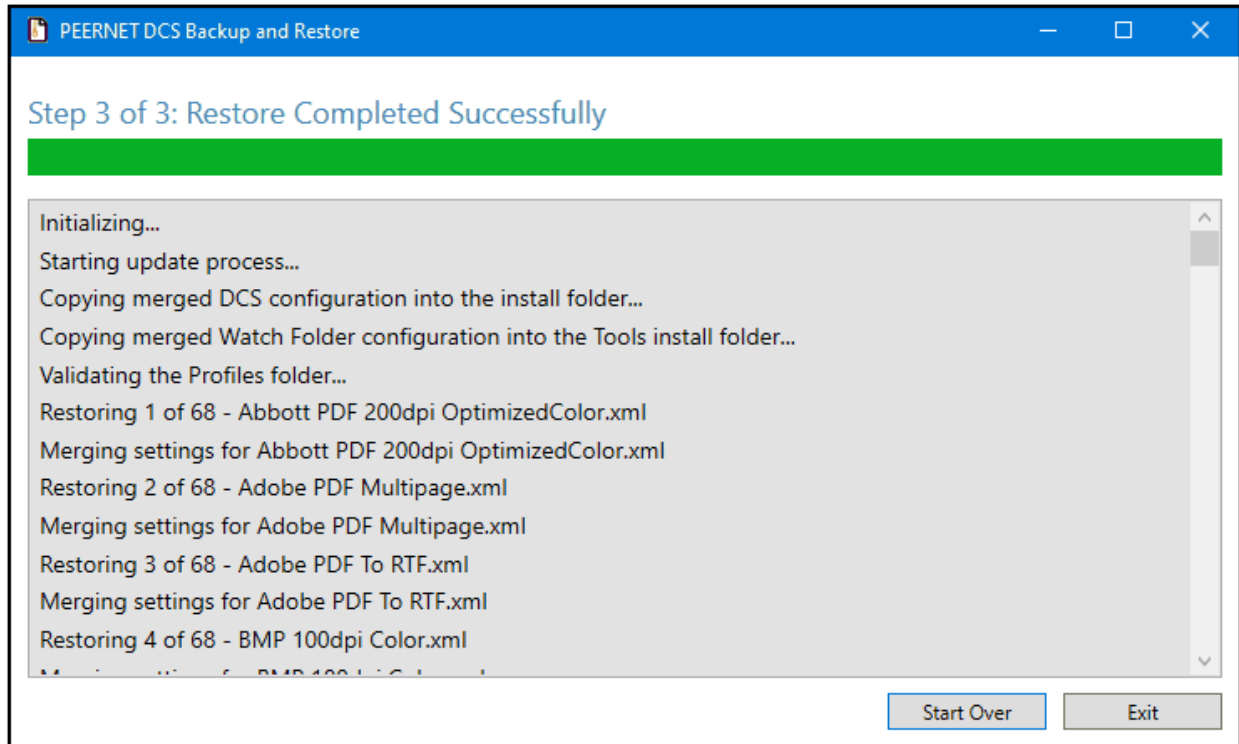
Click the **Next** button to review the successfully merged Watch Folder configuration file, or to edit it if any errors occurred. Again, the merged file is not copied to its final destination until Step 3 below.



Preview and edit your merged Watch Folder configuration file from this screen. Click **Accept Changes** when you are done.



The last step in the restore process is to copy the merged configuration files, as selected, and any selected profiles into their respective locations. Beginning in version 3.0.031, the saved profiles are also merged with the new versions in the install. Prior to this version the saved conversion profiles were copied and not merged.



Manually Backup and Restore the Files

The DCS configuration, Watch folder configuration and conversion profiles can also be backed up by manually copying the files to a saved location and then visually merging any changes after.

The DCS Configuration File

Backing up the File

1. You can also open the configuration file by going to **Start – All Programs – Document Conversion Service 3.0– Edit DCS Configuration File**. Older versions of DCS will open this in Notepad, while newer versions will open in the [DCS Editor](#).
2. Once open, save a copy of the file in a safe location.

Restoring the File

After you have updated or upgraded, always check the new configuration file for changes and new settings that may have been introduced with the new update or upgrade.

If there have been no new settings added to the configuration file, then you can simply copy your saved DCS configuration file to **C:\Program Files\PEERNET Document Conversion Service 3.0\Core**.

If there have been changes, and you wish to have access to the new settings, you can automatically merge the old and new DCS configuration file and see a preview by [Using the Configuration Merge Tool](#).

To manually merge the settings by visually comparing the files, do the following:

1. Go to **Start – All Programs – Document Conversion Service 3.0 – Edit DCS Configuration File** to open the new file. Older versions of DCS will open this in Notepad, while newer versions will open in the [DCS Editor](#).
2. Open the original file you saved. This can be opened in Notepad, or another text editor of your choice.
3. Compare the two files and copy your changes from the old DCS configuration file to the new file.
4. Save your file. If you are using the [DCS Editor](#), the file will be validated for syntactical errors before saving.

The Watch Folder Configuration File

Backing up the File

1. Open the Watch Folder Service configuration file by going to the **DCS Dashboard**, clicking the **Watch Folder Settings** tile, and then **Edit Configuration** tile. Or you can go to **Start – All Programs – Document Conversion Service 3.0 – Watch Folder – Configure Watch Folder Settings** to open the configuration file. Older versions of DCS will open this in Notepad, while newer versions will open in the [DCS Editor](#).
2. Once open save a copy of the file in a safe location.

Restoring the File

After you have updated or upgraded, always check the new Watch Folder configuration file for changes and new settings that may have been introduced with the new update or upgrade.

If there have been no new settings added to the configuration file, then you can just copy the Watch Folder configuration file that you had saved to **C:\Program Files\PEERNET Document Conversion Service 3.0\Tools\Watch Folder Service**.

If there have been changes, and you wish to have access to the new settings, you can automatically merge and preview the old and new Watch Folder configuration file and see a preview by [Using the Configuration Merge Tool](#).

To manually merge the settings by visually comparing the files, do the following:

1. In the **DCS Dashboard**, click the **Watch Folder Settings** tile, and then the **Edit Configuration** tile. You can also go to **Start – All Programs – Document Conversion Service 3.0 – Watch Folder – Configure Watch Folder Settings** to open the new configuration file. Older versions of DCS will open this in Notepad, while newer versions will open in the [DCS Editor](#).
2. Open your original, modified file you saved. This can be opened in Notepad, or another text editor of your choice.
3. Compare the two files and copy your changes from the old Watch Folder configuration file to the new file.
4. Save your file. If you are using the [DCS Editor](#), the file will be validated for syntactical errors before saving.

The Conversion Profiles

The conversion profiles are used by the sample programs, the command line utilities, and by the PEERNET.ConvertUtility.dll .NET assembly.

If you have modified any of the sample profiles that DCS installs by default, or if you have added your own, you should back up the contents of this folder before upgrading.

Backing up the Profiles

1. In the **DCS Dashboard**, click the **DCS Settings** tile, then the **Open Profiles Folder** tile to open the profiles folder. You can also open it by going to **Start – All Programs – Document Conversion Service 3.0 – Samples – Open Conversion Profiles Folder** in the Start menu.
2. This folder contains File Extension to Converter Map.xml file, and all other files are conversion profiles. Copy all of the files from here and save them in a safe location.

Restoring the Profiles

After you have updated or upgraded, always check the File Extension to Converter Map.xml and any edited profiles, or default profiles that you are using, against the new ones for any changes and new settings that may have been introduced with the new update or upgrade.

If there have been no new settings added to the file, then you can simply copy your saved files back to the profiles location at **Start – All Programs – Document Conversion Service 3.0 – Samples – Open Conversion Profiles Folder**.

If there have been changes, and you wish to have access to the new settings, you can automatically merge the old and new profile and see a preview by [Using the Configuration Merge Tool](#).

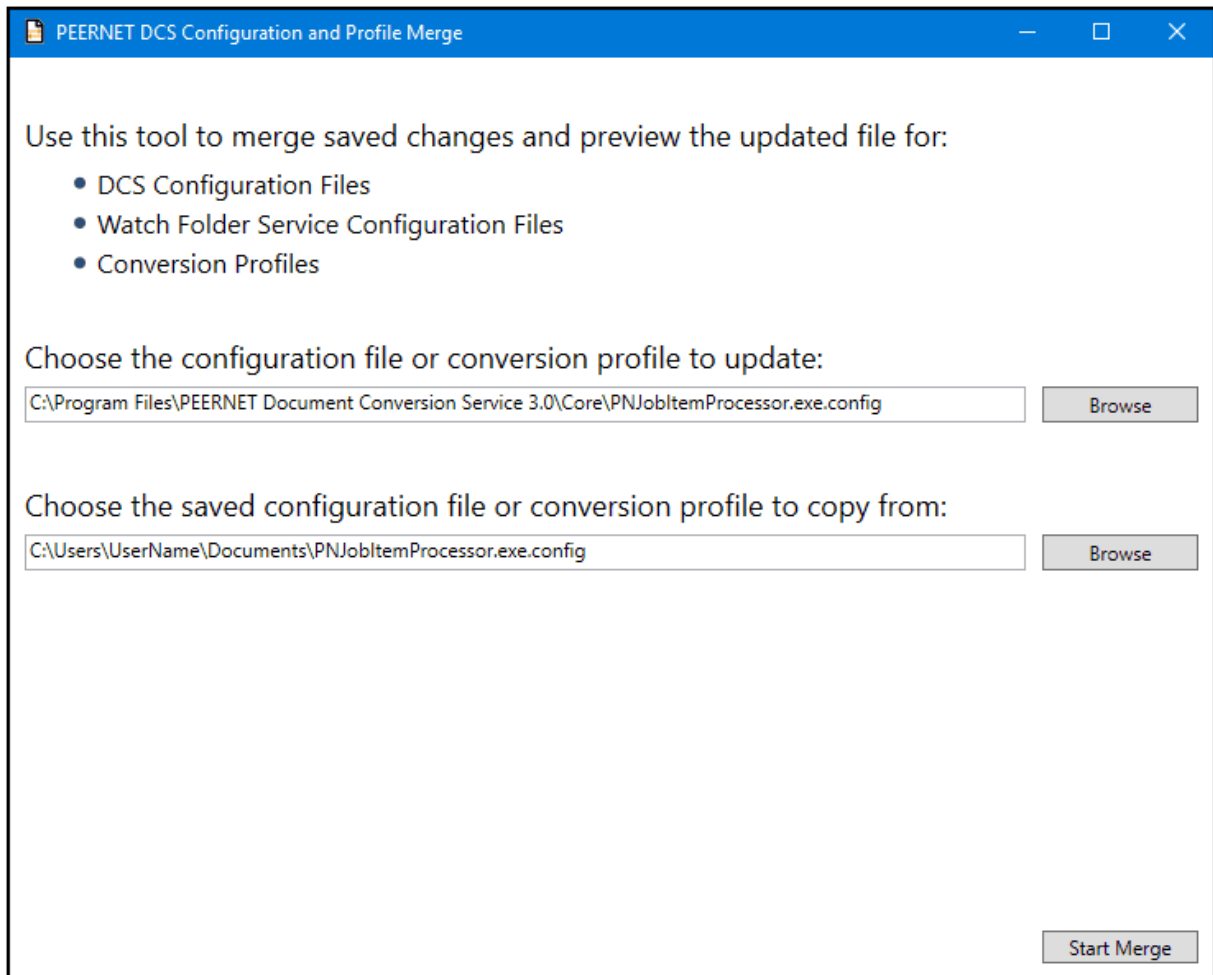
To manually merge the settings in individual profiles by visually comparing the files, do the following:

1. In the **DCS Dashboard**, click the **DCS Settings** tile, and then the **Edit and Create Profiles** tile. You can also go to **Start – All Programs – Document Conversion Service 3.0 – Edit Conversion Profiles** to open the new profile in the [DCS Editor](#).
2. From the editor's open flyout, click on **Choose Conversion Profile** to open the new profile you wish to compare. To open the File Extension to Converter Map.xml, choose the **File Extension Converter Map** button.
3. Open the original, modified file you saved. This can be opened in Notepad, or another text editor of your choice.
4. Compare the two files and copy your changes from the old profile to the new profile.
5. Save your file. If you are using the [DCS Editor](#), the file will be validated for syntactical errors before saving.

Using the Configuration Merge Tool

When manually restoring configuration files or conversion profiles, the DCS Configuration and Profile Merge tool can be used to merge your saved configuration files and conversion profiles with the new installed files.

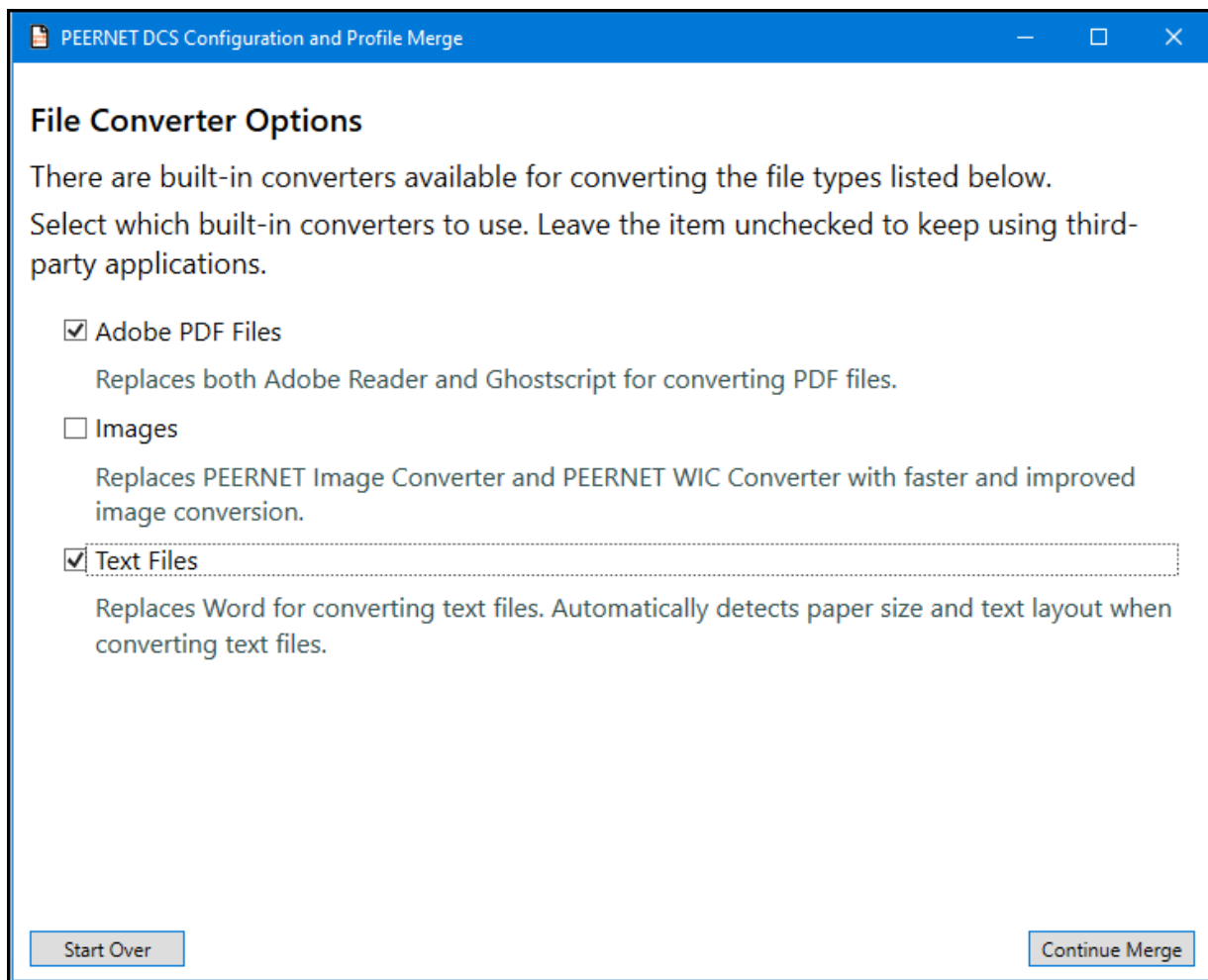
Select the new configuration or profile, and then select the matching older, saved file you want to copy any settings from. Select **Start Merge** to begin merging the files.



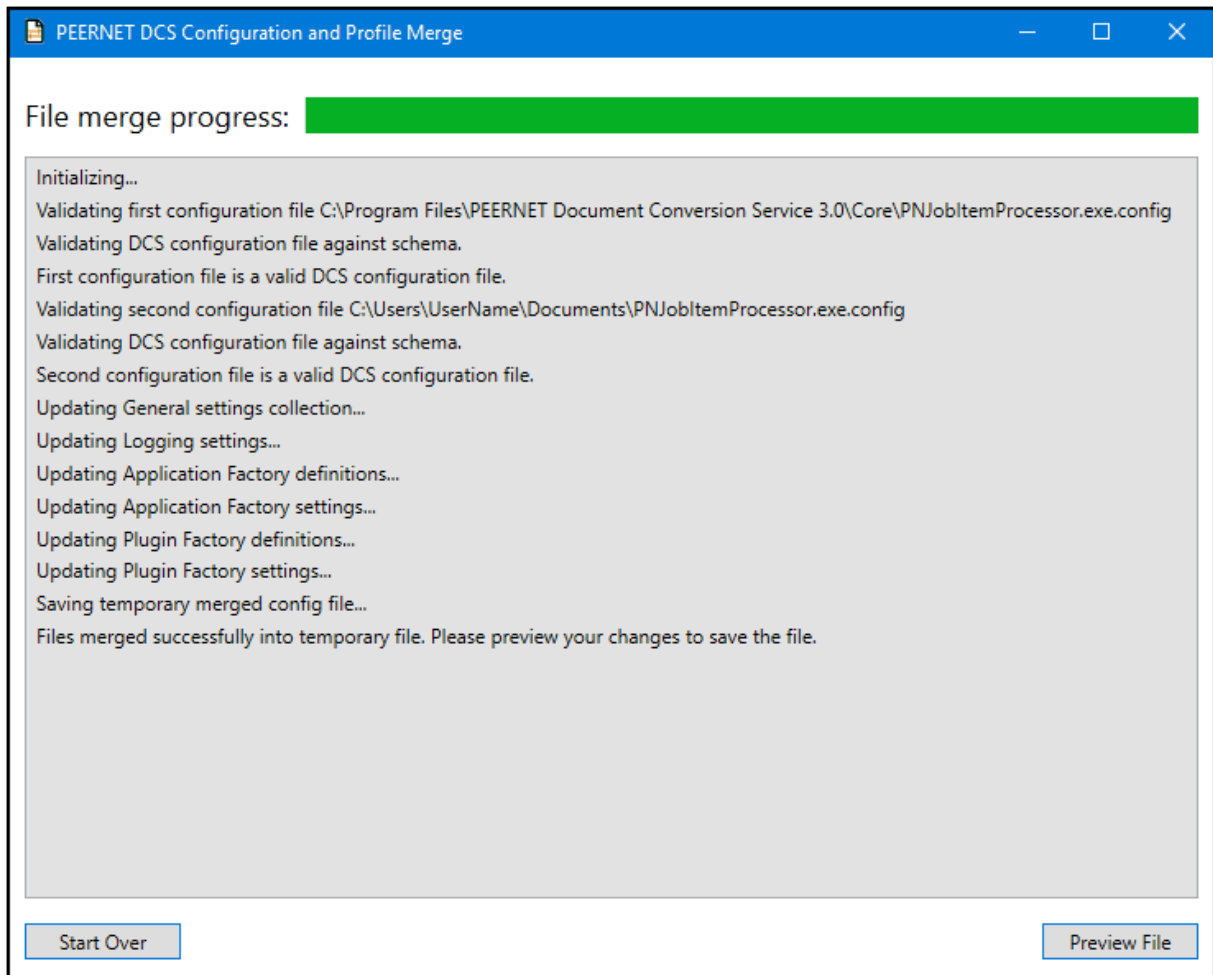
Starting with Document Conversion Service 3.0.031, when you are merging a DCS configuration file or a Watch Folder Service configuration file you will now see the following option screen for the new built-in converters for Adobe PDF files, text documents, and images files.

These converters do not require any third-party applications to be installed, and offer improvements for conversion speed and support for converting new file types. Running a merge in previous versions of Document Conversion Service will not show this option.

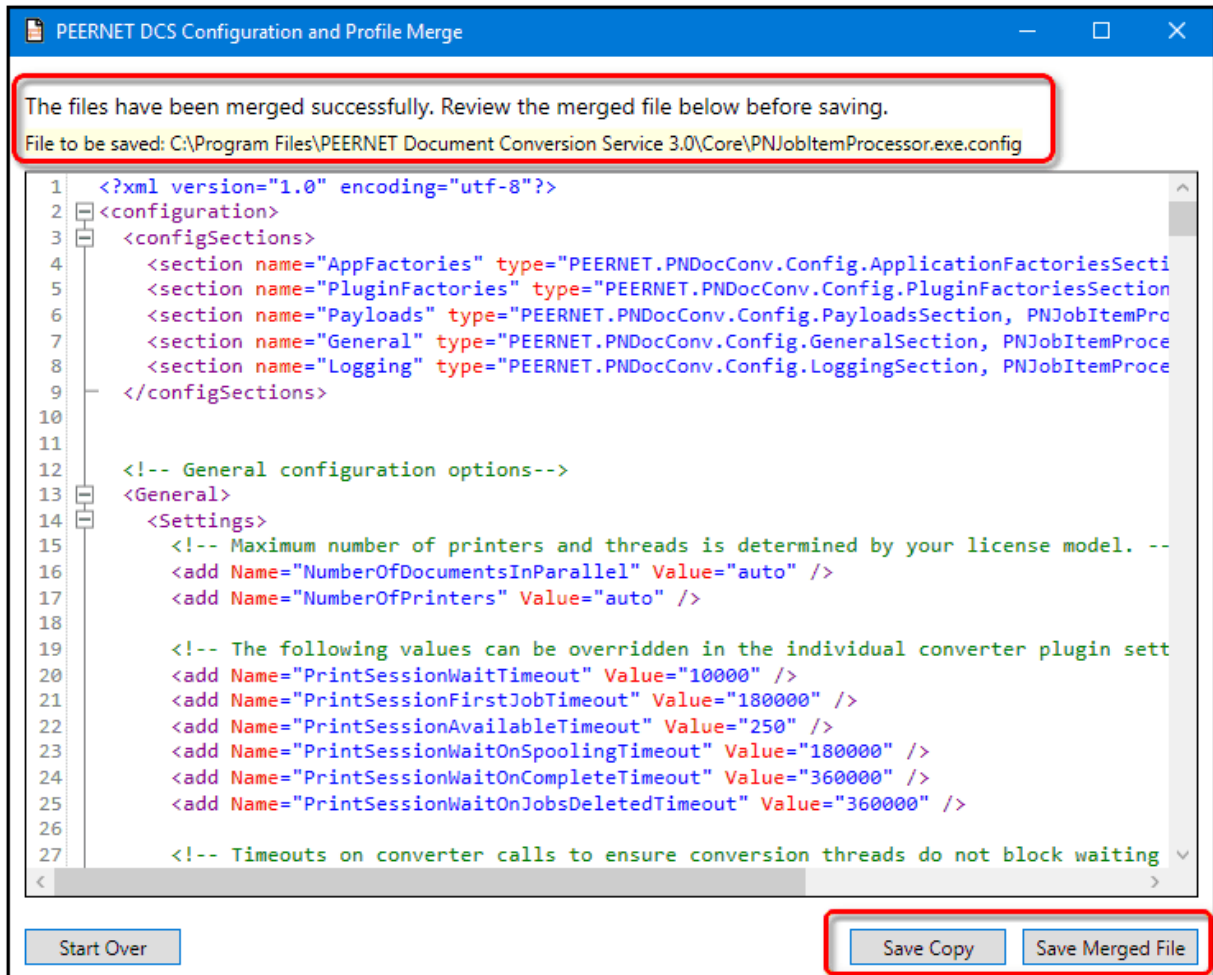
To use the new converters, check which ones you want to use on this screen. This will enable them in your updated configuration for DCS or Watch Folder.



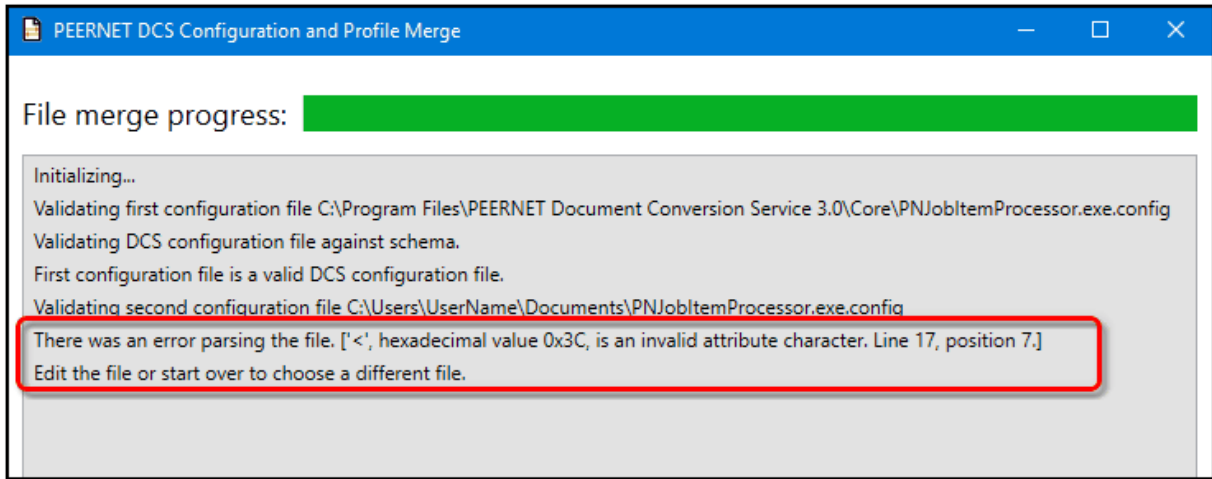
The progress of the merge is shown as the files are evaluated and combined. Click **Preview File** to see the results of a successful merge.



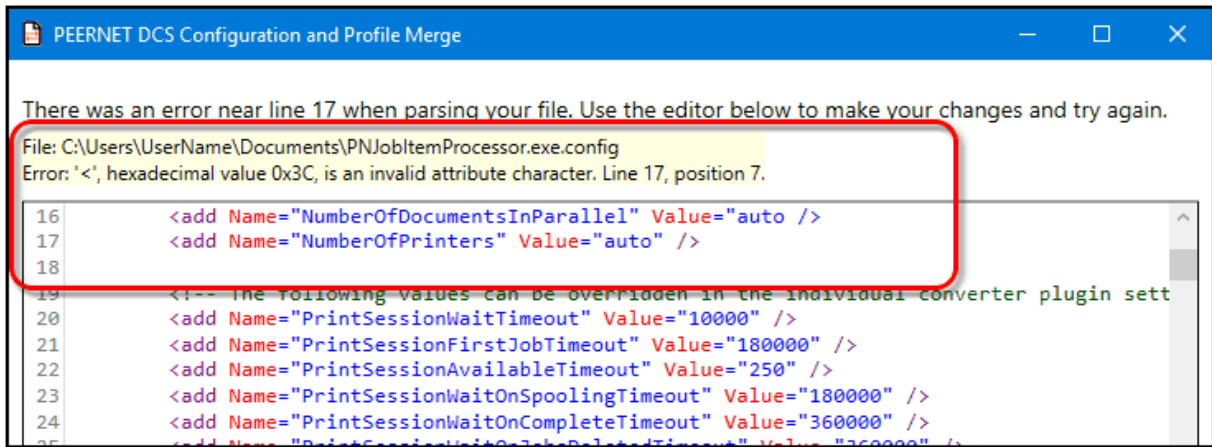
The preview allows you to verify that all settings have been copied over. The file that will be updated is shown at the top of the screen. Click **Save Merged File** to update the file, or **Save Copy** to create a copy of the merged file.



If any errors are found during the merge, they are listed first on the merge progress screen. To edit the merged file to fix the errors, click **Edit File**.



When editing a merged file with errors, the error information is shown at the top of the screen, and the file is scrolled to the closest location of the error. The sample shown below is missing a quotation mark on line 16. This causes a syntax error when starting the next line. Fix the errors and click **Save and Merge Again** to repeat the process until there are no syntax errors on the merged file and it merges successfully.



Installing Document Conversion Service Silently

Document Conversion Service and the client PNDocConvClientSetup_3.0.exe can be installed silently allowing the main service application to be installed on servers using push software and to allow the client install to be bundled with custom software.

Installing Document Conversion Service Silently

Document Conversion Service can be installed silently using the following command line arguments. When the install is not run silently, the command line arguments are ignored except for the SHOWDASHBOARD argument.

The /S argument and the PASSWORD= argument are required, all other arguments are optional.



Note

Silent installation was introduced in Document Conversion Service 2.0.018 in February 2015. Earlier versions of the 2.0 build, and previous install versions did not have the silent install options.

```
pndscsetup_3.0.###.exe /S
                        /L=<logfile.ext>
                        PASSWORD="password"
                        [DCSUSER="domain\user" ]
                        [LAUNCHDCS=TRUE|FALSE]
                        [RUNWATCHSERVICE=TRUE|FALSE]
                        [OPENDASHBOARD=TRUE|FALSE]
```

Sample Command Lines

<pre>pndscsetup_3.0.###.exe /S PASSWORD="password"</pre>
<p>Runs the setup silently with no user interface. If one does not exist, a local administrative account will be created for the user 'DCSAdmin' and using the supplied password.</p> <p>If it already exists, the account will be validated and used with the supplied password. If the password is invalid, the install will fail.</p>
<pre>pndscsetup_3.0.###.exe /S /L="C:\PEERNET\dcsllog.txt" DCSUSER=".MyDCSAdminUser" PASSWORD="password"</pre>
<p>Runs the setup silently with no user interface. The account must already exist. It is validated using the supplied password. If the password is invalid, the install will fail. A log of the install, <i>dcsllog.txt</i>, is created in C:\PEERNET. The directory must already exist.</p>
<pre>pndscsetup_3.0.###.exe /S DCSUSER="DOMAIN\MyUser" PASSWORD="password" LAUNCHDCS=TRUE</pre>
<p>Runs the setup silently with no user interface. The domain account MyUser will be validated using the supplied password. If the password is invalid, the install will fail. The install will launch Document Conversion Service at the end of the installation step.</p>

/S - Silent Install

This will run the installation silently with no user interface (no setup wizard). Installing silently requires that the *PASSWORD=* variable be provided. When used without the *DCSUSER=* variable, the password is used to create or validate an existing *DCSAdmin* account. If a *DCSUSER* variable is provided, the password is used to validate that account. If the accounts cannot be validated, or the *PASSWORD* information is not provided the setup will terminate.

/L - Create a Logging File

Pass in a fully qualified path to a filename to create a logging file.

DCSUSER="domain\user"

This argument is optional. If not provided we default to our local account *DCSAdmin*

The services and configuration for Document Conversion Service require a user account, local or domain-level, that has administrative privileges. We normally recommend that you let us create and use our local account *DCSAdmin*.

If you cannot use this account you can specify a different user through this argument. If using a domain account, you need to specify the domain and user name. The install process also needs to be able to validate the account. The setup will fail if the account cannot be validated. If you are using a different local account, specify the local account using the dot syntax for local, ".*MyLocalUser*".

PASSWORD="password"

The install requires a user account with administrative privileges to initialize the services and configure for client-server conversion. A password must be supplied to create the *DCSAdmin* account, or validate the account if an existing one is used. If the account cannot be validated, or the password variable is not supplied, the setup will terminate.

LAUNCHDCS=TRUE|FALSE

This argument is optional and defaults to *FALSE*. If passed as *TRUE* then the setup will automatically start Document Conversion Service when the install is complete.

RUNWATCHSERVICE=TRUE|FALSE

This argument is optional and defaults to *FALSE*. If passed as *TRUE* then the setup will automatically start the Watch Folder Service when the install is complete.

OPENDASHBOARD=TRUE|FALSE

This argument is optional and defaults to *TRUE* unless this is a silent install, where it defaults to *FALSE*. If passed as *TRUE* then the setup will automatically launch the Dashboard when the install is complete.

Installing PNDocConvClientSetup_3.0.exe Silently

This client software can be installed as a separate step from your application, called from your installation, or you can bundle it with your own install by using command line arguments to run the install silently.

There are two types of setup that can be controlled from the command line - BASIC, and FULL. The BASIC setup is the same as the *Minimum* install and only installs the required components for remote conversion in a client-server environment. The FULL setup is the same as a *Complete* install and includes the Watch Folder Service and sample code, the command line conversion tools and all additional sample code.

When the client install is not run silently, the command line arguments are ignored.

```
PNDocConvClientSetup_3.0.exe /S
PASSWORD="password"
[SETUPTYPE=BASIC|FULL]
[DCSUSER="domain\user"]
```

Sample Command Lines

<code>PNDocConvClientSetup_3.0.exe /s PASSWORD="password"</code>
<p>Runs the basic client setup silently with no UI. If one does not exist, a local administrative account will be created for the user 'DCSAdmin' and using the supplied password.</p> <p>If it already exists, the account will be validated with the supplied password. If the password is invalid, the install will fail.</p>
<code>PNDocConvClientSetup_3.0.exe /s SETUPTYPE=BASIC DCSUSER=".MyLocalUser" PASSWORD="password"</code>
<p>Runs the basic client setup silently with no UI.</p> <p>The local account MyLocalUser will be validated with the supplied password. If the password is invalid, or the account not exist, the install will fail.</p>
<code>PNDocConvClientSetup_3.0.exe /s SETUPTYPE=FULL DCSUSER="DOMAIN\MyUser" PASSWORD="password"</code>
<p>Runs the full client setup silently with no UI.</p> <p>The domain account MyUser will be validated with the supplied password. If the password is invalid, or the account not exist, the install will fail.</p>

/S - Silent Install

This will run the installation silently with no wizard. If no `SETUPTYPE` is specified, then a `BASIC` install is done.

The client install also requires that the `PASSWORD=` variable be provided. When used without the `DCSUSER=` variable, the password is used to create or validate an existing `DCSAdmin` account. If not provided the setup will terminate.

`PASSWORD="password"`

The client install requires a user account with administrative privileges to initialize the services and configure for client-server conversion. A password must be supplied to create the account, or validate the account if an existing one is used. If the account cannot be validated the setup will terminate.

`SETUPTYPE=BASIC|FULL`

Choose the setup type - *BASIC* or *FULL*. The *BASIC* setup only installs the required components for remote conversion in a client-server environment. The *FULL* setup will also install the Watch Folder Service and sample code, the command line conversion tools and all additional sample code.

When this argument is not specified, a *BASIC* setup is installed.

`DCSUSER="domain\user"`

The services and configuration for client-server conversion require a user account, local or domain-level, that has administrative privileges. We normally recommend that you let us create and use our local account `DCSAdmin`.

If you cannot use this account you can specify here a different user. If using a domain account, you need to specify the domain and user name. The install process also needs to be able to validate the account. The setup will fail if the account cannot be validated. If you are using a different local account, specify the local account using the dot syntax for local, `".MyLocalUser"`.

Activating Document Conversion Service

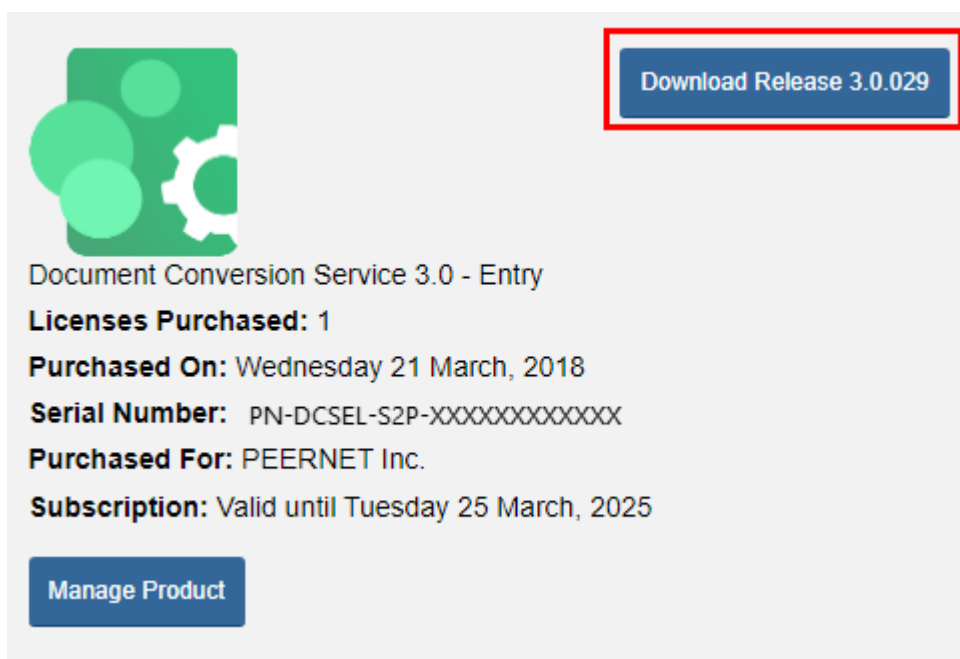
The first step to activating Document Conversion Service is installing the latest version of the software which is available through your online account.

If you installed the trial version of Document Conversion Service before purchasing, you still need to install your purchased copy of the software.

Any modifications made to configuration files for Document Conversion Service, Watch Folder Service and any edited conversion profiles are backed up and restored as part of the installation process. See how to create your backup and restore your settings [During the Installation Process](#).

Installing Document Conversion Service:

Log into your [PEERNET online account](#) and find your recently purchased software in the **My Products** list. Select the **Download Release** button to download the latest release of the software.



In most cases, the downloaded setup, pndscsetup_3.0.###.exe will be saved to your **Downloads** folder. Open File Explorer and click the Downloads quick access link or browse to C:\Users\YOUR USERNAME\Downloads\.

Double-click the EXE file to run the Document Conversion Service install.

If you installed the trial version of Document Conversion Service before purchasing, when you run the install for the purchased copy, the previously installed trial version will prompt you to uninstall it. Select **Next** to remove the trial version and replace it with your purchased copy. Remember to [backup and restore your settings during the installation process](#).

At the end of the installation process, you are given the option to launch the Document Conversion Service **Dashboard**. Leave "Open the DCS Dashboard now" checked in order to open the DCS Dashboard directly. This is the fastest way to proceed to activating your software.

Launching the License Wizard

The **License Wizard** is accessible from the Document Conversion Service **Dashboard**.

To launch the **License Wizard**, begin by opening the **Dashboard**. Double-click the DCS Dashboard icon on your desktop or select Document Conversion Service 3.0 - DCS Dashboard from the **Start** menu to open the dashboard.



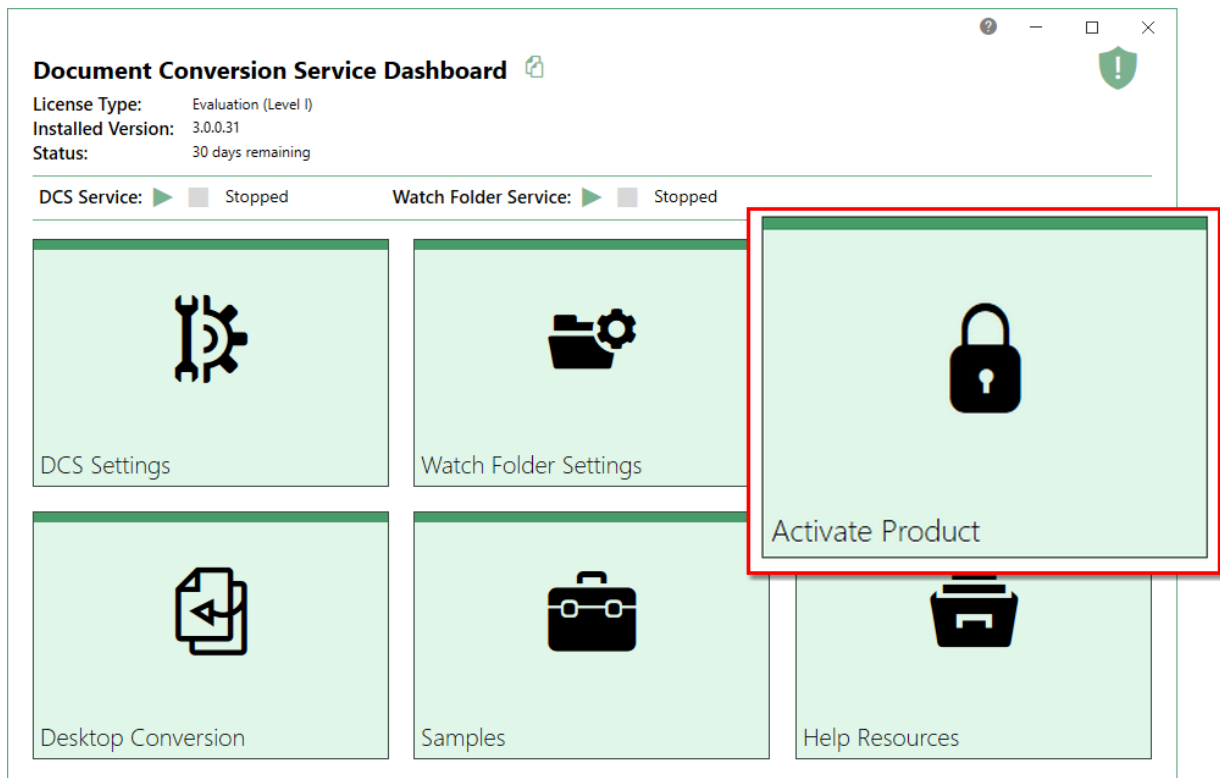
The tile to open the license wizard can have different titles based on your activation state.

Document Conversion Service starts as a 30-day watermarked trial unless previously activated on this computer. When Document Conversion Service is in evaluation mode, not activated, is deactivated, or your subscription has expired or is in an error state, the tile will say **Activate Product**.

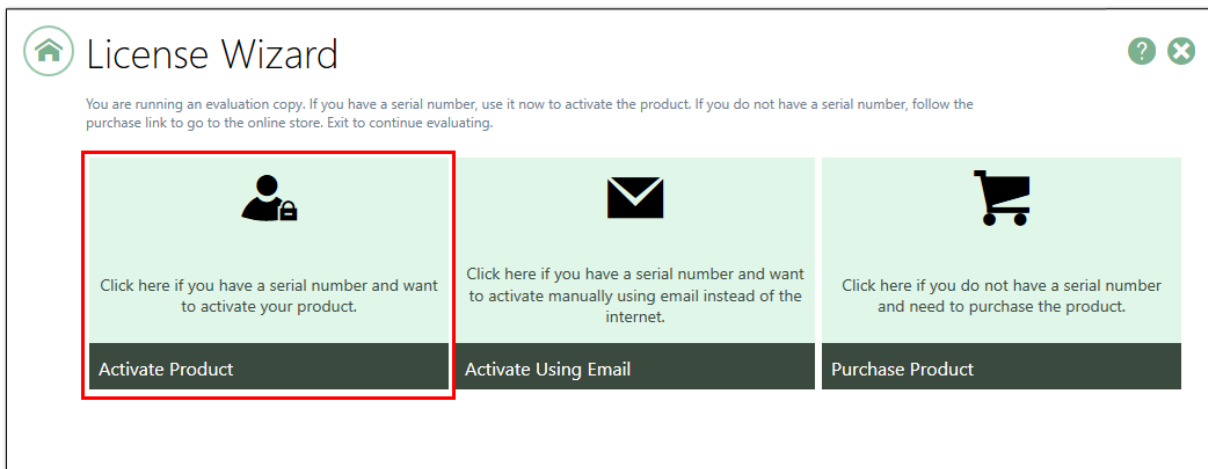
If you are in the middle of the manual activation process, the tile will say **Finish Manual Activation** instead.

If the software is already licensed, the tile is titled **License Wizard**.

Click the tile to open the License Wizard.



If you have time remaining in your trial, you will see the following message and options. If your trial has expired or you are trying to re-activate an expired subscription, you will see the same options but no message.



If you have not purchased the product, click **Purchase Product** to go to our online store to complete your purchase. Once purchased, an order confirmation notification containing your serial number will be sent to you by email.

If you have a serial number, click **Activate Product** to activate using the internet or **Activate Using Email** to activate when you do not have access to the internet. This will take you to the next step, [Entering Your Serial Number](#).

Entering Your Serial Number

To activate your product you need to enter in your serial number and your user information. Your serial number is included with your order confirmation email and can also be found through your [PEERNET online account](#).

Entering your serial number

Enter the serial number into the box on the screen. If you copy your entire serial number from your email and then return to this dialog it will automatically be filled into the box.

The serial number is case sensitive and it is important to type the serial number exactly as it is received. Be sure not to leave any spaces before or after the serial number when typing or pasting, and note that the serial number ends with a series of hexadecimal characters (0-9,A-F).

Entering your user information

When possible, your Name and Company information is automatically picked up from your system settings. The information in these fields can be change if required.

Name, company and email addresses entered here are only used internally by PEERNET to identify users for license adjustments. We will never rent or sell our customer's and client's information to third parties.

License Wizard

Serial Number
PN-DCSEL-3SP-XXXXXXXXXXXX

Your serial number can be found in your order confirmation email or in your online account.

Name
PEERNET

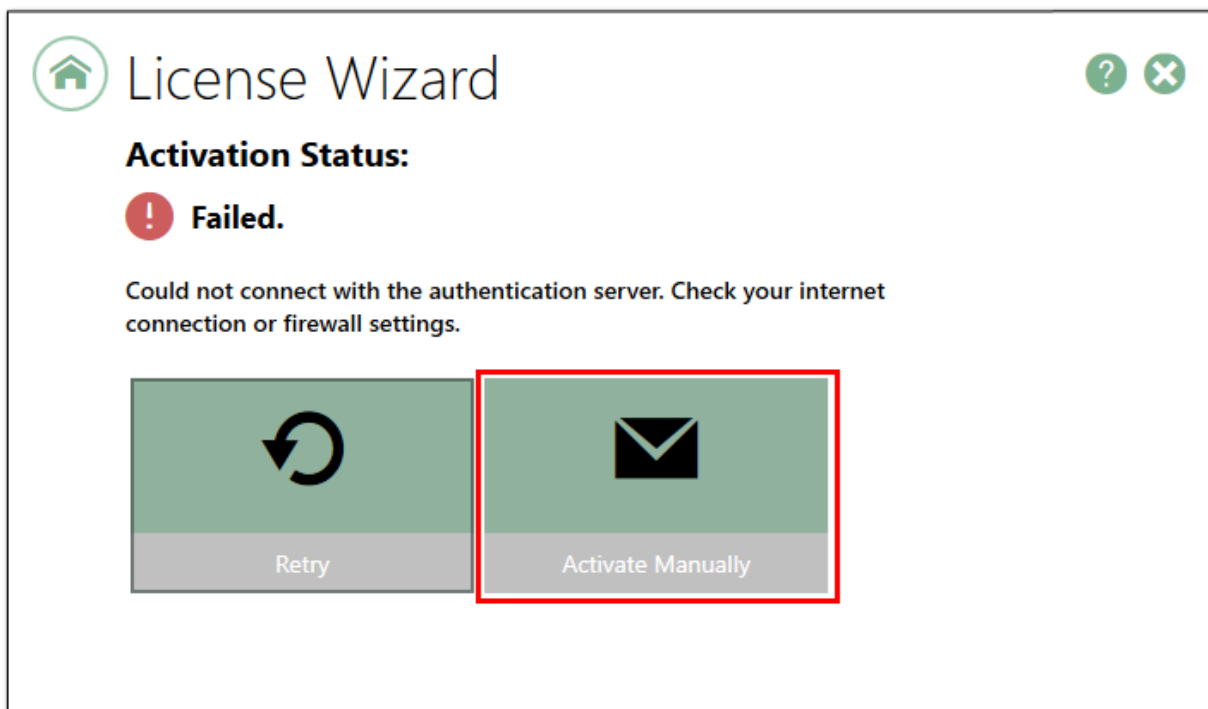
Company
PEERNET

Email
peernet@peernet.com

☐ Activate manually by email instead of using the internet

Click Next.

When activating over the internet, the License Wizard will attempt to validate an internet connection, and will prompt with the choice to license manually if it cannot connect. Activating over the internet may also fail if you have a firewall or anti-virus software blocking the connection. Click **Activate Manually** to begin the manual activation process, or click **Retry** to try activating online again.



Activating without an internet connection

If you are having difficulty connecting to the internet, or do not want to activate over the internet, you can choose to manually activate the product by checking "**Activate manually by email instead of using the internet**" option at the bottom of the screen.

☒ **Activate manually by email instead of using the internet**

Manual activation does not require an internet connection on the computer the software is installed on, but it does require that you have the ability to email an encrypted file to PEERNET for authentication. We will return the authenticated file to you, which you then import using the License Wizard to complete the activation process. These files are processed by PEERNET's technical staff from 09h00 to 17h00, Monday to Friday, Eastern Standard Time. See the section [Manually Activating Document Conversion Service](#).

Manually Activating Document Conversion Service

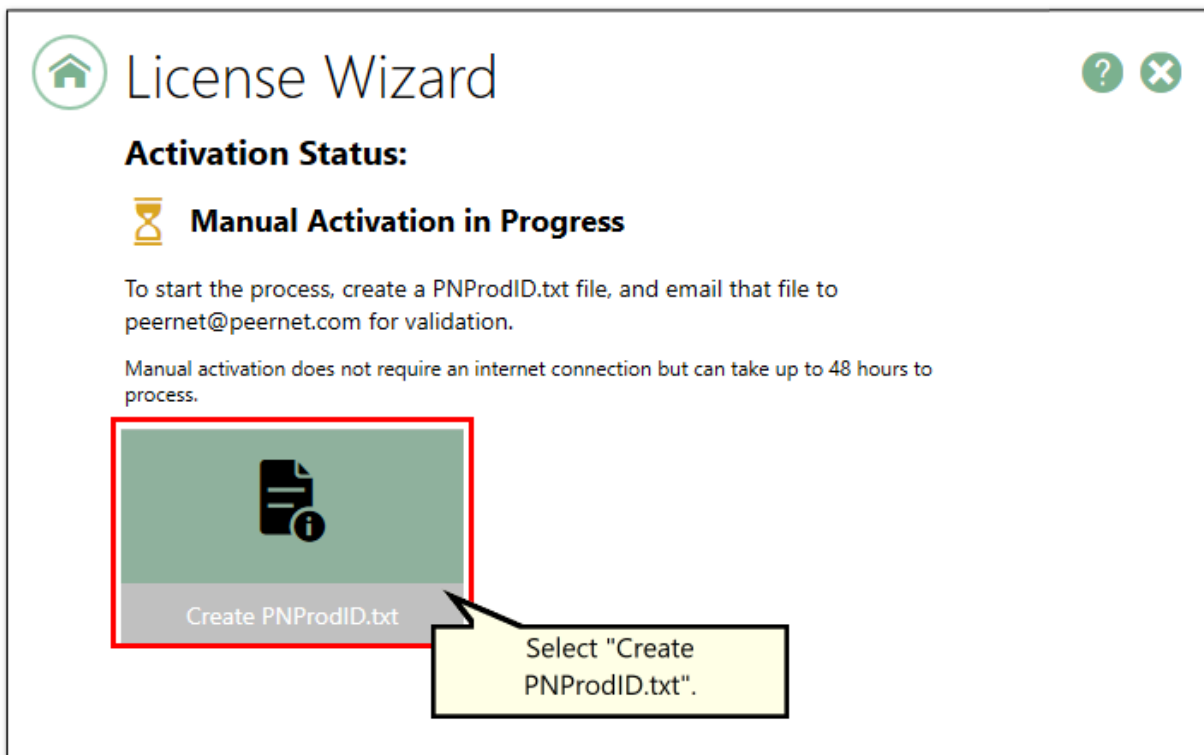
In most cases, you will not have to activate your product manually. This only happens when Document Conversion Service is installed on a computer that has no access to the internet, or the computer is configured such that the user cannot access the internet. Company policies, firewall programs or anti-virus software are all common reason our attempt to connect with our license server can be blocked.

If you do have to activate manually, you will need to follow the steps below. Please note that these files (PNProdID files) are only authorized during business hours, which are 09h00 to 17h00, Monday through Friday, Eastern Standard Time (excluding statutory holidays).

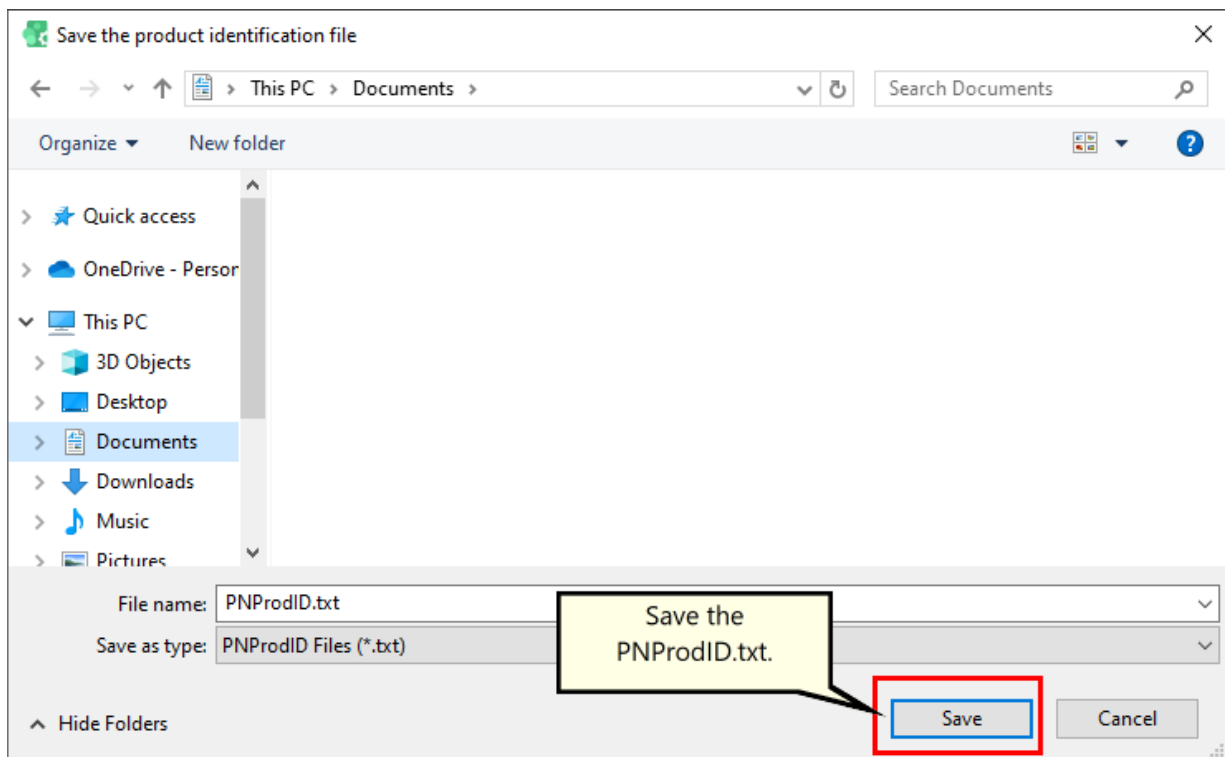
1. Use the **License Wizard** to create the encrypted file, **PNProdID.txt**.
2. Email the file to peernet@peernet.com for manual activation. For computers with no email capability, you can save the file to a shared network drive, or use an external storage device such as a USB flash drive (also known as thumb drives), or a MicroSD storage card to copy the file to a computer with email capabilities.
3. A file named **PNProdAU.txt** will be emailed back to you. Copy this file back to the computer where Document Conversion Service is installed and restart the **License Wizard** to complete the license activation.

Exporting the PNProdID.txt file

To create the file click **Create PNProdID.txt**.



A save dialog box will appear prompting you to choose where to save the **PNProdID.txt** product identification file. This dialog may look slightly different depending on which version of Windows you are running. Save this file in an easy to remember location, like your Desktop or your Documents folder.

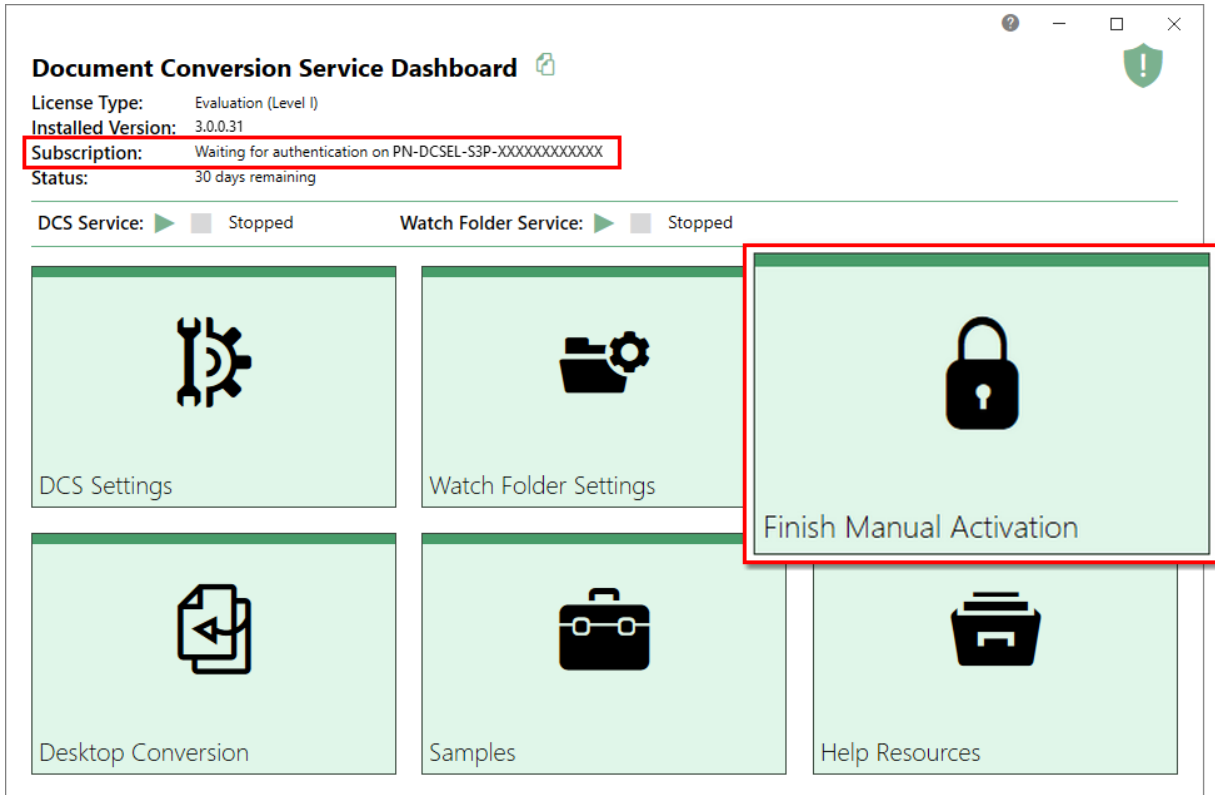


You need to email this file to peernet@peernet.com. For computers with no email capability, you can save the file to a shared network drive, or use an external storage device such as a USB flash drive or a MicroSD storage card to copy the file to another computer.

Importing the PNProdAU.txt file

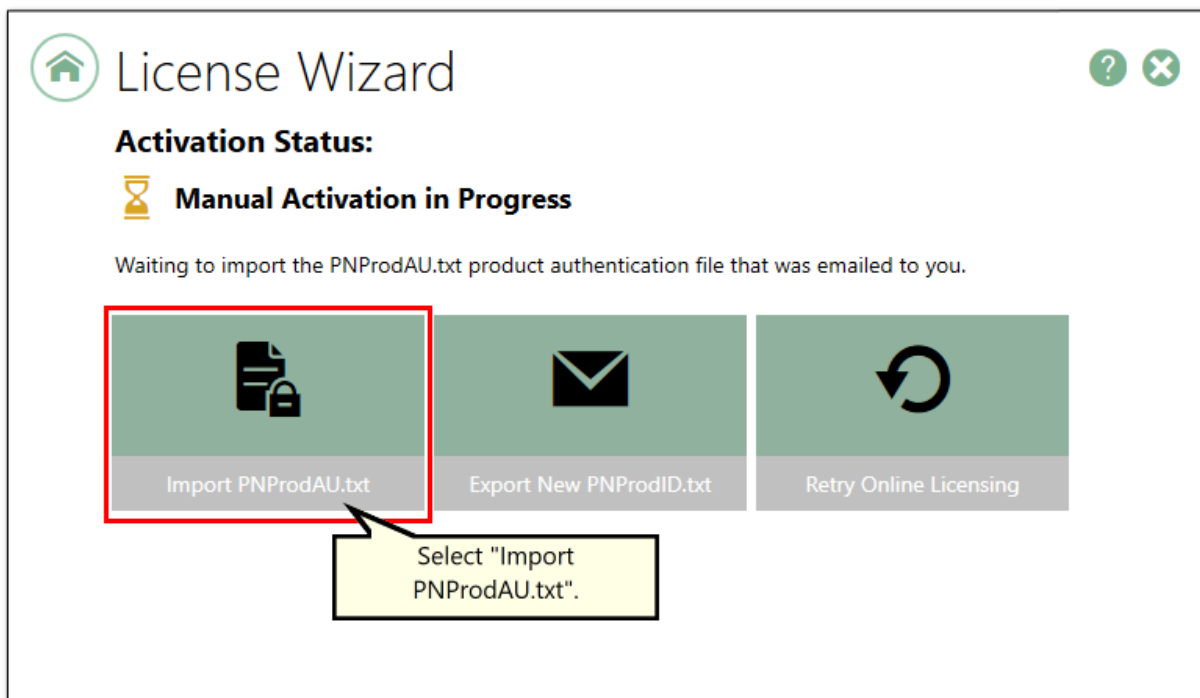
When you have received the product authentication file **PNProdAU.txt** from PEERNET, you will need save the file in an easy to remember location, like your Desktop or your Documents folder. If you need to move the authentication file back to the computer where Document Conversion Service is installed, do so now.

On the computer where Document Conversion Service is installed, open the Dashboard following the steps outlined in [Launching the License Wizard](#). The dashboard will show that you are waiting to finish a manual activation. Click the "Finish Manual Activation" tile to return to the license wizard.

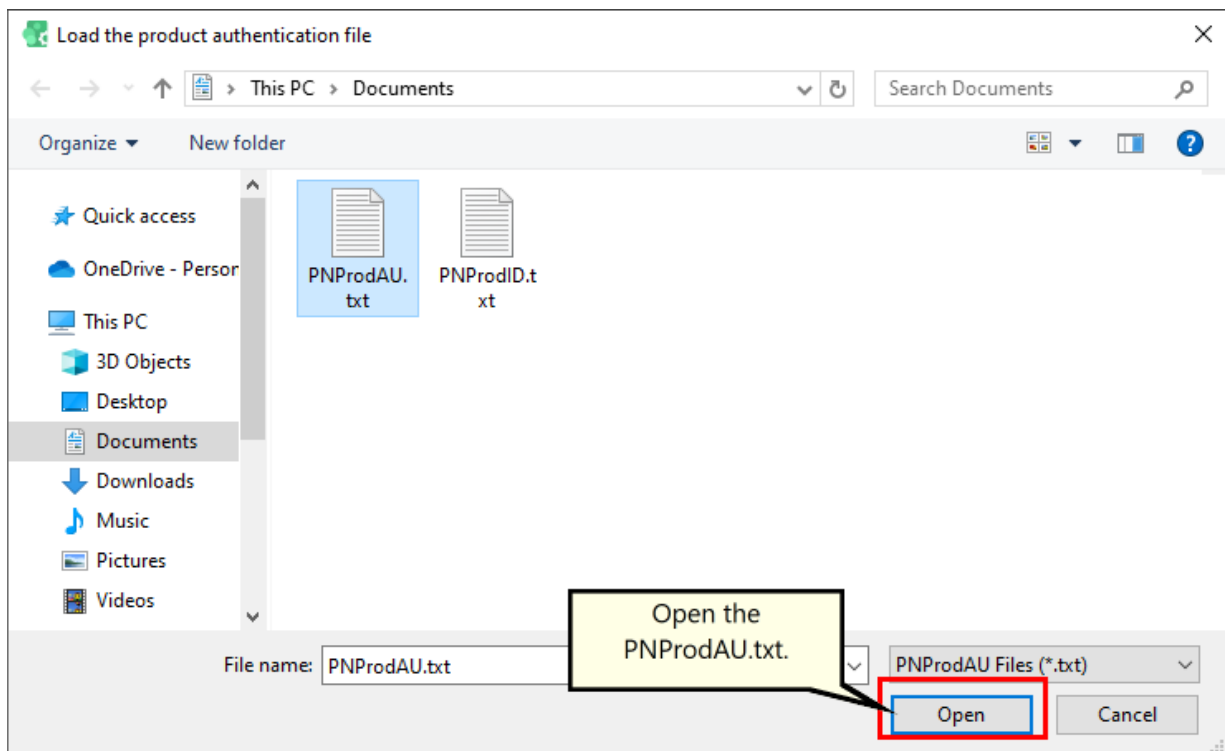


The license wizard will automatically start at the import screen.

Click **Import PNProdAU.txt**. If you did not receive a PNProdAu.text file, you can restart the manual licensing process. You also have the option to retry online activation if you now have an internet connection or have resolved an issue with your firewall settings.



A browse dialog box will appear. This dialog may look slightly different depending on which version of Windows you are running. Locate where you saved the **PNPProdAU.txt** file you received from PEERNET and click the Open button to import the file.

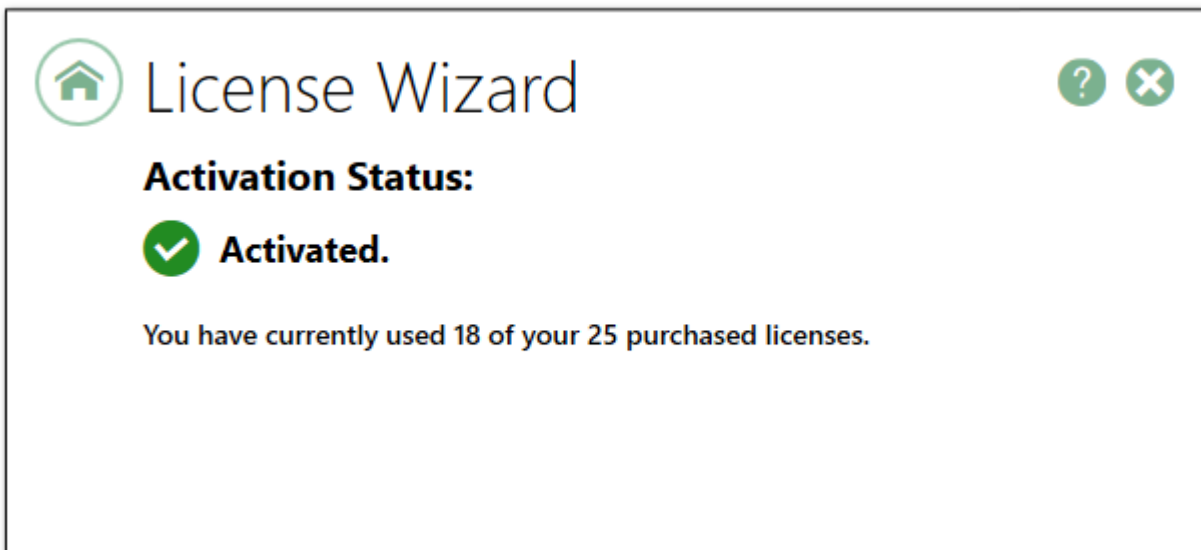


The authentication file is verified and you are automatically moved to the [Activation Status Results](#) screen.

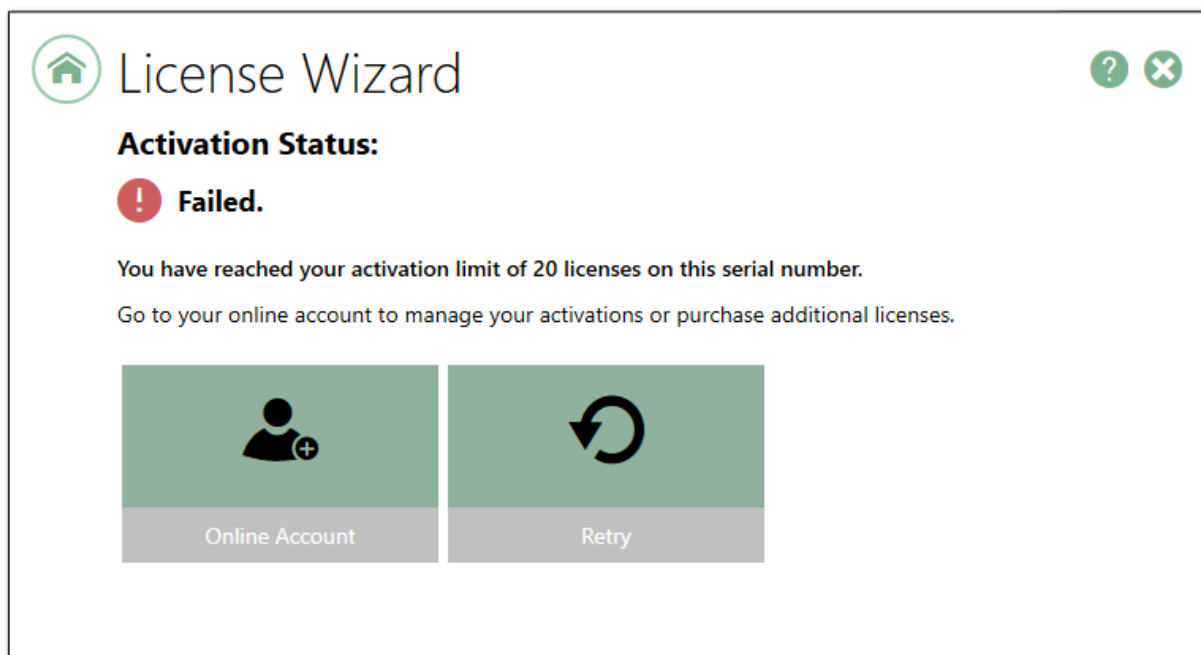
Activation Status Results

This screen displays your activation status.

If the product is successfully activated, the Activation Status will display your status as **Activated**.



If an error occurred during activation, the Activation Status will display your status as **Failed** with an explanation of the error that occurred. The most common error message that occurs is that you have consumed all of your licenses.



When you have used all your licenses, you will not be able to use the product on this computer until:

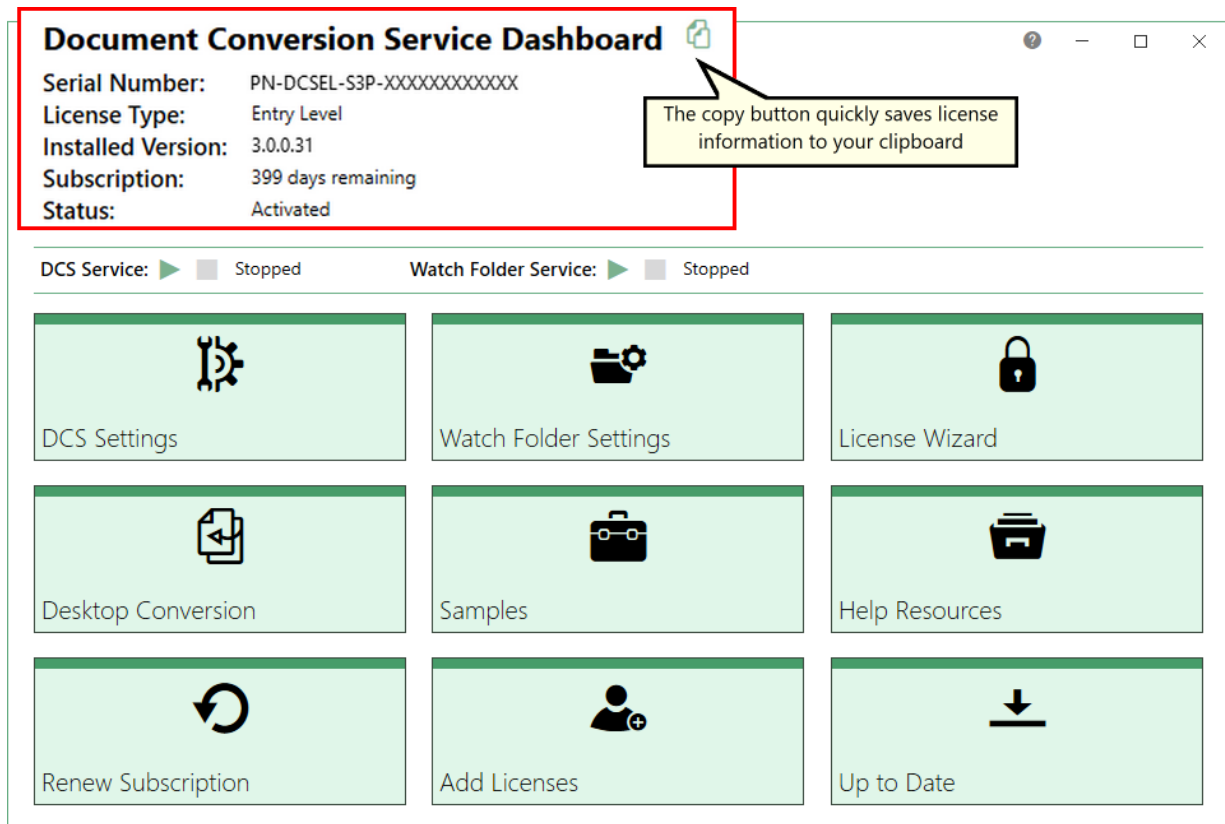
1. you purchase additional licenses, or
2. you adjust your serial number to deactivate a license on an old computer that is no longer in use.

In either case, abort the activation process by clicking the **Close icon** in the upper right-hand corner. Once you have either purchased additional licenses or deactivated a license that is no longer in use, re-start the activation process by [Launching the License Wizard](#).

For instructions on how to manage your licenses through your [PEERNET online account](#), please see the topic [View Activation Details](#).

Viewing Your Activation Status

Your activation status is always readily available in the top left-hand corner of the Document Conversion Service **Dashboard**.



The **Dashboard** always shows your current activation and licensing information. The copy button lets you quickly save the information to the clipboard.

- **Serial Number** - The serial number used to activate Document Conversion Service on this computer.
- **License Type** - This is the type of Document Conversion Service license that is activated on this computer. This is one of *Evaluation (Level I)*, *Entry Level*, *Level I*, *Level II*, *Level V* or *Level X*.
- **Installed Version** - The version number of Document Conversion Service currently installed on this computer.
- **Subscription** - The number of days remaining in your current subscription period.
- **Activation Status** - The status of the product activation on this computer.
 - *Not Activated* - Document Conversion Service has been installed but had not yet been activated using your serial number.
 - *Waiting for manual activation* - a PNProdID.txt file has been created to activate manually, and you are waiting for the PNProdAU.txt to be sent to complete activation.
 - *Activated* - Document Conversion Service has been activated using your serial number.

- *Deactivated* - The activation for this computer has been deactivated by PEERNET at the request of the user.
- *Expired* - Your annual subscription has expired. Once the annual renewal is purchase through your [PEERNET online account](#), the status will automatically switch back to Activated.
- *Error* - An event has occurred that has impacted the validity of your activation status.

Changing Your Activation Status

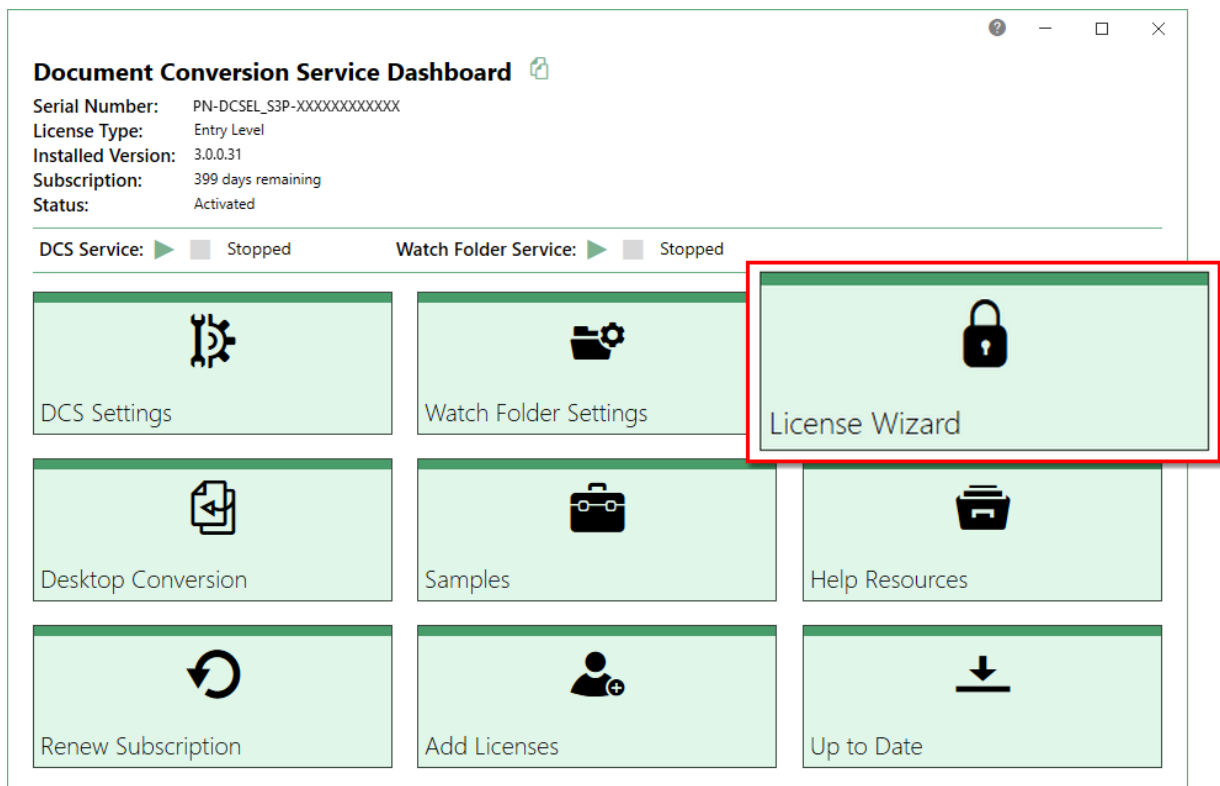
Any changes you need to make to your activation status are available through the **License Wizard**.

To launch the **License Wizard**, begin by opening the **Dashboard**.

Double-click the DCS Dashboard icon on your desktop or select Document Conversion Service 3.0 - DCS Dashboard from the **Start** menu to open the dashboard.



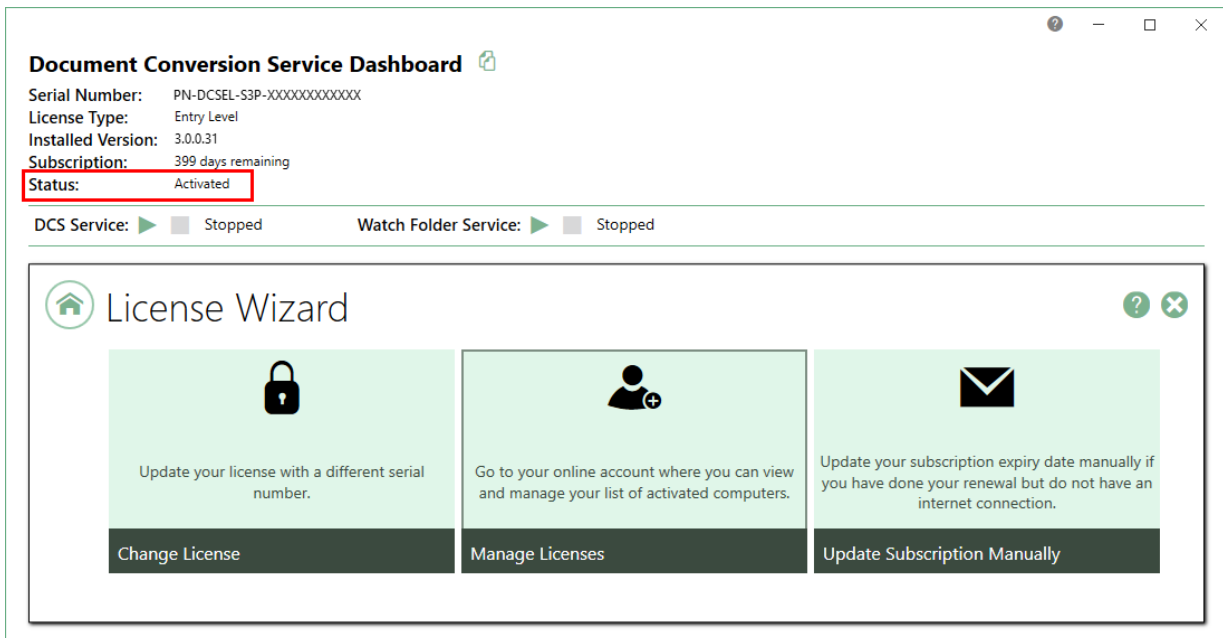
From the dashboard, select **License Wizard** to begin.



Activated

If your current activation status is Activated, there are three options available for changing your activation status:

- **Change License** - This will allow you to enter a new serial number. Use this option when you have merged multiple serial numbers into one serial number, or you have changed to a different license level and need to apply the new serial number.
- **Manage Licenses** - This will take you to your [PEERNET online account](#) where you can [view your activation details](#).
- **Update Subscription Manually** - **Only applicable to users that do not have an internet connection.* After purchasing your annual renewal, this allows you to update the expiry date of your subscription manually if you do not have an internet connection. If you are connected to the internet, the expiry date is updated automatically.

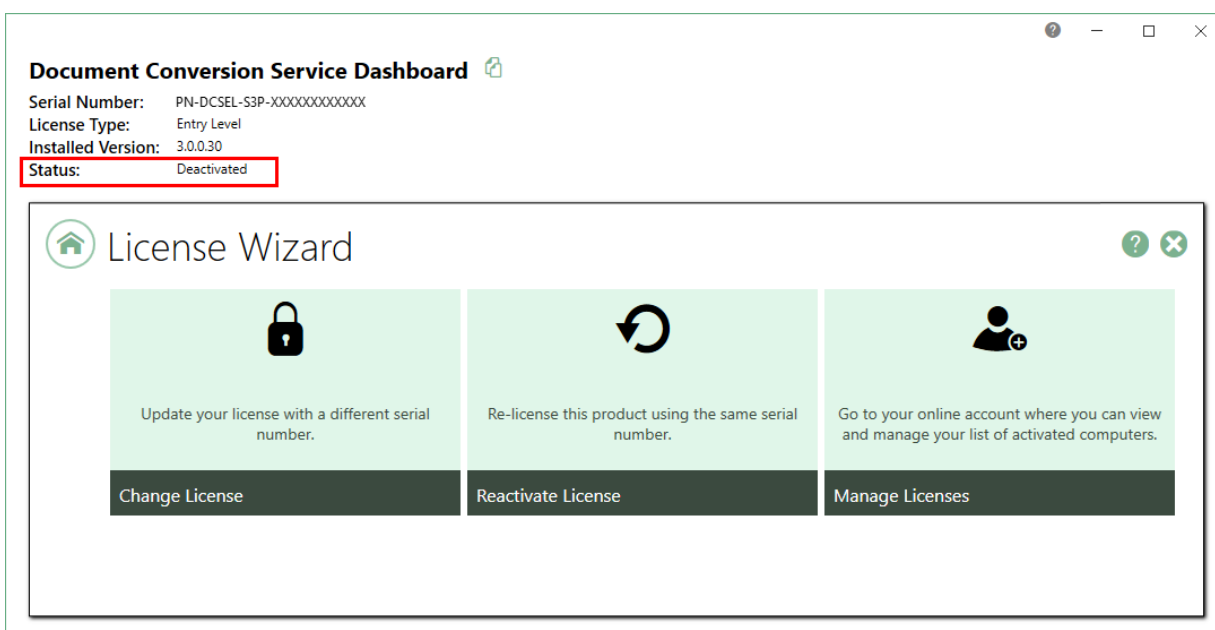


Note: If you are moving your license to a new computer, or if you have to re-install the software on your computer due to a crash, please contact PEERNET Sales at peernet@peernet.com with your current serial number for assistance.

Deactivated

Deactivated means that the license for this computer has been deactivated by PEERNET at the request of the user. If your current activation status is **Deactivated**, there are three options available for changing your activation status:

- **Change License** - This will allow you to enter a new serial number. This would be applicable if you have merged multiple serial numbers into one serial number, or you have changed to a different license level and need to apply the new serial number.
- **Reactivate License** - This will try to activate Document Conversion Service again using the same serial number that was previously used to activate the product. This would be applicable if you have deactivated a license on an old computer that is no longer in use, or have purchased more licenses.
- **Manage Licenses** - This will take you to your [PEERNET online account](#) where you can [view your activation details](#). This would be applicable if you are looking to deactivate a license on an old computer that is no longer in use, so that you can reactivate the deactivated computer.

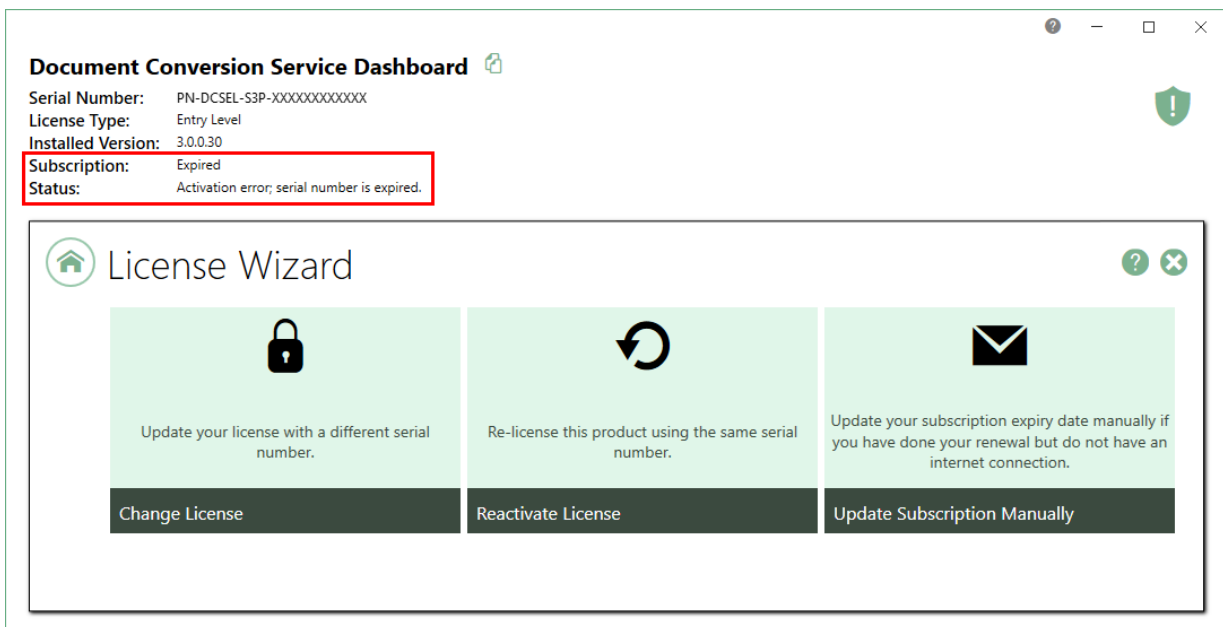


Expired

Expired means that your annual subscription has expired. If your annual subscription has expired, you need to renew your subscription through your [PEERNET online account](#). Once the renewal is purchased, restarting the dashboard will automatically update your license information and set it back to Activated.

If your current activation status is Expired, there are two options available for changing your activation status:

- **Change License** - This will allow you to enter a new serial number. This would be applicable if you have merged multiple serial numbers to one serial number, or if you have upgraded to a higher license level and receive a new serial number.
- **Reactivate License** - This will try to activate Document Conversion Service again using the same serial number that was previously used to activate the product.
- **Update Subscription Manually** - **Only applicable to user that do not have an internet connection.* If you do not have an internet connection the dashboard cannot update your expiry date automatically after purchasing your annual renewal. You will need to update the expiry date of your subscription manually by emailing a PNProdID.txt file.



Error

An error state means that something on the computer has changed so that Document Conversion Service believes it is no longer activated correctly. Examples of items that effect the license validity are: change of domain or workgroup, change in the network, change in the computer name, reformatting and reloading the computer, or system has virus.

If your current activation status is Error, you will be prompted to activate the again. Clicking the "Activate Product" tile takes you to the screen to [enter your serial number](#). Any information

The screenshot shows the 'Document Conversion Service Dashboard' window. At the top, it displays the following information:

- Serial Number: PN-DCSEL-S3P-XXXXXXXXXXXX
- License Type: Entry Level
- Installed Version: 3.0.0.30
- Status: In Error (highlighted with a red box)

Below this information is a 'License Wizard' window. It contains the following fields:

- Serial Number: PN-DCSEL-S3P-XXXXXXXXXXXX
- Name: PEERNET
- Company: PEERNET
- Email: peernet@peernet.com

A green arrow button is located at the bottom right of the License Wizard window.

Renewing Your Annual Subscription

A subscription license of Document Conversion Service must be renewed annually in order to continue using the product. When your annual renewal is approaching, you will begin to receive notifications of the upcoming renewing both via email and via notification messages when you use Document Conversion Service.

You can always see how many days are remaining in your current subscription through the Document Conversion Service **Dashboard** and through your [PEERNET online account](#).

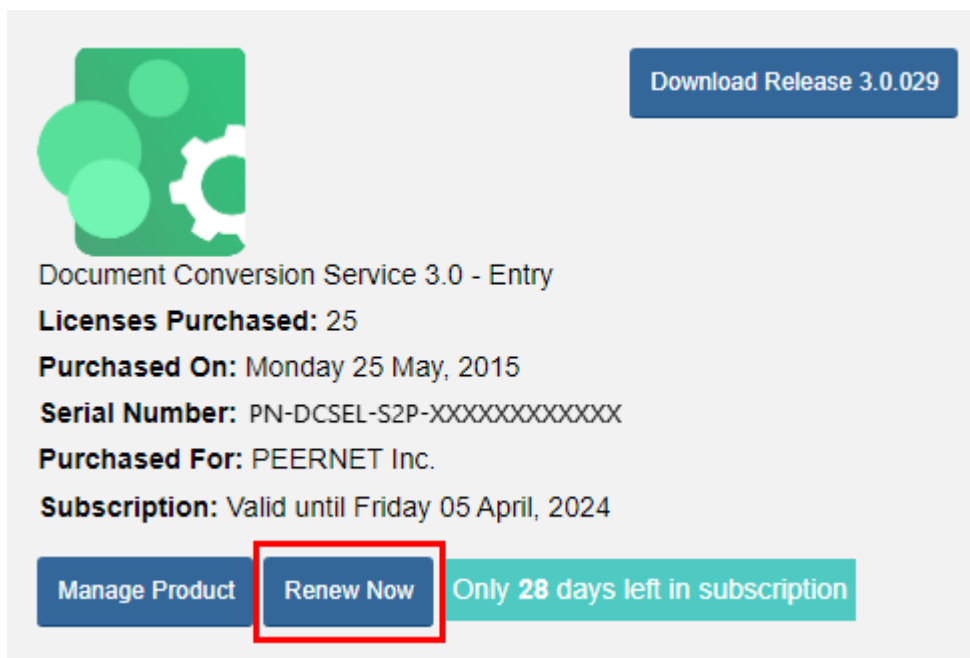
Renewing Document Conversion Service

To renew your product, first log into your [PEERNET online account](#).

On the **My Products** page, you will see all of your purchased PEERNET products. Search the listing for the relevant serial number and select "Renew Now" to put the renewal directly into your shopping cart.

Select "Checkout" and finish your renewal purchase. You will receive an email confirmation of your renewal.

As long as the computer(s) activated with the renewed serial number have an internet connection, the annual renewal will be automatically recognized. The Document Conversion Service **Dashboard** will automatically update to show the new expiry date.



Document Conversion Service 3.0 - Entry

Licenses Purchased: 25

Purchased On: Monday 25 May, 2015

Serial Number: PN-DCSEL-S2P-XXXXXXXXXXXX

Purchased For: PEERNET Inc.

Subscription: Valid until Friday 05 April, 2024

Manage Product **Renew Now** Only 28 days left in subscription

Renewing without an internet connection

If the computer(s) to be updated with the renewed serial number do not have an internet connection, you will need to manually update the expiry date by following the instructions below. These steps must be taken after the annual renewal has been purchased through your [PEERNET online account](#) as shown above.

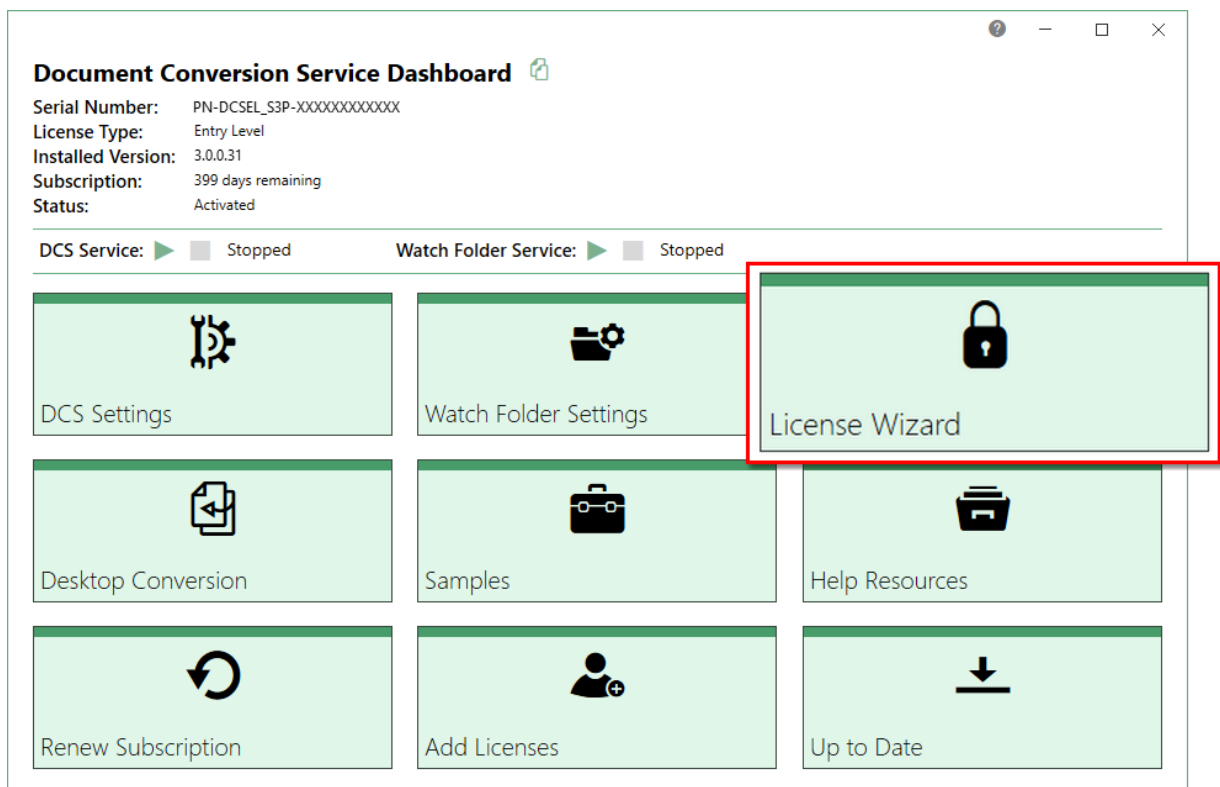
Document Conversion Service 3.0

To launch the **License Wizard**, begin by opening the **Dashboard**.

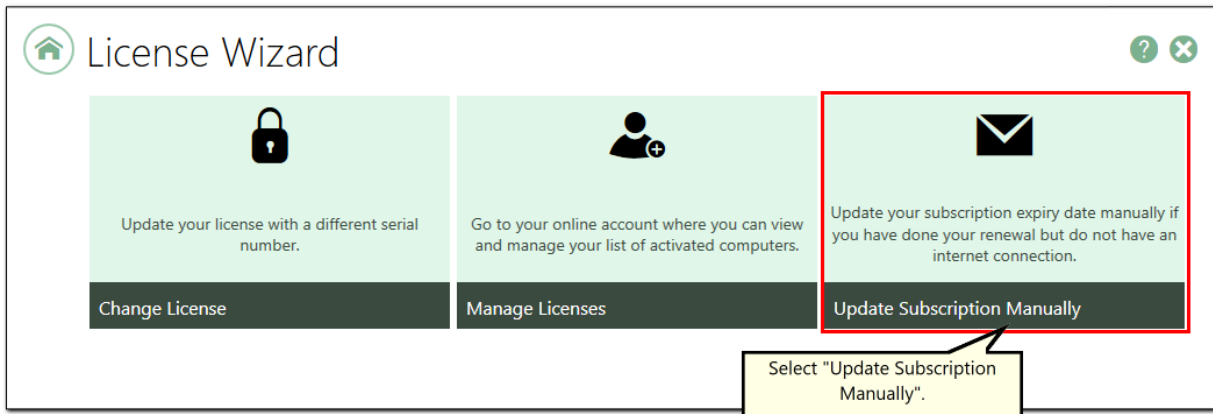
Double-click the DCS Dashboard icon on your desktop or select Document Conversion Service 3.0 - DCS Dashboard from the **Start** menu to open the dashboard.



Select "License Wizard" to open the License Wizard.



From the **License Wizard**, select "Update Subscription Manually".



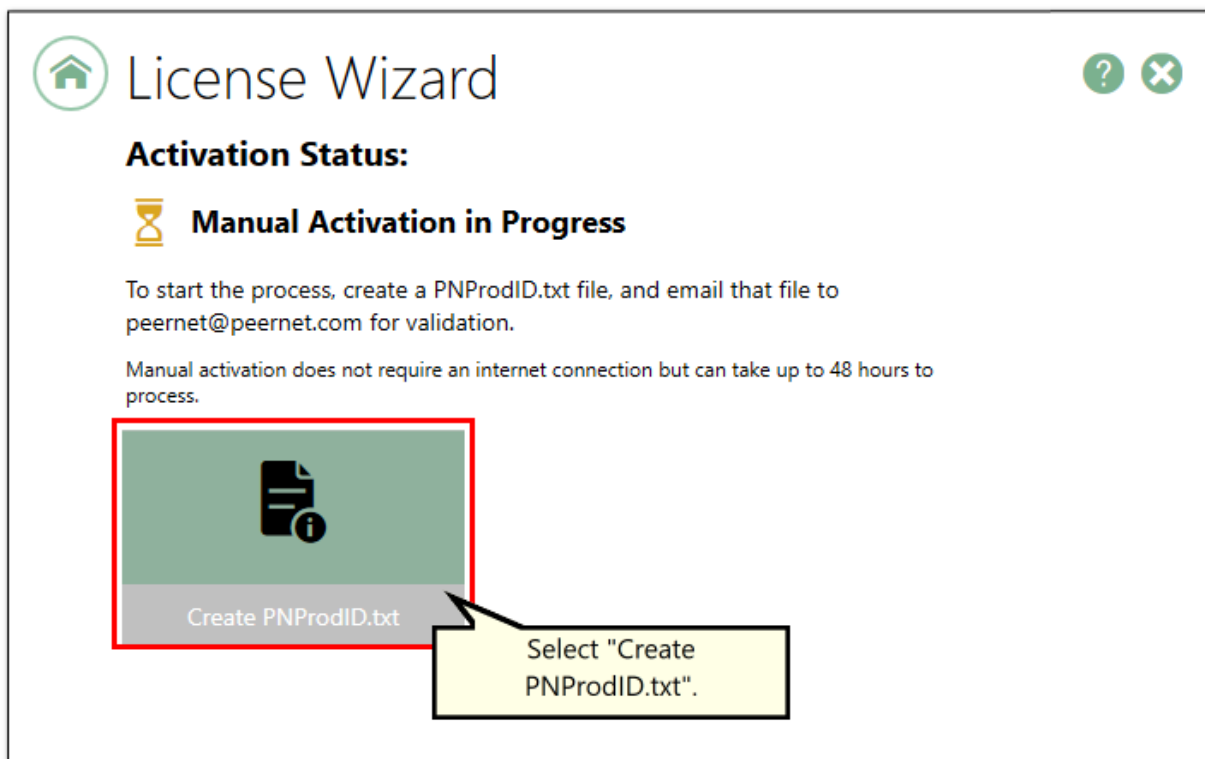
This opens the serial information screen with the serial number, name, company, and email address automatically populated from your original activation. Leave the option "Activate manually by email instead of using the internet" turned on. Click the next arrow to continue.

The screenshot shows the 'License Wizard' window with the following information:

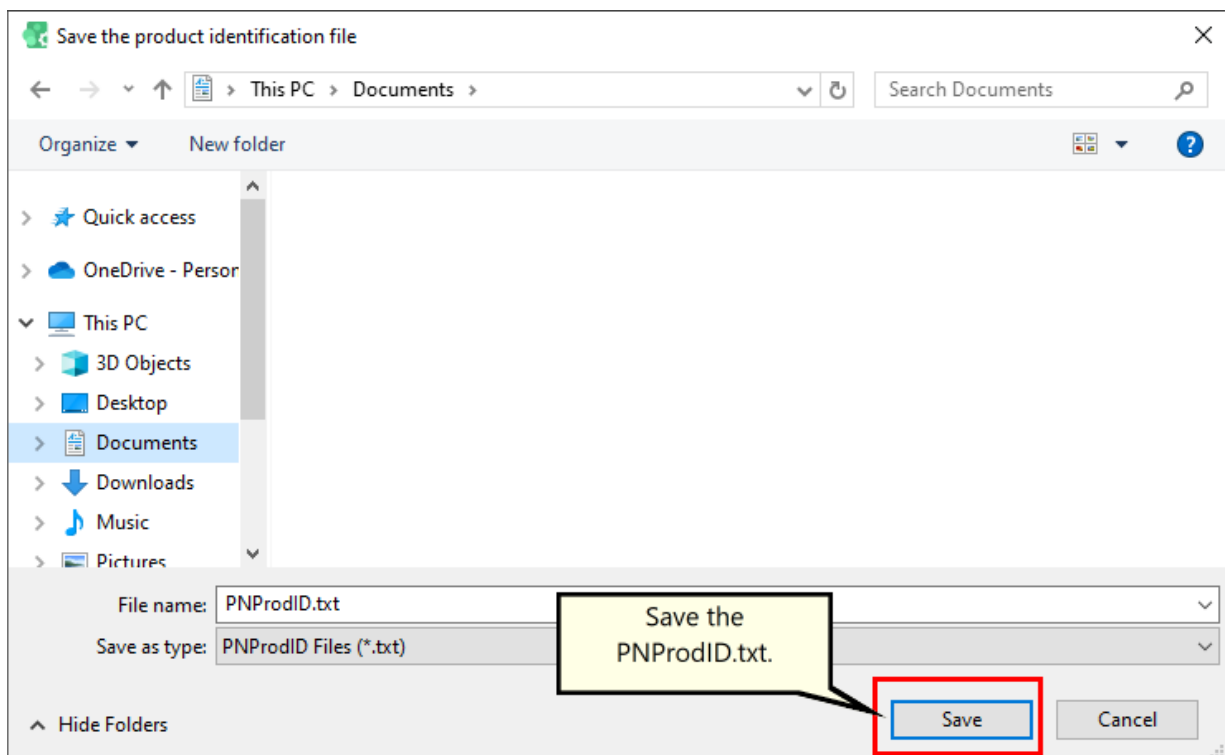
- Serial Number**: PN-DCSEL-S3P-XXXXXXXXXXXX
- Name**: PEERNET
- Company**: PEERNET
- Email**: peernet@peernet.com

The 'Activate manually by email instead of using the internet' toggle is turned on. A callout points to it with the text: "Manual activation using email is turned on". Another callout points to the 'Next' button with the text: "Click Next".

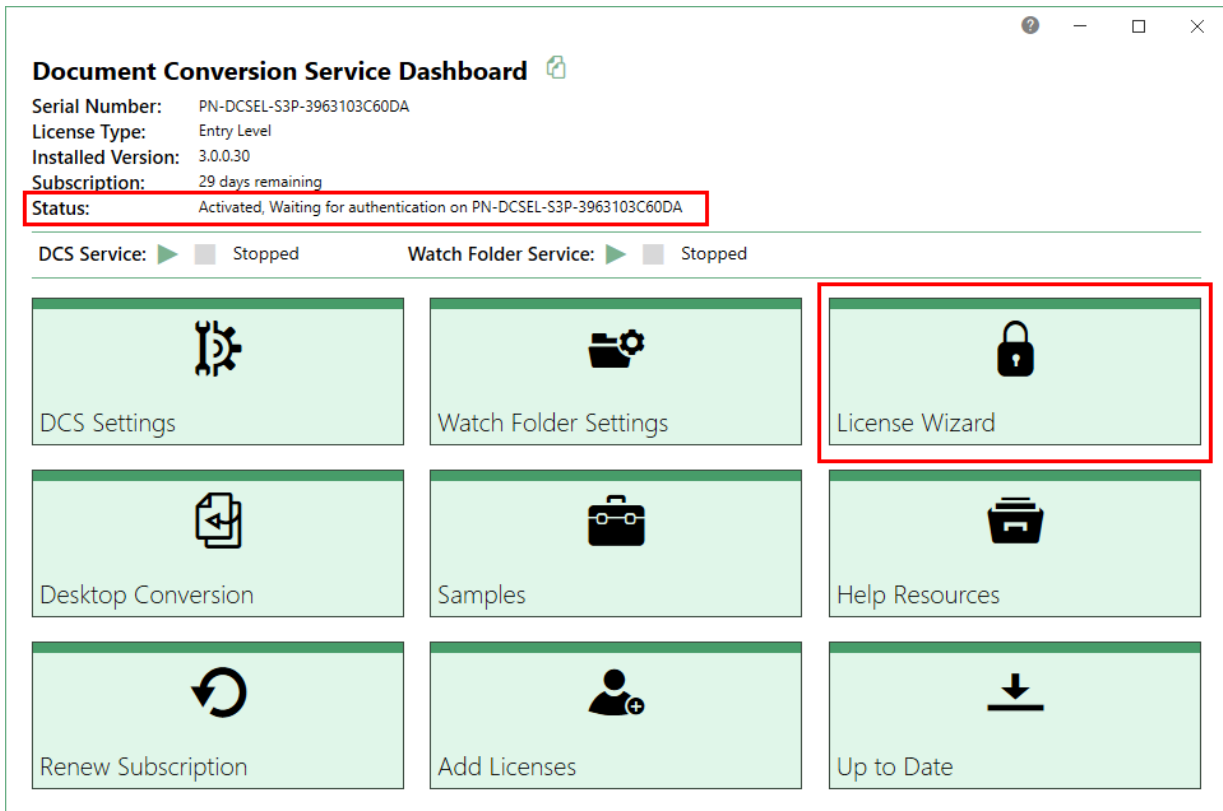
From the next screen click **Create PNProdID.txt**.



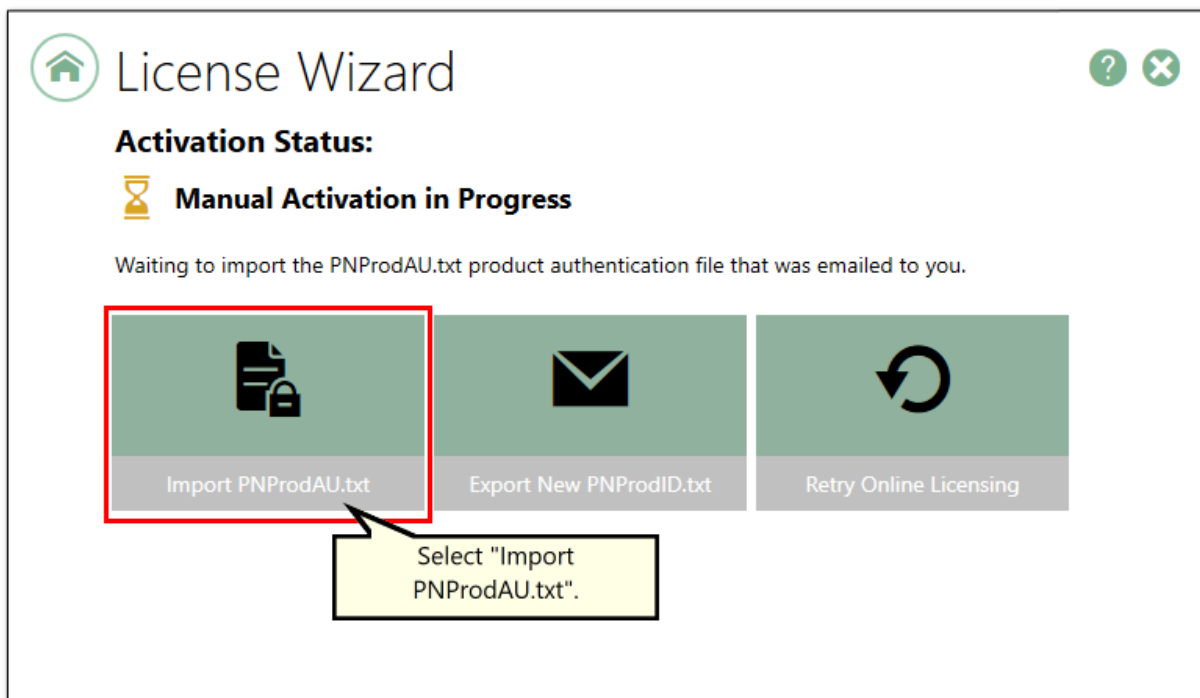
Save the **PNProdID.txt** product identification file and email the file to peernet@peernet.com for manual activation. Please note that these files (PNProdID files) are only authorized during business hours, which are 09h00 to 17h00, Monday through Friday, Eastern Standard Time (excluding statutory holidays).



When you have received the product authentication file **PNProdAU.txt** from PEERNET, restart the **License Wizard** by clicking on the "Finish Manual Activation" tile.



Click **Import PNProdAU.txt** and complete the activation.

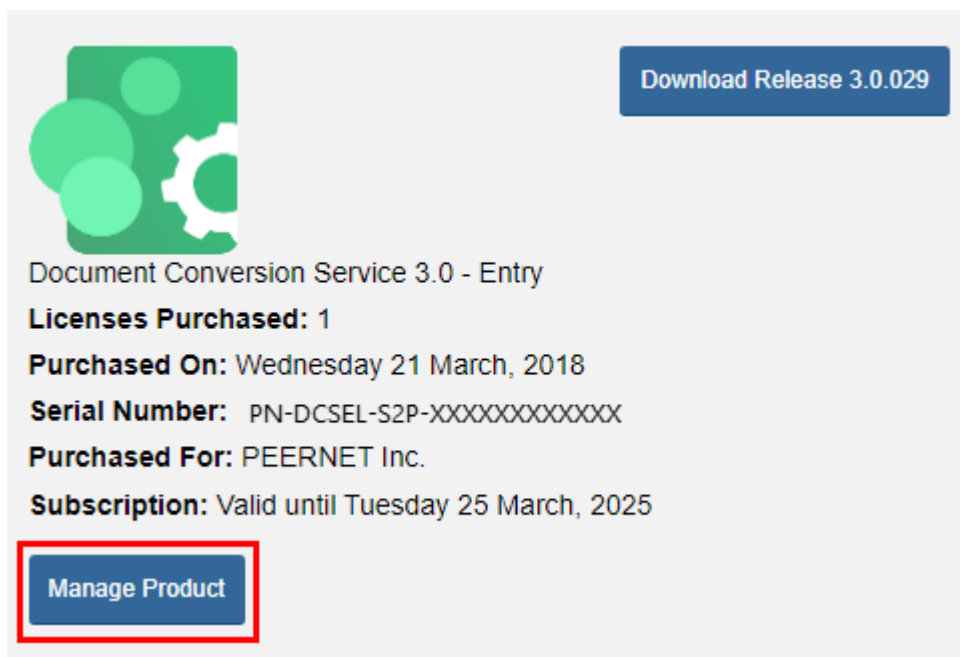


View Activation Details

With the release of Document Conversion Service 3.0.030, users can now view the activation details for their serial number(s) directly through their PEERNET online account.

Edit Activation Details

Log into your [PEERNET online account](#) and find the listing for the relevant serial number in your **My Products** list. Select **Manage Product** to see details for that serial number.



Document Conversion Service 3.0 - Entry

Licenses Purchased: 1

Purchased On: Wednesday 21 March, 2018

Serial Number: PN-DCSEL-S2P-XXXXXXXXXXXX

Purchased For: PEERNET Inc.

Subscription: Valid until Tuesday 25 March, 2025

[Manage Product](#)

Scroll down to the **Activation Details** section. Here you will see a list of all computers currently activated using this serial number.

ACTIVATION DETAILS				
Activated/Validated On	Email	User Name	Computer Name	
Wednesday 21 March, 2018	peernet@peernet.com	PEERNET	WIN-TESTING	

If you are running Document Conversion Service 3.0 on a server that is being decommissioned, and need to install and activate your license on a new server prior to the decommissioning date, please contact [PEERNET](#) with your serial number, old computer name(s), and decommissioning date.

Working With Document Conversion Service

Document Conversion Service consists of the following components:

- A [dashboard application](#) for quick access to licensing, managing and configuring DCS and Watch Folder services, samples and help resources.
- The logging console used to monitor Document Conversion Service when it is running.
- A user account, DCSAdmin, with administrative privileges, that can optionally be created during application installation. This account, or another pre-existing account with administrative privileges is required to run Document Conversion Service.
- The Document Conversion Service, the main application that performs the conversion, which includes:
 - A collection of converters for converting the most commonly used document formats.
 - A suite of command line utilities for converting files and folders that can be called from the command line, in batch files, as scheduled tasks, or from any application that can call an external program.
 - A .NET library, PEERNET.ConvertUtility.dll for converting files and folders that is callable from any language; C++, C#, VB, PowerShell, and others.
 - A default payload plug-in for converting files; this payload can be called from any programming language with COM support.
 - Advanced configuration through its application configuration file.
- PEERNET Document Conversion Service Monitor 1.0, the Windows service that runs and monitors the Document Conversion Service.
- Document Conversion Service 3.0 printer used by the Document Conversion Service.
- A system tray application that also provides quick access to common tasks.

The DCSAdmin Account

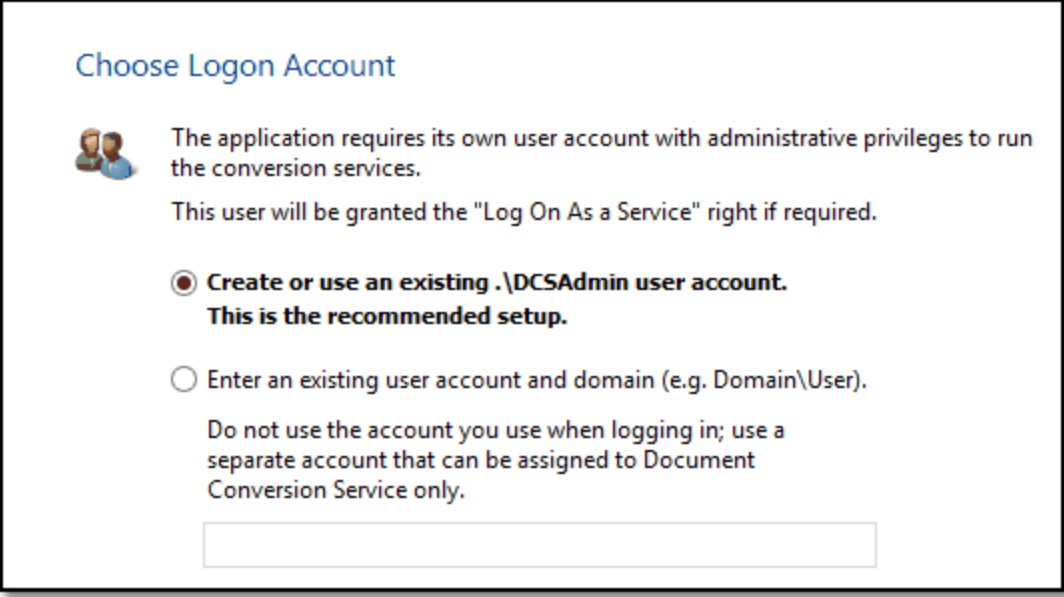
Document Conversion Service requires a user account with administrative privileges to run the underlying conversion services. This account can be created as a local account, or a pre-existing account can be chosen to be used during the application installation. This account is used by the PEERNET Document Conversion Service Monitor 1.0 service as well as the included Watch Folder Service sample.

We recommend creating and using the DCSAdmin for your conversion service. This provides you with a clean environment in which to run your document conversions and ensures that all native applications used by the converters will be automatically configured correctly for use with Document Conversion Service.

Creating the DCSAdmin During Installation


During installation you are prompted to create the DCSAdmin account or to choose an already existing account. You will be asked to create or supply the existing password on the next screen.

If you are planning on converting files remotely using DCOM, as outlined in the section [Converting on a Remote Computer \(DCOM\)](#), you will need to choose the second radio button and supply an account on the domain with administrative rights. If you have created, or are using, a Domain Group for the purposes of granting DCOM permissions, this user needs to be part of that group.



The screenshot shows a dialog box titled "Choose Logon Account". It contains the following text and options:

Choose Logon Account

 The application requires its own user account with administrative privileges to run the conversion services.
This user will be granted the "Log On As a Service" right if required.

☒ **Create or use an existing .\DCSAdmin user account.**
This is the recommended setup.

☐ Enter an existing user account and domain (e.g. Domain\User).
Do not use the account you use when logging in; use a separate account that can be assigned to Document Conversion Service only.

Below the radio buttons is a text input field.

If you forget the the DCSAdmin password.

If you are upgrading versions or re-installing Document Conversion Service and have forgotten the original password you used when creating the DCSAdmin, you will need to do one of the following:

Re-install Document Conversion Service

This is the simplest method, but care must be taken to backup any configuration changes you may have made.

1. Stop any running Document Conversion Service services. This includes the Watch Folder Service if you are using it.
 - a. See [Starting and Stopping the Watch Folder Service](#) to stop the watch folder service.
 - b. Stop the Document Conversion Service as outlined in [Starting and Stopping the Service](#).
 - c. If you have any of your own services or programs, stop those as well.
2. Uninstall Document Conversion Service, either through Add/Remove Programs in the Control Panel or by going to Start - All Programs - PEERNET Document Conversion Service 3.0 - Uninstall Document Conversion Service 3.0.
3. Go to Control Panel - System and Security - Administrative Tools and select Computer Management.
 - a. Under the *Local Users and Groups* item, select *Users* and delete the existing DCSAdmin account. It will warn you about deleting an account with administrative access, but continue with the deletion as we will be re-creating this account when Document Conversion Service is installed.
4. Re-install Document Conversion Service and re-create the DCSAdmin as part of the install.

Delete the Existing DCSAdmin and Create a New One Manually.

This allows you to keep any configuration changes you have made to Document Conversion Service. When doing this step you will also need to reset the logon account for the PEERNET Document Conversion Service Monitor 1.0 service and the Watch Folder Service, if you are using it.

1. Stop any running Document Conversion Service services. This includes the Watch Folder Service if you are using it.
 - a. See [Starting and Stopping the Watch Folder Service](#) to stop the watch folder service.
 - b. Stop the Document Conversion Service as outlined in [Starting and Stopping the Service](#).
 - c. If you have any of your own services or programs, stop those as well.
2. Go to Control Panel - Administrative Tools and select Computer Management.
3. Under the *Local Users and Groups* item, select *Users* and delete the existing DCSAdmin account. It will warn you about deleting an account with administrative access, but continue with the deletion as we will be re-creating this account in the next step.

4. Right-click in the center panel and select *Create User...* to create a new user.
 - a. Enter *DCSAdmin* as the user name and type in a new password.
 - b. Uncheck the *User must change password at next logon* option and check *Password never expires*.
 - c. Click the Create button to create the user, then Close to exit.

New User

User name: DCSAdmin

Full name: Document Conversion Service User Account

Description:

Password:

Confirm password:

☐ User must change password at next logon

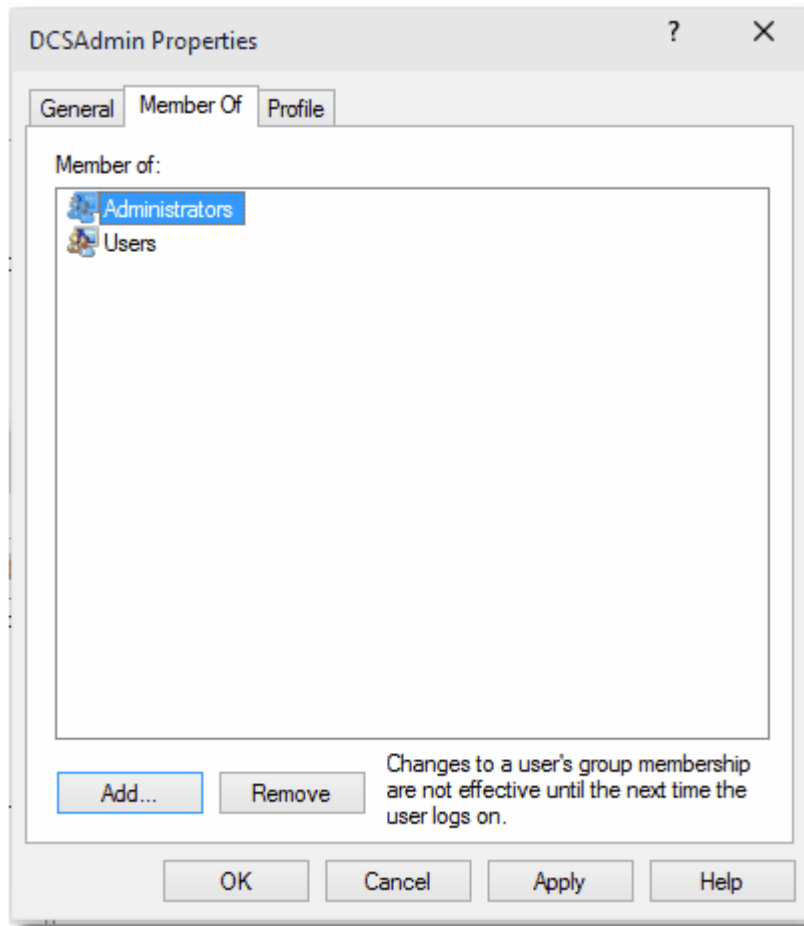
☐ User cannot change password

☒ Password never expires

☐ Account is disabled

Help Create Close

5. Once the new account has been created, it also has to be added to the local Administrators group.
 - a. Double-click the new DCSAdmin user to bring up the properties dialog for the user.
 - b. On the *Member Of* tab add the *Administrators* group to the list of groups in which DCSAdmin is a member. Click Apply and OK to save the changes.



6. Now that the account has been created, the PEERNET Document Conversion Service Monitor 1.0 service needs to be updated to use the new account. Follow the steps for [Changing the Service Log On Account](#) to complete this step. If the Watch Folder Service is being used, you will need to update the Log On account for that service as well.

What Files Can I Convert?

A basic set of converters is included with Document Conversion Service to provide conversion of the most commonly used document formats. These converters use the concept of application pooling (running multiple instances of each application or converter) to achieve fast document conversion with high throughput and fault tolerance for rogue processes.

Some converters are stand-alone and do not depend on any other third-party applications. Other converters use a third-party application to convert the file to ensure that the converted file matches the original document as closely as possible. For these converters, this means that the necessary third-party applications must be installed and licensed for the converters you need to use.

While Document Conversion Service configures these third-party applications automatically where possible, some applications require other components or custom configuration as outlined in [Configuring Third-Party Applications Used by Document Conversion Service](#) section.



Installing New Applications

If you install new applications while the Document Conversion Service is running, you will need to restart Document Conversion Service in order to detect and use the new applications.

When starting, Document Conversion Service auto-detects what converters to run. The default is to try and start all converters. A stand-alone converter will always run. A converter that needs a third-party application will only run if the application is on the computer.


Setting a Converter Start-Up State

You can customize the start-up state for each converter and turn off converters that handle file types you do not need to convert.


From the **DCS Dashboard**, go to **DCS Settings** and click the **Edit DCS Configuration** tile. Clicking this tile opens a flyout that lists each converter with its current startup option.

A converter has three modes:

- **Auto** - DCS auto-detects if this converter and any third-party application it may require are available and can be loaded. When DCS cannot load a converter set to auto, it skips it and moves to the next one.
- **On** - Document Conversion Service will always try to load the converter. If it cannot, the service will not start. This option is a good choice for file types you require to be able to be converted.
- **Off** - You can turn off converters for file types you do not need to convert.



Edit DCS Configuration



Choose which converters you want available to use when converting files.

AUTO - DCS will auto-detect if the converter is available and will load it.
ON - Set a converter ON when it is required to be available.
OFF - Turn off converters for files types you are not converting.

Choose Converters

☒ Set all converters to Auto

Auto	PEERNET ArcPDF
Auto	Adobe PDF - Builtin
Auto	Image - Builtin
Auto	Text - Builtin
Auto	Cadd - Builtin
Auto	Microsoft Word
Auto	Microsoft Excel
Auto	Microsoft PowerPoint
Auto	Microsoft Outlook
Auto	Microsoft Publisher
Auto	Microsoft Visio
Auto	Microsoft XPS
Auto	Adobe Acrobat Reader
Auto	Internet Explorer
Auto	Autodesk Design Review

SAVE CHANGES GO TO ADVANCED EDITOR

After making your changes to the configuration, click the **Save Changes** button at the bottom to save them. If the DCS service is running, you will need to [stop and restart the service](#) for the changes to take effect.

The **Go To Advanced Editor** button will open the XML-formatted configuration file in the [DCS Editor](#) for manual editing. This editor view gives access to advanced configuration options that normally do not need to be changed. See [Controlling the Converters](#) in the Advanced Configuration section for more information.

Converter File Type Table

The table below outlines each converter, the file types it converts, and any required application it needs.



Caution

The Adobe PDF - Builtin, Cadd - Builtin, Image - Builtin, and Text - Builtin converters are not supported on Microsoft® Windows Server 2008 R2 and Microsoft® Windows 7.

Supported Document Type and Converter Name	Third-Party Application
Converter Name: Adobe PDF - Builtin <ul style="list-style-type: none"> • Adobe PDF Documents (*.pdf) 	Built-in, no additional applications required.
Converter Name: Adobe Acrobat Reader <ul style="list-style-type: none"> • Adobe PDF Documents (*.pdf) 	Adobe Reader X, XI, DC (32-bit only)
Converter Name: Cadd - Builtin <ul style="list-style-type: none"> • AutoDesk 2D Drawings (*.dwf, *.dwfx) • HPGL Plot files (*.plt) • Gerber RS-274D, Gerber RS-274X, Gerber X2 (*.gbx) 	Built-in, no additional applications required.
Converter Name: Autodesk Design Review <ul style="list-style-type: none"> • Design Review Drawings (*.dwf) 	Autodesk Design Review 2012-2013, 2018
Converter Name: Autodesk Design Review <ul style="list-style-type: none"> • AutoCAD Drawings (*.dwg) 	Autodesk Design Review 2012-2013 with DWG TrueView 2012-2013 also installed. Autodesk Design Review 2018 with DWG TrueView 2018 also installed.
Converter Name: Microsoft Excel <ul style="list-style-type: none"> • Excel Workbooks (*.xlsx, *.xlsm, *.xls) • Excel Templates (*.xltx, *.xltm, *.xlt) • Excel Binary Workbook (*.xlsb) 	Microsoft Office 2003 SP3 <i>(with Microsoft Office Compatibility Pack)</i> Microsoft Office 2007 (32-bit and 64-bit) Microsoft Office 2010 (32-bit and 64-bit) Microsoft Office 2013 (32-bit and 64-bit) Microsoft Office 2016 (32-bit and 64-bit)

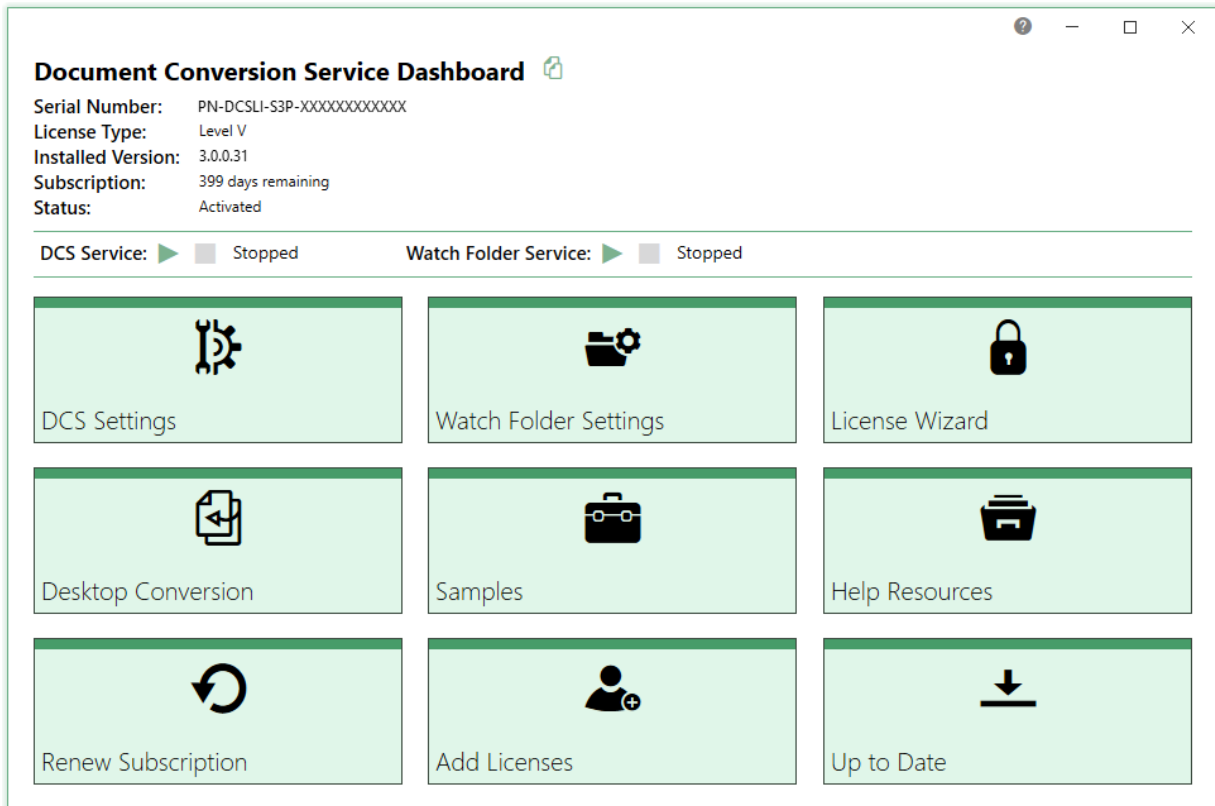
Supported Document Type and Converter Name	Third-Party Application
	Microsoft Office 2019 (32-bit and 64-bit) Microsoft Office 2021 (32-bit and 64-bit) Microsoft Office 365 (32-bit and 64-bit)
<p>Converter Name: Image - Builtin, PEERNET Image Converter</p> <ul style="list-style-type: none"> • JPEG images (*.jpg) • TIFF images (*.tif) • High Efficiency Image Files (*.heif, *.heic) • Google WebP Images (*.webp) • AVIF Images (*.avif) • Windows Bitmap images (*.bmp) • ZSoft PCX images (*.pcx) • ZSoft DCX images (*.dcx) • CServe Portable Network Graphics images (*.png) • Graphics Interchange Format image files (*.gif) • Icon Format (*.ico) • Windows Media Photo images (*.wdp, *.hdp, *.jxr) • ImageMagick images (100+ image formats) 	<p>Built-in, no additional applications required, includes optional support for animated images and movies with ffmpeg.exe.</p>
<p>Converter Name: PEERNET Image Converter</p> <ul style="list-style-type: none"> • DejaVu files(*.djvu) 	<p>Requires DjVu Shell Extension Pack</p>
<p>Converter Name: PEERNET Image Converter</p> <ul style="list-style-type: none"> • 45+ image formats and 500+ raw digital camera formats 	<p>Requires FastPictureViewer Codec Pack</p>
<p>Converter Name: Internet Explorer</p> <ul style="list-style-type: none"> • HTML Files (*.htm, *.html) • Secure HTML (*.shtm, *.shtml) • Web Archive (*.mht) 	<p>Internet Explorer 8.0 - 11.0</p>
<p>Converter Name: Microsoft Outlook</p> <ul style="list-style-type: none"> • Outlook Message Files (*.msg) • Outlook Templates (*.oft) • vCard Files (*.vcf) • vCalendar Appointment Files (*.vcs) • iCalendar Appointment Files (*.ics) • Electronic Mail messages (*.eml) 	<p> Microsoft Office 2003 (*.oft and *.msg only) Microsoft Office 2007 (32-bit and 64-bit) Microsoft Office 2010 (32-bit and 64-bit) Microsoft Office 2013 (32-bit and 64-bit) Microsoft Office 2016 (32-bit and 64-bit) Microsoft Office 2019 (32-bit and 64-bit) Microsoft Office 2021 (32-bit and 64-bit) Microsoft Office 365 (32-bit and 64-bit) </p>

Supported Document Type and Converter Name	Third-Party Application
<p>Converter Name: Outside-In AX</p> <p>Supports over 500 common file formats; see the documentation that came with your Outside In Technology product.</p>	<p>Oracle Outside In Viewer Technology (ActiveX)</p>
<p>Converter Name: Ghostscript</p> <ul style="list-style-type: none"> • Postscript Files (*.ps) • Encapsulated Postscript Files (.eps) • Adobe PDF Documents (*.pdf) 	<p>Ghostscript 9.05 or later (32-bit only)</p> <p><i>There are known handle leak issues with earlier 9.0X versions of Ghostscript.</i></p>
<p>Converter Name: Microsoft PowerPoint</p> <ul style="list-style-type: none"> • PowerPoint Presentations (*.pptx, *.pptm, *.ppt) • PowerPoint Shows (*.ppsx, *.ppsm, *.pps) • PowerPoint Templates (*.potx, *.potm, *.pot) 	<p>Microsoft Office 2003 SP3 (with Microsoft Office Compatibility Pack)</p> <p>Microsoft Office 2007 (32-bit and 64-bit)</p> <p>Microsoft Office 2010 (32-bit and 64-bit)</p> <p>Microsoft Office 2013 (32-bit and 64-bit)</p> <p>Microsoft Office 2016 (32-bit and 64-bit)</p> <p>Microsoft Office 2019 (32-bit and 64-bit)</p> <p>Microsoft Office 2021 (32-bit and 64-bit)</p> <p>Microsoft Office 365 (32-bit and 64-bit)</p>
<p>Converter Name: Microsoft Publisher</p> <ul style="list-style-type: none"> • Publisher Files (*.pub) 	<p>Microsoft Office 2003 SP3 (with Microsoft Office Compatibility Pack)</p> <p>Microsoft Office 2007 (32-bit and 64-bit)</p> <p>Microsoft Office 2010 (32-bit and 64-bit)</p> <p>Microsoft Office 2013 (32-bit and 64-bit)</p> <p>Microsoft Office 2016 (32-bit and 64-bit)</p> <p>Microsoft Office 2019 (32-bit and 64-bit)</p> <p>Microsoft Office 2021 (32-bit and 64-bit)</p> <p>Microsoft Office 365 (32-bit and 64-bit)</p>
<p>Converter Name: Text - Builtin</p> <ul style="list-style-type: none"> • Text Files (*.txt) • Logging Files (*.log) • Source Files(*.c, *.cpp, *.cs, etc.) • Config and INI Files (.config, *.ini) 	<p>Built-in, no additional applications required.</p>

Supported Document Type and Converter Name	Third-Party Application
<ul style="list-style-type: none"> PowerShell (*.ps1) Batch Files (*.cmd, *.bat) Any other text-based file 	
<p>Converter Name: Microsoft Visio</p> <ul style="list-style-type: none"> Visio Drawings (*.vsd) 	<p>Microsoft Visio 2003 Microsoft Visio 2007 Microsoft Visio 2010 (32-bit and 64-bit) Microsoft Visio 2013 (32-bit and 64-bit) Microsoft Visio 2016 (32-bit and 64-bit)</p>
<p>Converter Name: Microsoft Word</p> <ul style="list-style-type: none"> Word Documents (*.docx, *.docm, *.doc) Word Templates (*.dotx, *.dotm, *.dot) Rich Text Documents (*.rtf) 	<p>Microsoft Office 2003 SP3 (with Microsoft Office Compatibility Pack) Microsoft Office 2007 (32-bit and 64-bit) Microsoft Office 2010 (32-bit and 64-bit) Microsoft Office 2013 (32-bit and 64-bit) Microsoft Office 2016 (32-bit and 64-bit) Microsoft Office 2019 (32-bit and 64-bit) Microsoft Office 2021 (32-bit and 64-bit) Microsoft Office 365 (32-bit and 64-bit)</p>
<p>Converter Name: Microsoft XPS</p> <ul style="list-style-type: none"> XPS Documents (*.xps) Open XPS Documents (*.oxps) 	<p>Uses Windows built-in XPS document support, no additional applications required.</p>
<p>Converter Name: PEERNET Passthrough</p> <ul style="list-style-type: none"> Any file type 	<p>Built-in, passes the file through the system without converting.</p>

The DCS Dashboard

The **Dashboard** is your hub for all things Document Conversion Service. It combines [product activation](#) and [status](#) with all tools and resources for Document Conversion Service into one place. It provides easy access to starting and stopping both Document Conversion Service and the Watch Folder Service. You can quickly find and edit the Document Conversion Service and Watch Folder Service configurations, add and edit profiles, and access the logging console and saved conversion logs. Other dashboard tiles offer fast access to sample programs, video tutorials, and other help resources.



Starting and Stopping DCS Services

At the top of the dashboard you have easy access to starting and stopping both Document Conversion Service and Watch Folder Service.

DCS Service: Stopped Watch Folder Service: Stopped

License Information

The **Dashboard** displays all status information about your current evaluation or activated license. Once you have activated your purchased copy of the software, you will find the following information in the top left-hand corner of the **Dashboard**:

- **Serial Number** - appears when product is activated.
- **License Type** - the type of license installed, such as Evaluation, Entry Level, etc.
- **Installed Version** - the current version of the product.
- **Subscription** - number of days remaining in your current subscription.
- **Status** - the current activation state.

Tools and Resources

All tools and resources from the Start menu are available through the Dashboard, divided into categories to make them easier to find.

- [DCS Settings](#) - go here to edit the DCS configuration file, manage the file extensions to converting mapping file, edit conversion profiles, view the conversion log or saved logs folder, and find tools for merging, backing up, and restoring configuration files.
- [Watch Folder Settings](#) - from this tile, you can edit the Watch Folder Service configuration, view its conversion log, and open its saved logs folder.
- [Activate Product/License Wizard](#) - depending on the state of your license, the title on this tile will vary. This is where you can activate your product, change your serial number, and manage your licenses.
- [Desktop Conversion](#) - provides access to desktop conversion utilities for converting files and folders of files, and using the command line conversion tools from the command prompt.
- [Samples](#) - go here to open the Visual Studio projects for the Convert File (C#/VB) and Watch Folder Service (C#) utilities, view the PowerShell sample or open the Samples folder directly.
- [Help Resources](#) - this tile contains links to online resources, video tutorials, quick start and user guides for Document Conversion Service, the command line tools, and the NET PEERNET.ConvertUtility API for custom programming and development.

The following tiles are available in the dashboard once the product has been activated.

- **Renew Subscription** - click this tile to log into your PEERNET online account where you can renew your annual subscription.
- **Add Licenses** - select this tile to log into your PEERNET online account where you can purchase additional licenses of Document Conversion Service on the same serial number.
- **Update Available/Check for Updates** - this tile will notify you when there is an update pending for download and install, or take you to your PEERNET online account to see what downloads are available to you.

The System Tray Icon

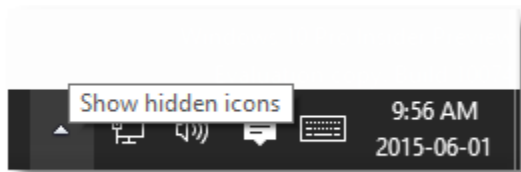
The system tray icon gives quick access to the following:


- running and stopping the Document Conversion Service
- the DCS Dashboard
- the logging console
- viewing saved log files

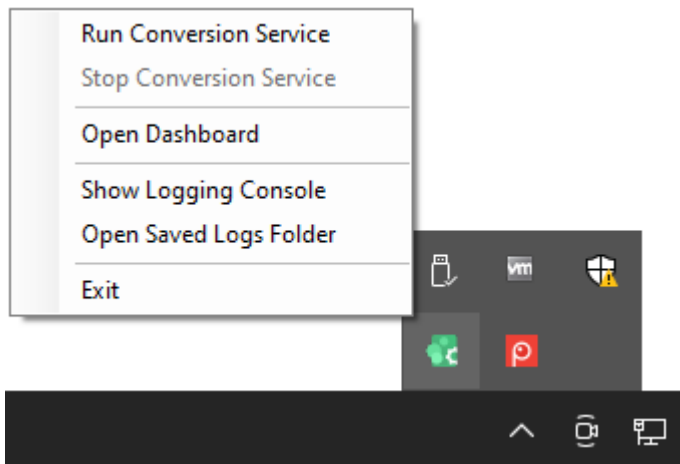
The System Tray Icon

The system tray icon is installed in the **Windows Startup** folder and automatically runs on startup. The icon will appear in the notification area on the far right of the taskbar. If the tray icon is missing, launch it from the DCS Dashboard by going into [DCS Settings](#) or clicking its shortcut in the Start menu.

1. If the icon isn't visible, click the arrow icon next to the notification area to see all hidden icons.



2. Click the system tray icon  to show the menu; both a left-click and a right-click will display the menu.



3. If the icon is not visible in the hidden icons window, you can launch it manually. From the **DCS Dashboard**, go to **DCS Settings** and click the tile **Show Tray Icon**. You can also go to Start - All Programs - Tools - PEERNET Document Conversion Service 3.0 - Show Tray I con to launch the system tray tool.

The Logging Console

The included logging console, the SmartInspect Redistributable Console, is used to monitor Document Conversion Service when it is running. It allows live logging that can be used to troubleshoot service start up issues and conversion errors should any occur.

When opened, the console will automatically start displaying any logging information coming from a running Document Conversion Service. All information being displayed is also saved to a rotating series of text files; see [Viewing Saved Logs](#) for more information.



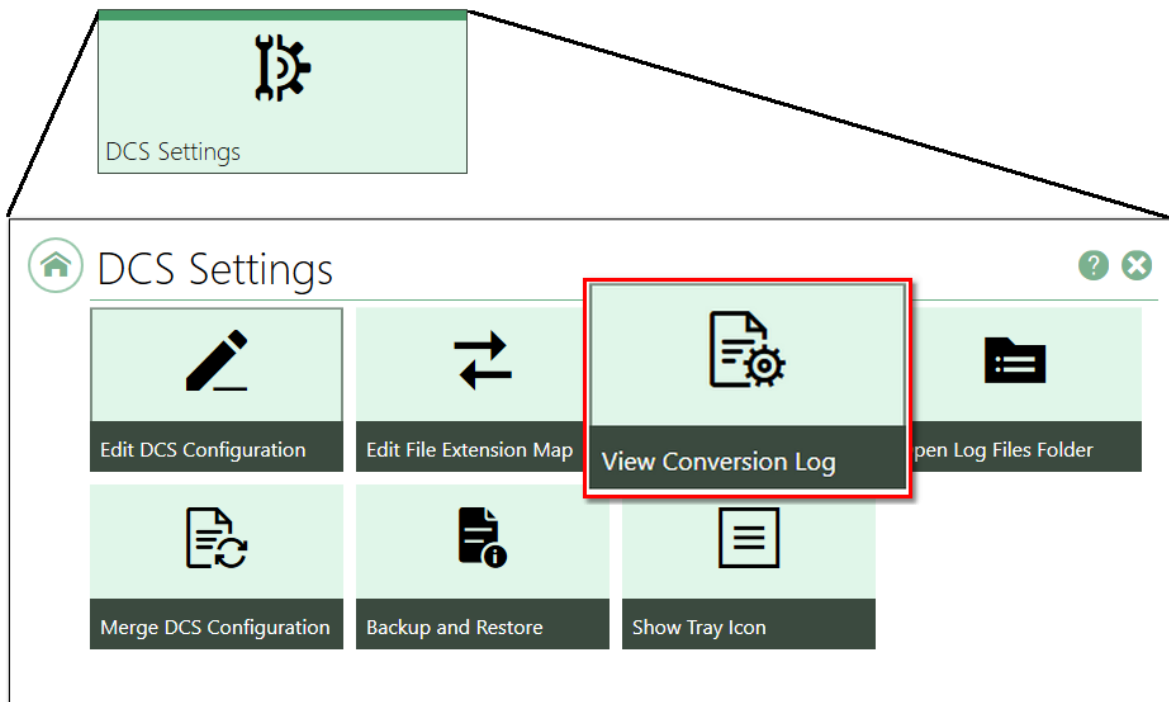
Do Not Leave the SmartInspect Redistributable Console Open

The SmartInspect Redistributable Console is meant for short term, live logging and troubleshooting access. Do not leave the logging console open for extended periods of time, such as overnight, or it will lock and cause issues with Document Conversion Service.

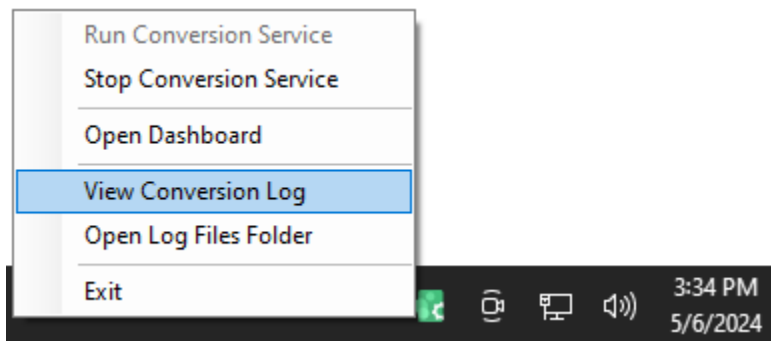
How to Open the Logging Console

The logging console can be opened from the [DCS Dashboard](#), the [system tray icon](#) or from the Start menu.

1. In the Document Conversion Service Dashboard, click the **DCS Settings** tile to open the panel with the DCS settings options. Click the **View Conversion Log** tile to open the logging console.



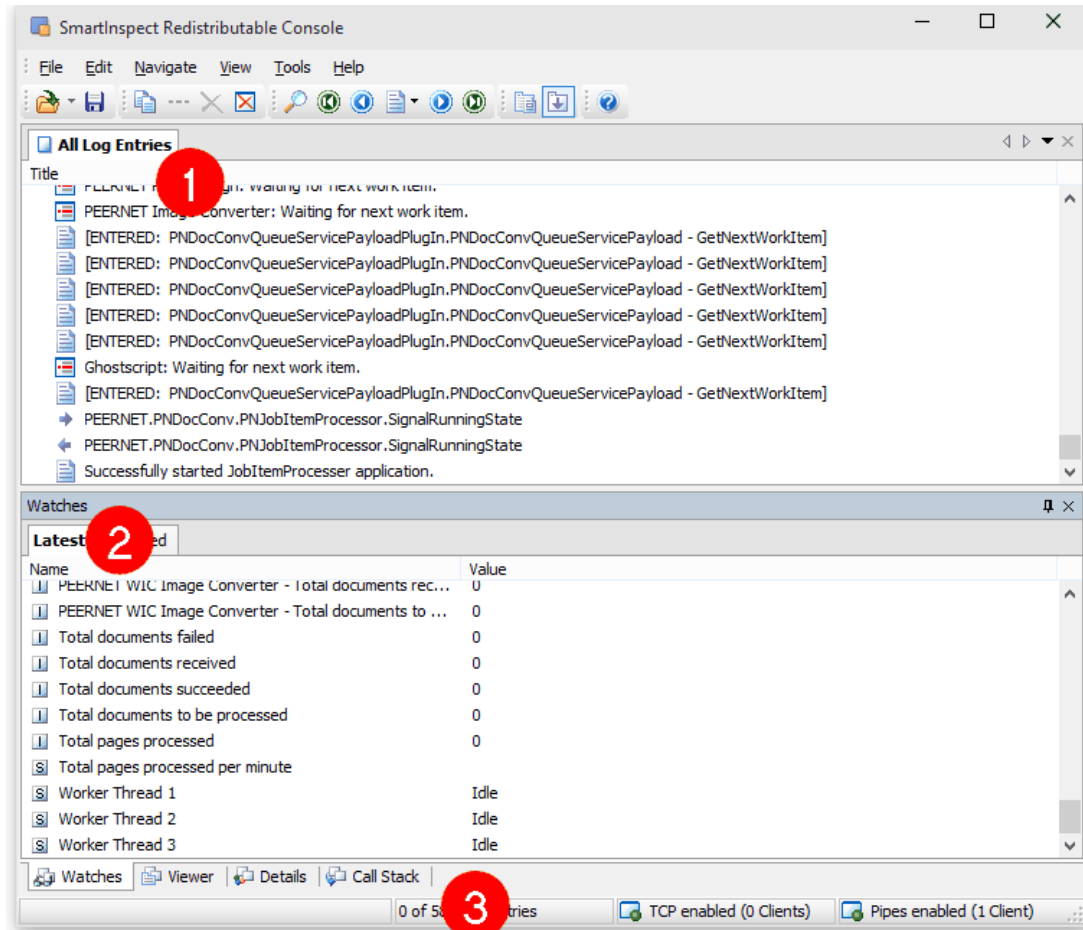
2. You can open the logging console from the [system tray icon](#) through the View Conversion Log menu item.



3. You can also open the logging console from Start - All Programs - PEERNET Document Conversion Service 3.0 - Open DCS Log Viewer.

Logging Console Main Window

The SmartInspect Redistributable Console main window consists of a view of the log entries and a toolbox section at the bottom which provide additional information.

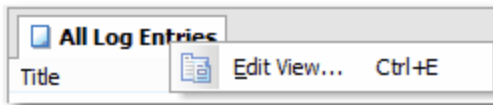


1. The All Log Entries view is the central part of the SmartInspect Redistributable Console main window and is responsible for displaying the individual logging entries as the service is running and processing files.
2. The Watches toolbox displays the state of the currently running threads, information on the total number of documents processed through the system and a collection of information about each converter that are running.
3. Other tabs available in the *Watches* section are:
 - a. Viewer - shows any attached data of the currently selected logging entry.
 - b. Details - shows the details of the log entry selected in the *All Log Entries* window.
 - c. Call Stack - show the call stack, if available, of the log entry selected in the *All Log Entries* window.

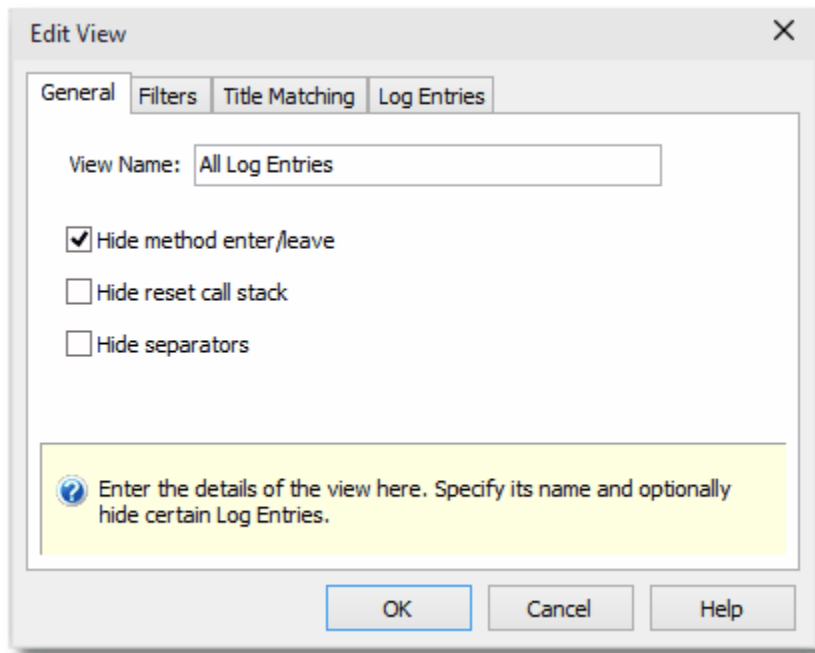
Filtering the Log Entries

You can filter the log entries so that you are only looking at entries that are reporting errors or warnings. There are many other ways to filter what is shown in the All Log Entries view; see the help file that is installed with SmartInspect Redistributable Console for more information.

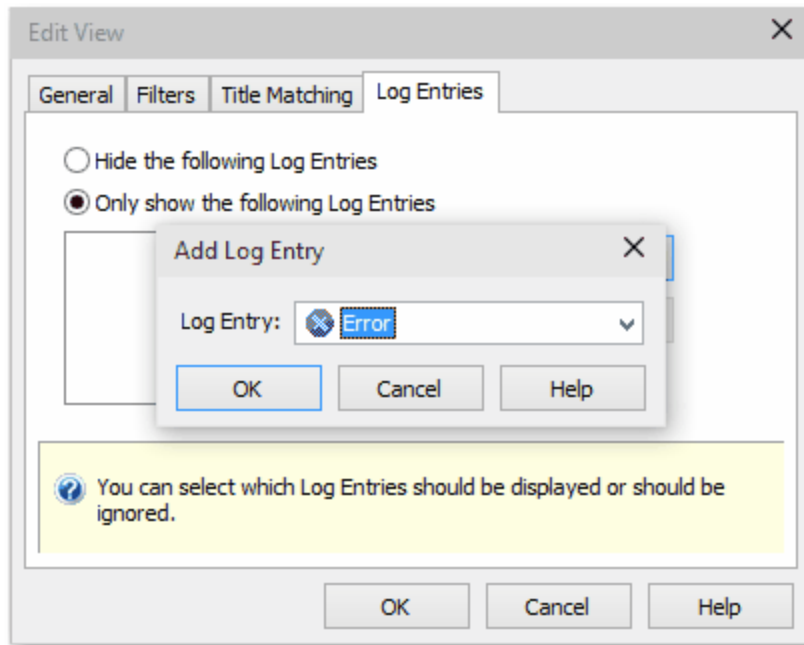
1. To apply a filter to the log entries, right-click the All Log Entries tab and select Edit View...



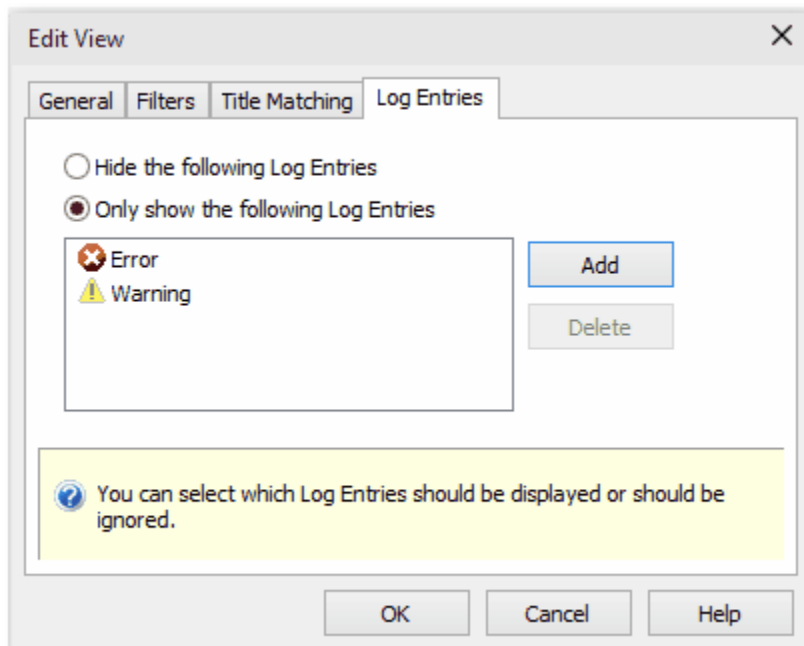
2. In the Edit View dialog, select the General tab then check the *Hide method enter/leave* option.



3. Select the Log Entries tab and then enable the *Only show the following Log Entries* option. Use the Add button to display the Add Log Entry dialog. Select the type of log entry you want to see and click OK to add it to the list.



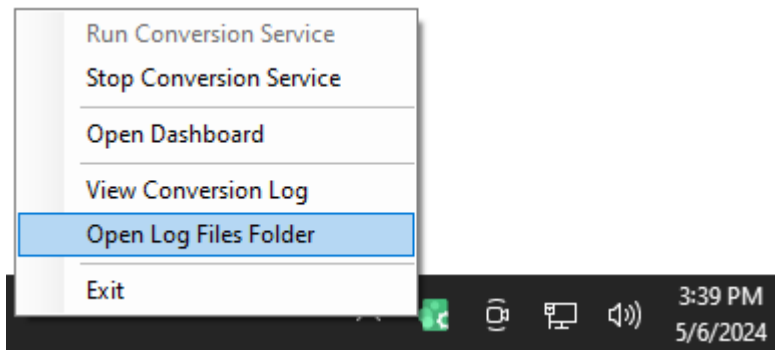
4. Repeat for all log entries (*Error*, *Warning*, etc.) that you want to see in the log entries view.



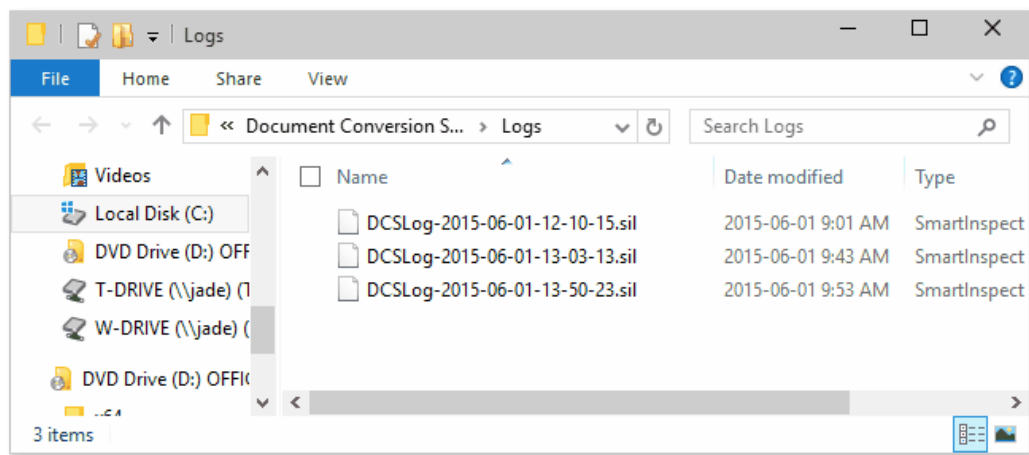
Viewing Saved Log Files

All logging information is also stored in a series of disk-based log files (up to 10 in total) that are rotated based on size and by day.

1. From the system tray icon select Open Log Files Folder.



2. The log files all start with the name *DCSLog* followed by date and time. Double-click any of these files to view that log in the logging console.



Starting and Stopping the DCS Service

Document Conversion Service was designed specifically to be run as Windows service through the PEERNET Document Conversion Service Monitor 1.0 service. The monitoring service will watch the running Document Conversion Service instance and automatically restart the conversion service if it is terminated unexpectedly.

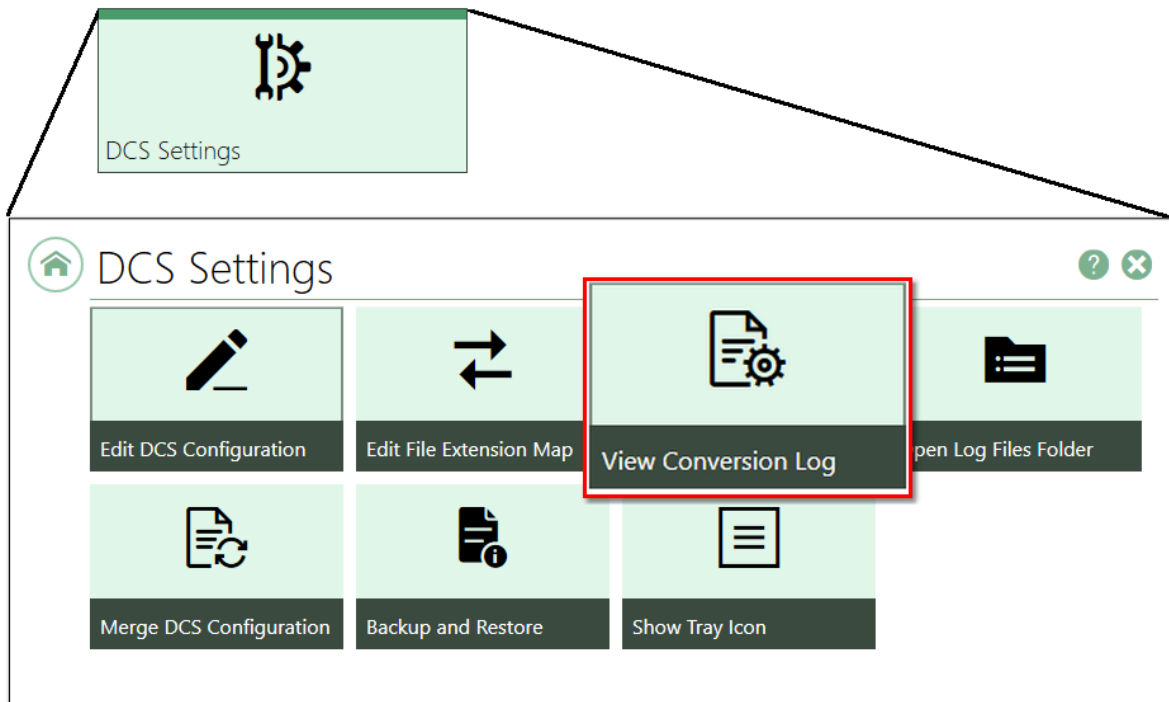
The monitoring service is installed as an *automatic* service under the user account specified during installation. This is usually the DCSAdmin account created as part of the install, or a different account chosen during the install.


When the PEERNET Document Conversion Service Monitor 1.0 service's start mode is set to *Automatic (Delayed Start)* the conversion service will run when the computer starts up, even when no one is logged into the computer. See [Changing Document Conversion Service's Startup Mode](#) for the steps needed to change PEERNET Document Conversion Service Monitor 1.0 service's start mode.

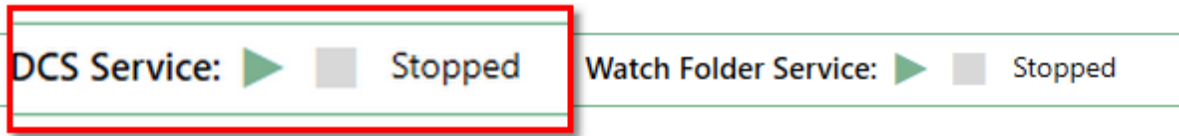
- [Starting and Stopping from the Dashboard](#)
- [Starting and Stopping from the System Tray Icon](#)
- [Using the Services Control Panel](#)

Starting and Stopping from the Dashboard

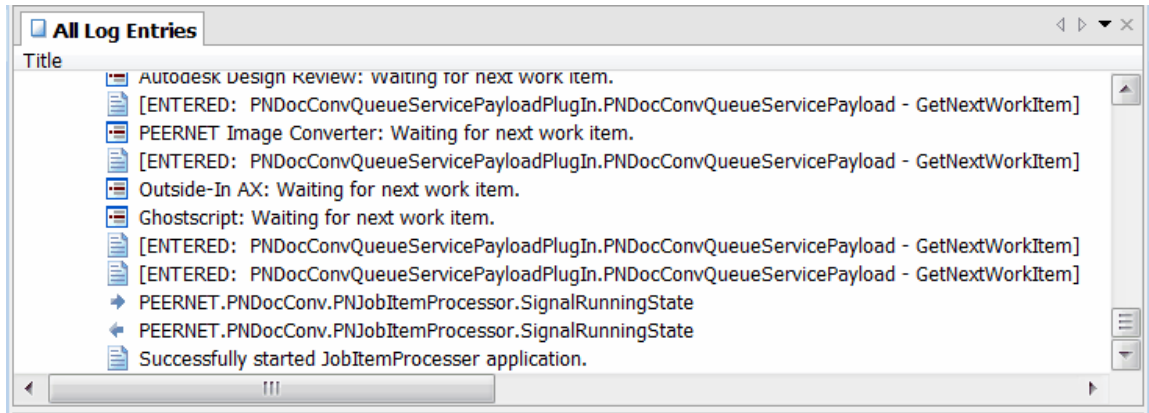
1. First, in the DCS Dashboard, go to **DCS Settings** and click on **View Conversion Log** to open [the logging console](#). This allows you to monitor the service as it starts up.



2. At the top of the DCS Dashboard you have easy access to starting and stopping Document Conversion Service. To start the service, click the green *play* icon (). The status will change to **Starting...**, and move to **Running** once the service is running.



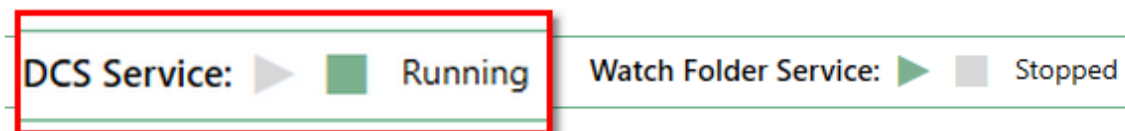
3. Logging messages detailing the status of the service will start appearing in the All Log Entries view in the logging console, if you have it open. When the *Successfully started JobItemProcessor application* message appears the service has finished initializing and is ready to start processing files.



4. The Watches toolbox displays the converters loaded. It shows the number of applications in each converter's application pool and status on documents received, processed or failed. The service auto-detects what converters can be run based on what applications are installed on the computer; if you do not see a converter listed that you expect to see check the logging messages to see why that application did not load.

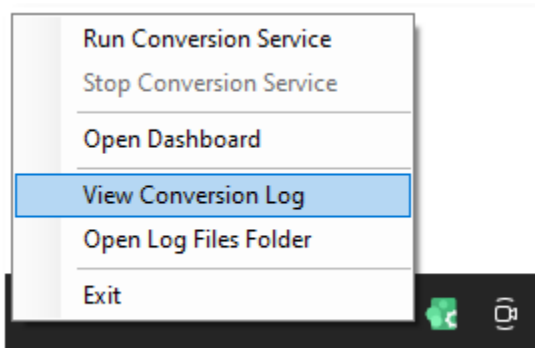
Watches	
Latest Selected	
Name	Value
Adobe Acrobat Reader - Number of successful conversions	0
Adobe Acrobat Reader - Total documents failed to start pro...	0
Adobe Acrobat Reader - Total documents received	0
Adobe Acrobat Reader - Total documents to be processed	0
Ghostscript - Number of applications in pool	3
Ghostscript - Number of failed conversions	0
Ghostscript - Number of successful conversions	0
Ghostscript - Total documents failed to start processing	0
Ghostscript - Total documents received	0
Ghostscript - Total documents to be processed	0
Internet Explorer - Number of applications in pool	3
Internet Explorer - Number of failed conversions	0
Internet Explorer - Number of successful conversions	0
Internet Explorer - Total documents failed to start processing	0
Internet Explorer - Total documents received	0
Internet Explorer - Total documents to be processed	0
Microsoft Excel - Number of applications in pool	3
Microsoft Excel - Number of failed conversions	0
Microsoft Excel - Number of successful conversions	0
Microsoft Excel - Total documents failed to start processing	0
Microsoft Excel - Total documents received	0

5. To stop the service, do click the green stop icon () at the top of the DCS Dashboard. The status will change to **Stopping...** and go to **Stopped** when the service has stopped running.

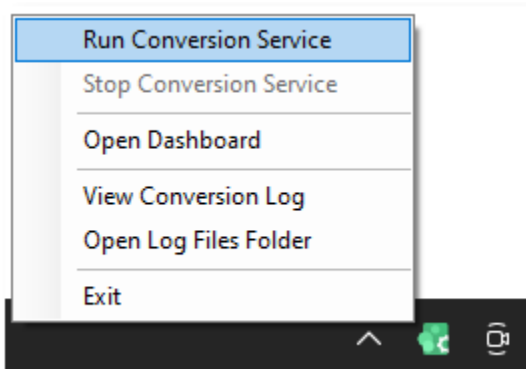


Starting and Stopping from the System Tray Icon

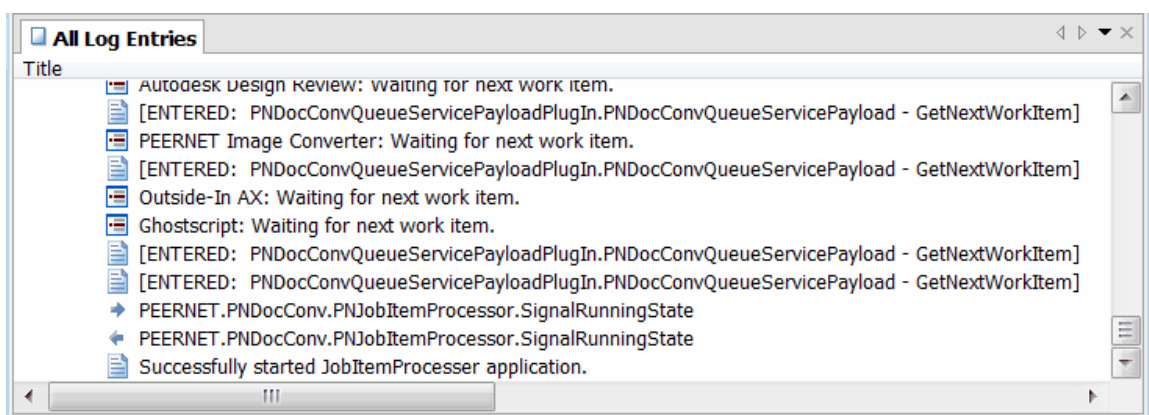
1. From the [system tray icon](#) menu select View Conversion Log to open the logging console. This allows you to monitor the service as it starts up.



2. From the system tray icon menu select Run Conversion Service.

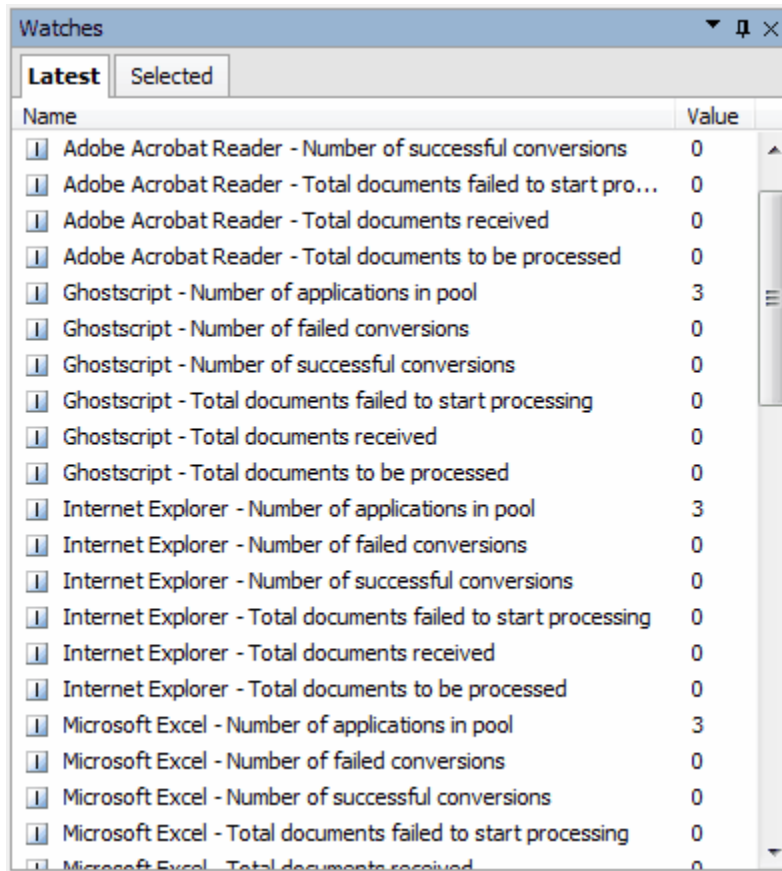


3. Logging messages detailing the status of the service will start appearing in the All Log Entries view in the logging console, if you have it open. When the *Successfully started JobItemProcessor application* message appears the service has finished initializing and is ready to start processing files.



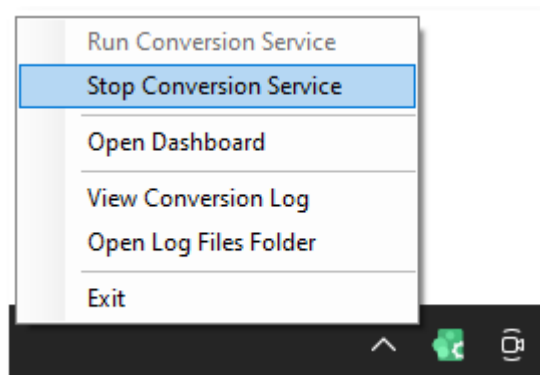
4. The Watches toolbox displays the converters loaded. It shows the number of applications in each converter's application pool and status on documents received, processed or failed. The service auto-detects what converters can be run based on what applications are installed on the

computer; if you do not see a converter listed that you expect to see check the logging messages to see why that application did not load.



Name	Value
Adobe Acrobat Reader - Number of successful conversions	0
Adobe Acrobat Reader - Total documents failed to start pro...	0
Adobe Acrobat Reader - Total documents received	0
Adobe Acrobat Reader - Total documents to be processed	0
Ghostscript - Number of applications in pool	3
Ghostscript - Number of failed conversions	0
Ghostscript - Number of successful conversions	0
Ghostscript - Total documents failed to start processing	0
Ghostscript - Total documents received	0
Ghostscript - Total documents to be processed	0
Internet Explorer - Number of applications in pool	3
Internet Explorer - Number of failed conversions	0
Internet Explorer - Number of successful conversions	0
Internet Explorer - Total documents failed to start processing	0
Internet Explorer - Total documents received	0
Internet Explorer - Total documents to be processed	0
Microsoft Excel - Number of applications in pool	3
Microsoft Excel - Number of failed conversions	0
Microsoft Excel - Number of successful conversions	0
Microsoft Excel - Total documents failed to start processing	0
Microsoft Excel - Total documents received	0

5. Select Stop Conversion Service from the [system tray icon](#) context menu to stop Document Conversion Service. This menu item is disabled if Document Conversion Service is not running.



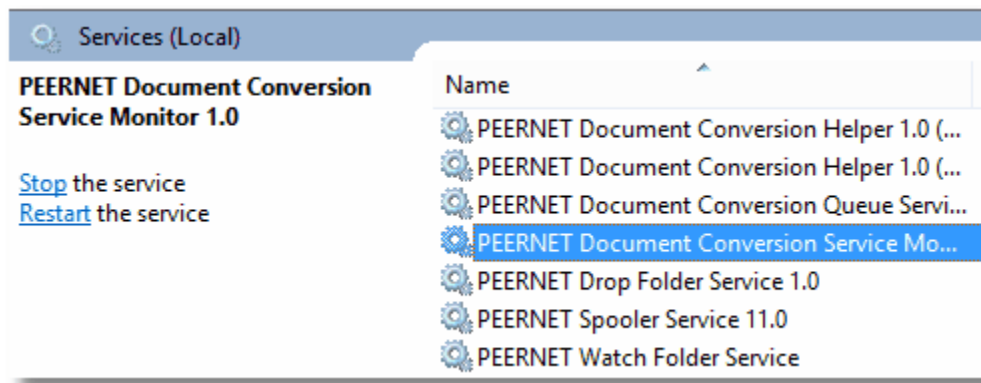
Using the Services Control Panel

The PEERNET Document Conversion Service Monitor 1.0 service can also be accessed through the Services control panel applet. From the Services control panel you can do the following:

- start and stop the monitoring service
- change the service Log On account
- change the service start up type

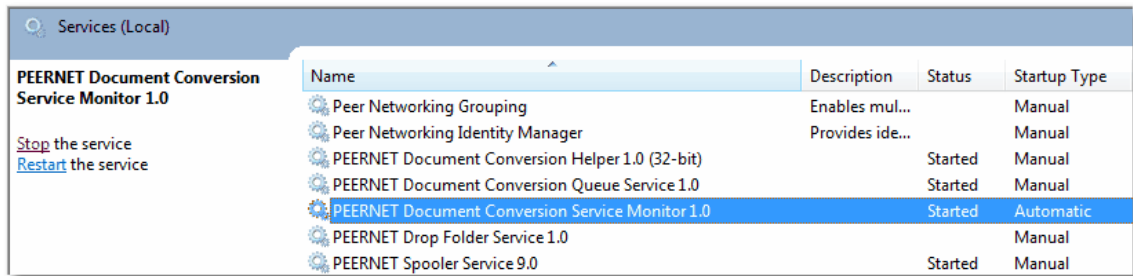
Start the Service

1. Go to Start - Control Panel - System and Security - Administrative Tools - Services (or type "Services" into the search field on the Start menu).
2. In the Services control panel applet, locate the service PEERNET Document Conversion Service Monitor 1.0.
3. Select Start from left hand side.



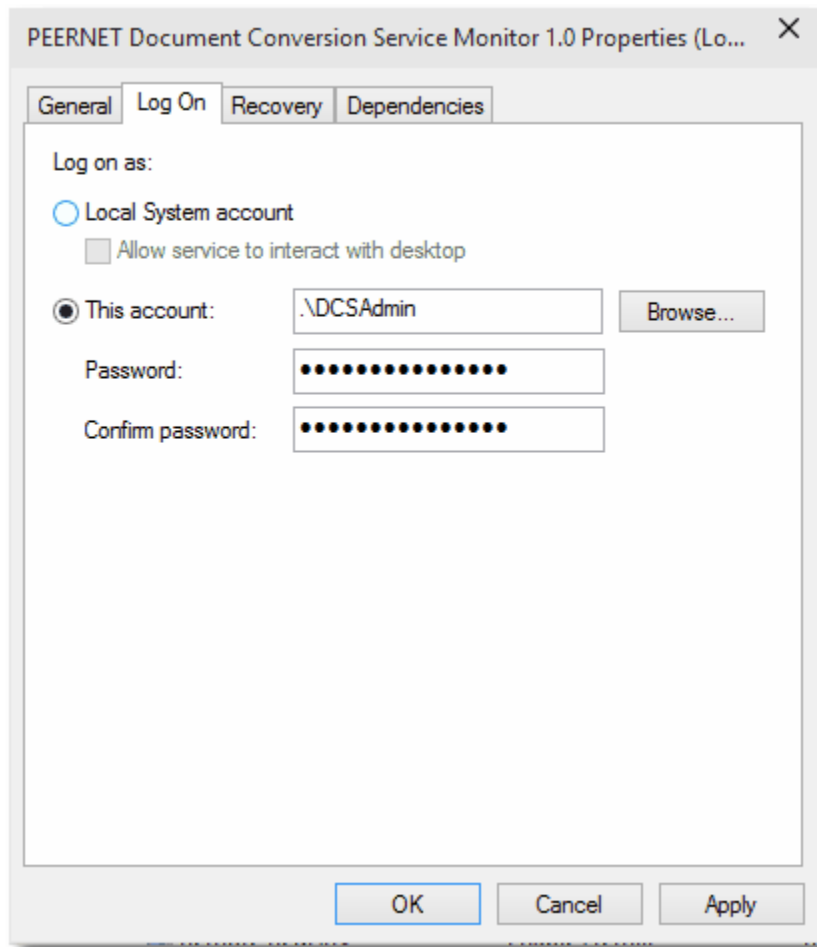
Stop the Service

1. Go to Start - Control Panel - System and Security - Administrative Tools - Services (or type "Services" into the search field on the **Start** menu).
2. In the Services control panel applet, locate the service PEERNET Document Conversion Service Monitor 1.0.
3. Select Stop from left hand side.



Changing the Service Log On Account

1. Go to Start - Control Panel - System and Security - Administrative Tools - Services (or type "Services" into the search field on the Start menu).
2. In the Services control panel applet, locate the service PEERNET Document Conversion Service Monitor 1.0.
3. Double-click the service to open the **Properties** dialog.
4. On the Log On tab, change the Log On account to the desired account.



Changing the Service Startup Mode

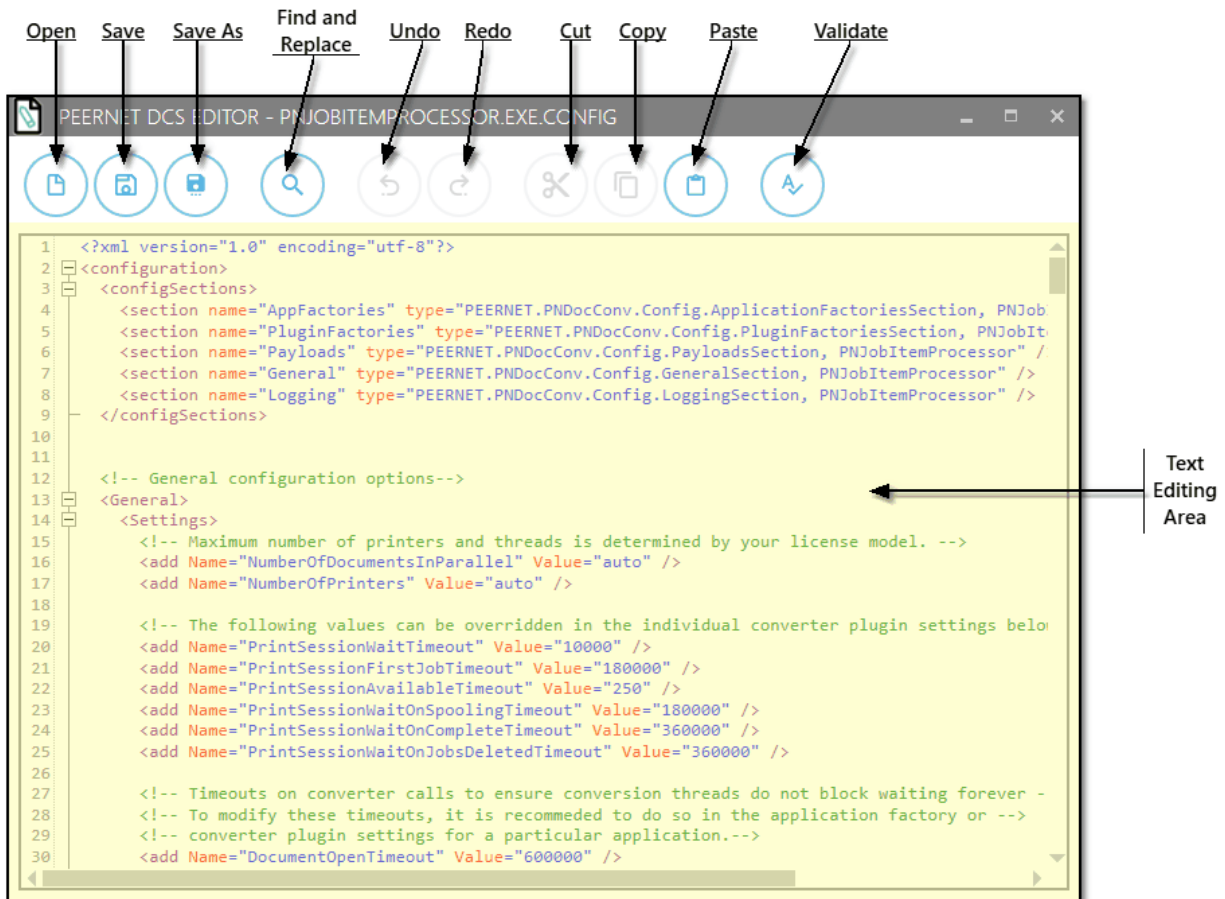
Document Conversion Service is initially installed as an automatic (delayed start) service. The section [Changing Document Conversion Service's Startup Mode](#) under [Advanced Configuration](#) has complete instructions on changing the service startup type.

Editing Files with the DCS Editor

Starting with Document Conversion Service 3.0.017, a DCS Editor is included to allow for easy editing of both DCS configuration files and Watch Folder configuration files, the file extension converter mapping file and conversion profiles. This editor is used both from the [DCS Dashboard](#) and from any program shortcuts from the Windows Start menu.

The editor offers colored syntax highlighting, error checking for syntax errors when files are loaded or saved, and basic validation when editing the configuration files or conversion profiles.

The start menu shortcuts for editing the DCS and Watch Folder configuration files, and conversion files now use this tool instead of Notepad.

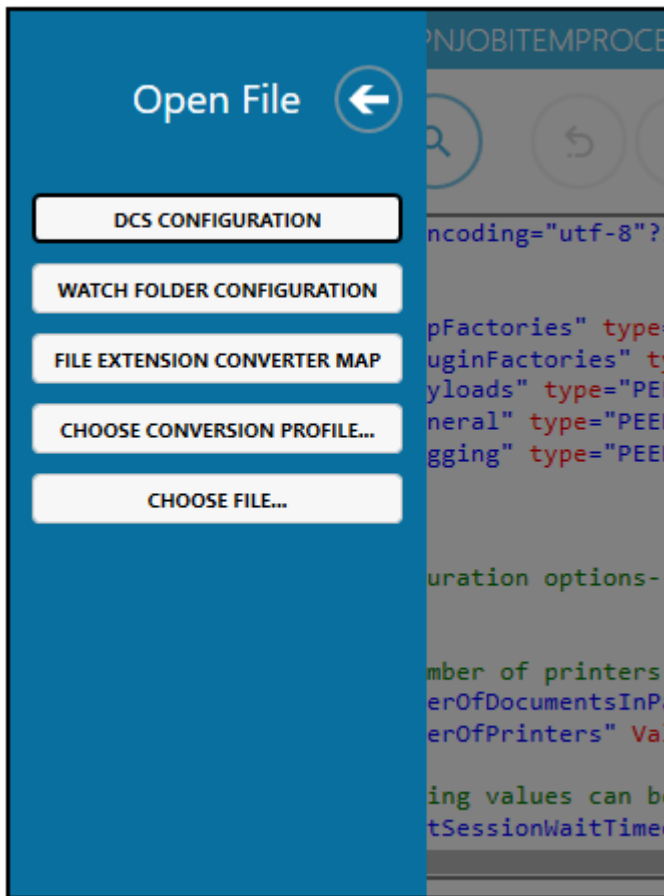


Opening Files



The editor is tailored for interacting with the DCS configuration file, the Watch Folder configuration file, the file extension converter mapping file, and the conversion profiles. The Open button presents a flyout on the left allowing you to choose editing one of these files. Clicking on the left arrow icon or away from the flyout will dismiss it.

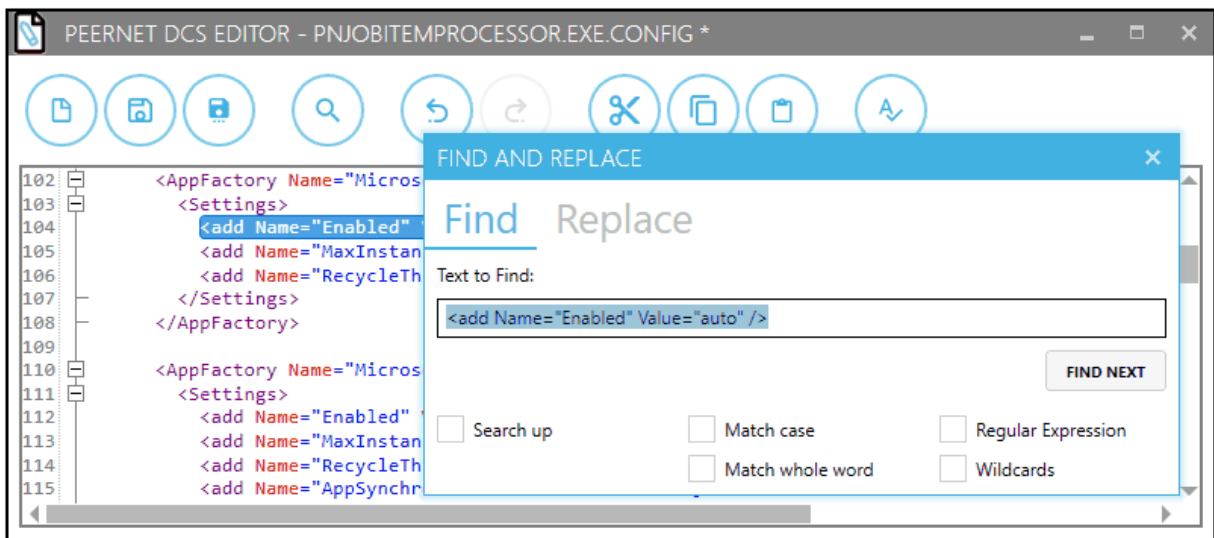
Files are validated for syntax errors upon opening and any errors are displays as shown in [Validating the Files](#) below.



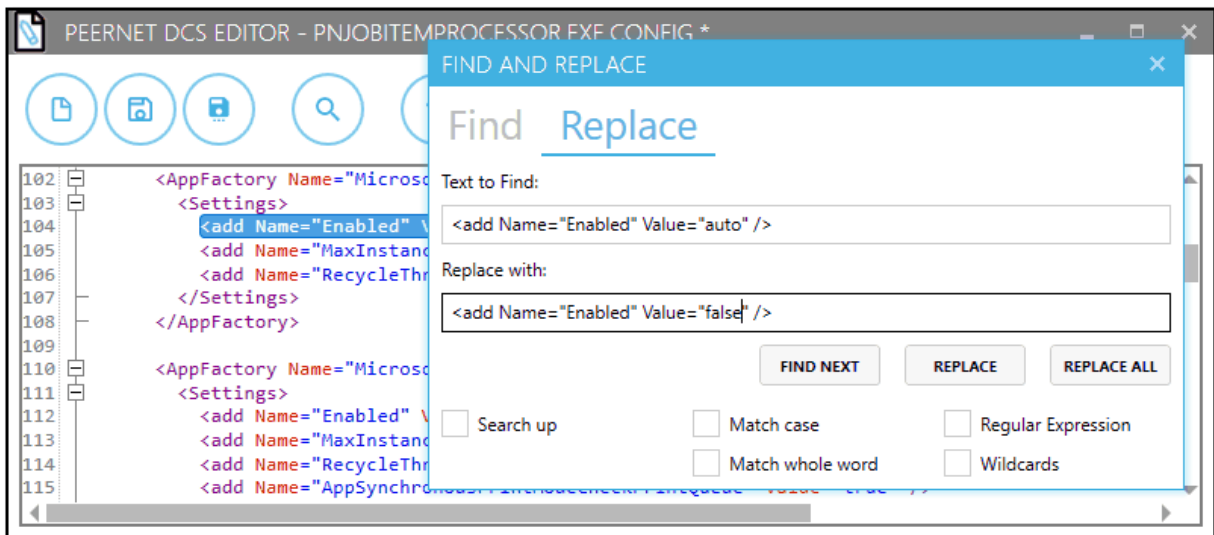
Find and Replace



One of the most common editing tasks is to enable or disable individual document converters in the DCS configuration file, based on the file types you need to convert. The editor includes a Find and Replace Tool that allows for searching throughout the files. Selecting a section of text before opening the find and replace window will fill in the search text with the selected text on the **Find** tab.



The **Replace** tab allows you to do a global find and replace of all occurrences of the string in the document.



Validating the Files



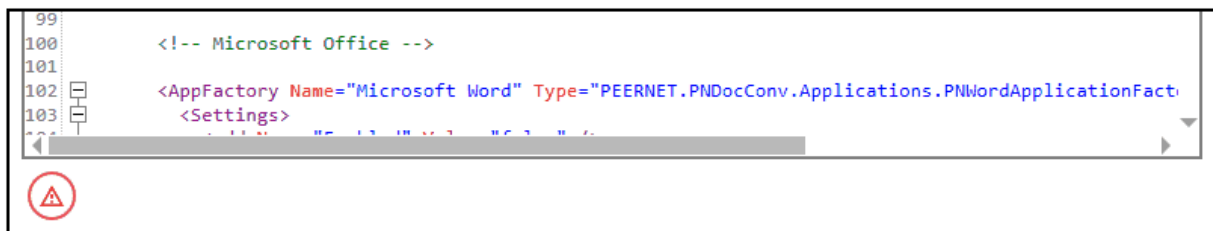
The editor can validate the file for syntax errors as well as checking that the configuration files and conversion profiles contain their required respective sections. The color syntax highlighting also provides visual cues if the file syntax is incorrect.

You can validate a file at any time using the **Validate** button. Files are also validated upon opening and saving.

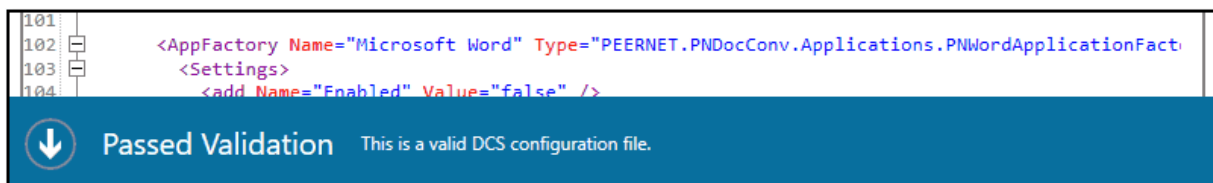


If an error is encountered, the error information is shown in an error flyout at the bottom of the screen, and if possible, the file is scrolled to the line number that contains the error. The error flyout can be dismissed by clicking above it, or on the down arrow icon on the left side of the flyout.

You can display the error message again by clicking on the warning symbol in the lower left corner.



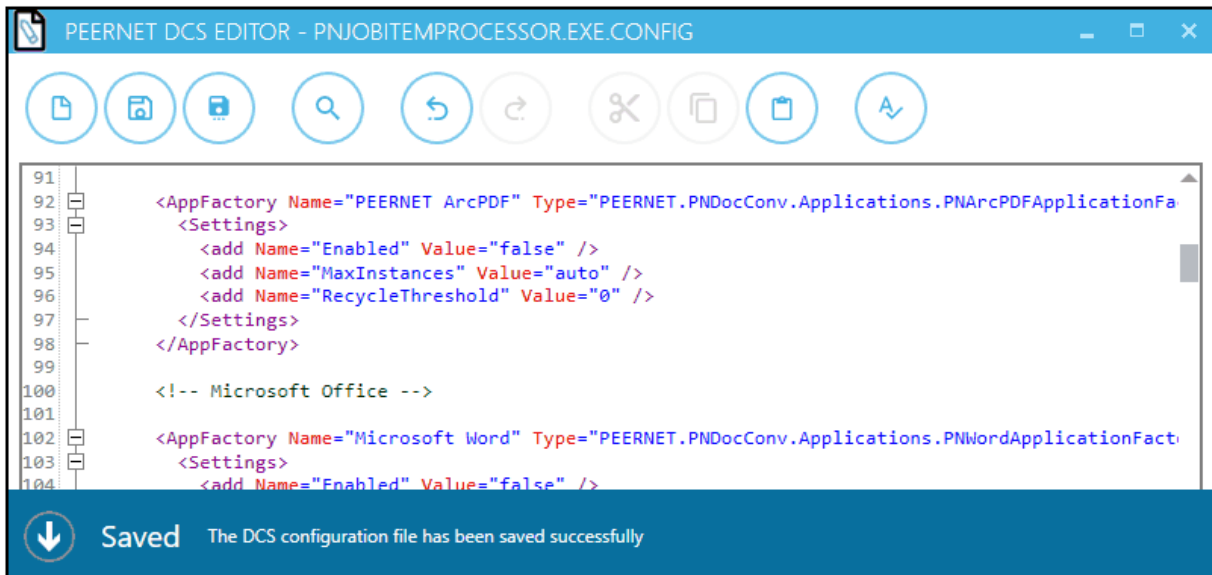
If the file contains valid syntax, the following success flyout is displayed and automatically dismissed.



Saving the Files

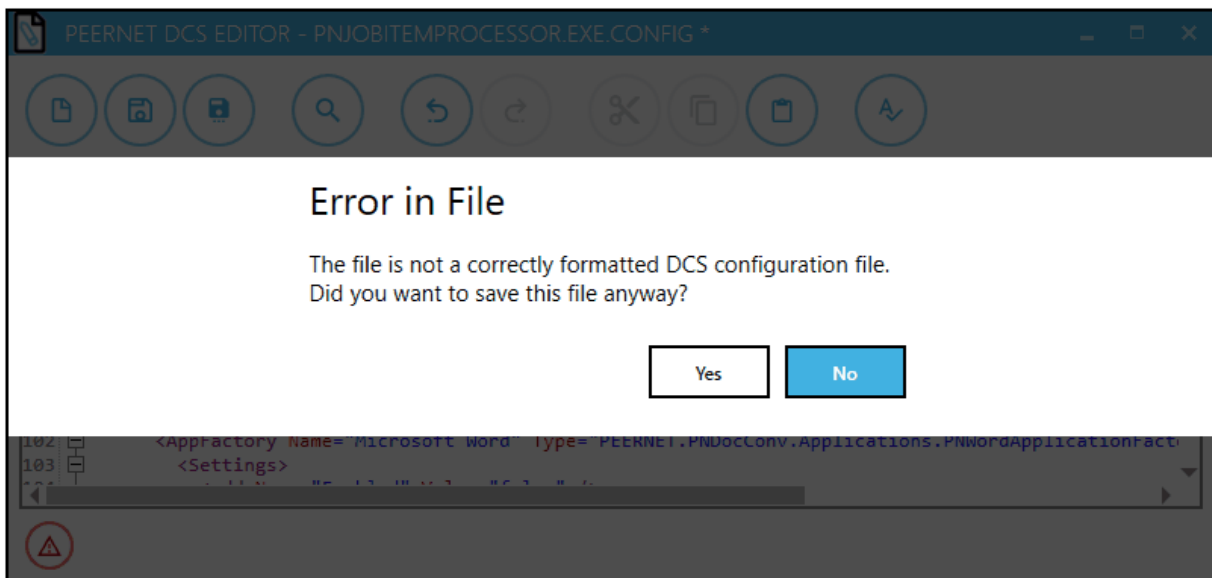


When saving, the files are also validated for syntax and correct sections. When the file is correct, it is saved and the following saved flyout message is displayed and automatically dismissed.



When the file is incorrect a warning message is displayed. Selecting **No** will return to the editor with the error message displayed in the error flyout at the bottom of the editor allowing you to fix the error.

Configuration files and conversion profiles with syntax errors will cause DCS, Watch Folder and the conversion utilities to not work correctly.



Configuring Third-Party Applications Used by Document Conversion Service

Document Conversion Service is designed to automatically configure any third-party applications for use within the conversion service. This includes running Document Conversion Service under new accounts, like the DCSAdmin account, that have never been logged into. If you chose to use a different account when installing, the same automatic configuration will take place.

Some of the native applications used by the converters do require that certain components need to be installed, or that a change is made to the Document Conversion Service configuration to use the desired version. Any special steps needed for an application are outlined below.

- [Configure Adobe Reader for Foreign Languages](#)
- [Flash Player for Adobe Reader](#)
- [AutoCAD Design Review and TrueView](#)
- [Setting the Ghostscript Version](#)
- [Microsoft Outlook](#)
- [Outside-In AX \(uses Oracle Outside In Technology\)](#)
- [Animated Images and Movies with FFmpeg.exe](#)
- [Windows Imaging Component \(WIC\) Add-Ons and Extensions](#)
- [Internet Explorer](#)

In most cases the only thing you have to do is install and license (activate) the appropriate third-party application used by the converter before running Document Conversion Service.



Third-Party Application Licensing

Any third-party applications that require activation, such as Microsoft Office, must be activated on the computer where Document Conversion Service is running.

Adobe Reader for Foreign Languages

If you think you will be converting PDF documents that will contain foreign languages, such as Hebrew, Arabic, Thai and others, you may need to download the Adobe Reader Font Pack for your version of Adobe Reader.

The font pack for the Adobe Reader DC can be found here: [32-bit Font Pack and Spelling Dictionary for Acrobat Reader DC](#)

You can also follow the instructions shown in Adobe Reader when opening the PDF file. The reader application will direct you where to download and install it's latest supported font packs.

Configuring Flash for Adobe Reader

Support for Flash Player in Adobe was discontinued after December 31, 2020 and the latest Adobe Reader DC now blocks Flash content from running in Flash Player beginning January 12, 2021.

If you think you will be processing PDF Portfolios or other PDF files that contain Flash content, you will need to contact the creator of those PDF files for updated versions.

Autodesk Design Review

With Autodesk Design Review installed you can convert DWF files. If you need to convert DWG files as well, DWG TrueView will also need to be installed.

Document Conversion Service will work with the following combinations of Autodesk Design Review and DWG TrueView installed:

- [Autodesk Design Review 2012](#) and [DWG TrueView 2012](#)
- [Autodesk Design Review 2013](#) and [DWG TrueView 2013](#)
- Starting with version 3.0.014, [Autodesk Design Review 2018](#) and [DWG TrueView 2018](#)
 - While Autodesk Design Review 2018 will work with DWG TrueView [2012-2017](#), we recommend using DWG TrueView 2018 when possible.

Prior to Document Conversion Service 3.0.013 you will need to turn off the unresolved references and missing SHX (shape) files prompts that are shown by DWG TrueView as outlined in [Turning off Prompts in DWG TrueView](#). Starting with 3.0.013, an updated *FixedProfile.aws* profile set to ignore unresolved references and missing SHX files is copied into the application data section when Document Conversion Service launches DWG TrueView. Any existing *FixedProfile.aws* is backed up as *PNDCSBackup.FixedProfile.aws*.

If you need to create vector Adobe PDF files, you will also need to [Add Printer Permissions](#) to the Microsoft XPS Document Writer printer for the *Everyone* account. Alternatively you can add the permissions for just the account that Document Conversion Service is running under - in most cases this is the *DCSAdmin* created as part of the install.

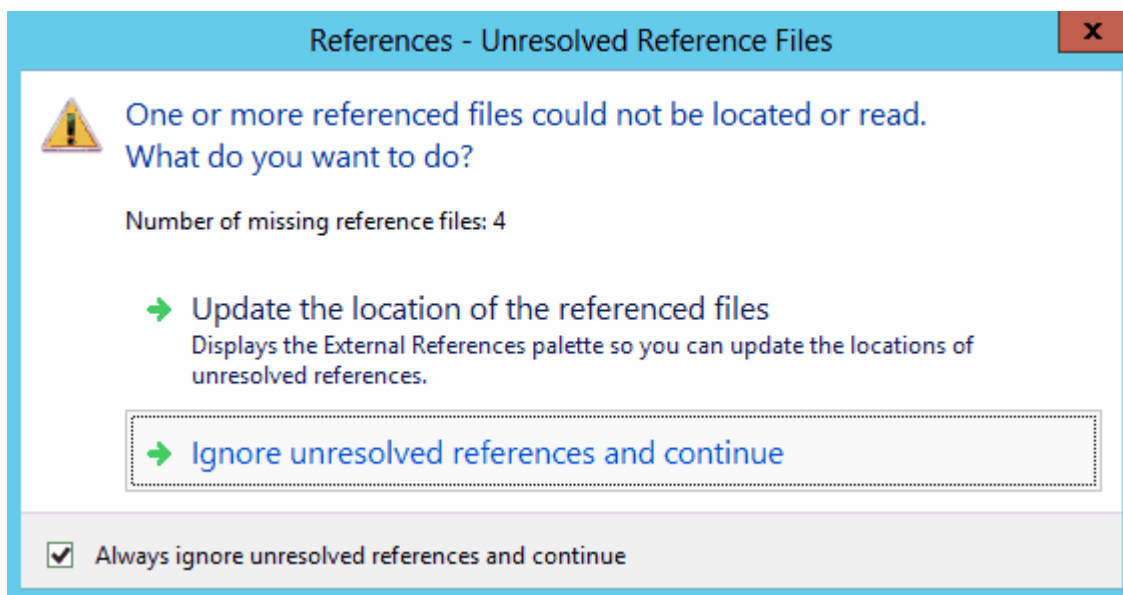
Turning off Prompts in DWG TrueView

These steps only apply if you are running Document Conversion Service prior to version 3.0.013.

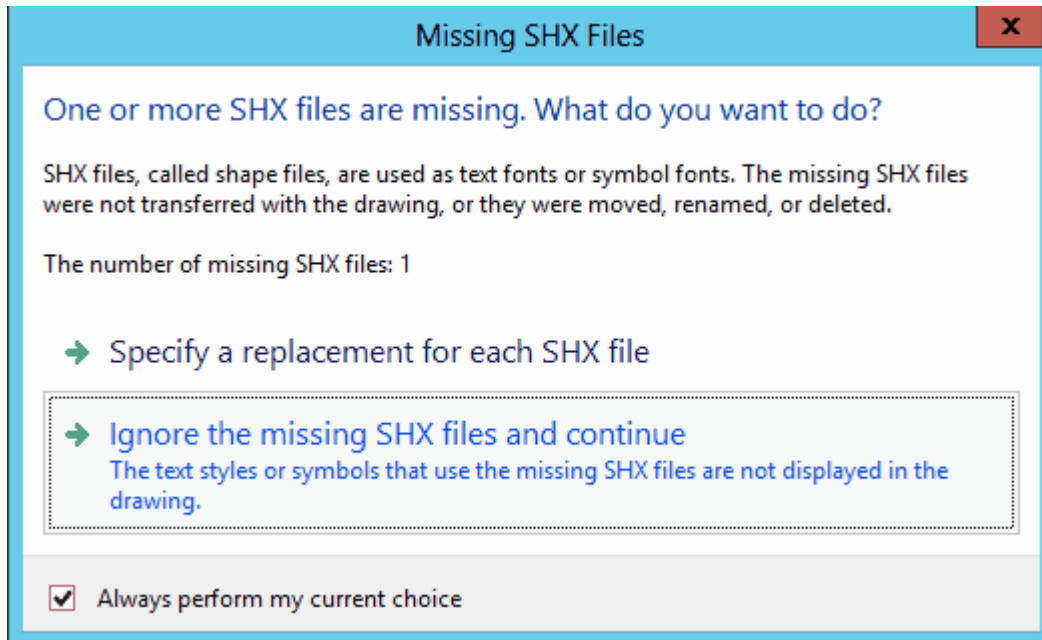
If a single DWG is processed that references missing files and/or shapes, DWG TrueView will prompt as to how to handle these missing elements. To manually prevent these dialogs from prompting and halting the conversion process the option to update or ignore unresolved references and shapes must be turned off.

These steps **MUST** be performed under the same user account the Document Conversion Service runs under. This is usually DCSAdmin.

1. First, copy a DWG file that references other DWG files into its own location and open this file from that location.
2. Open the moved DWG. If unresolved references are not already set to ignore you will see the **References - Unresolved References Files** dialog box. In this dialog:
 - a. Check the *Always ignore unresolved references and continue* option at the bottom of the dialog.
 - b. Select the *Ignore unresolved references and continue* option.



3. Next, if you have a DWG file that uses missing SHX files, open that file.
 - a. Check the *Always perform my current choice* option at the bottom of the dialog.
 - b. Select the *Ignore the missing SHX files and continue* option.



4. Close DWG TrueView to save the changes.

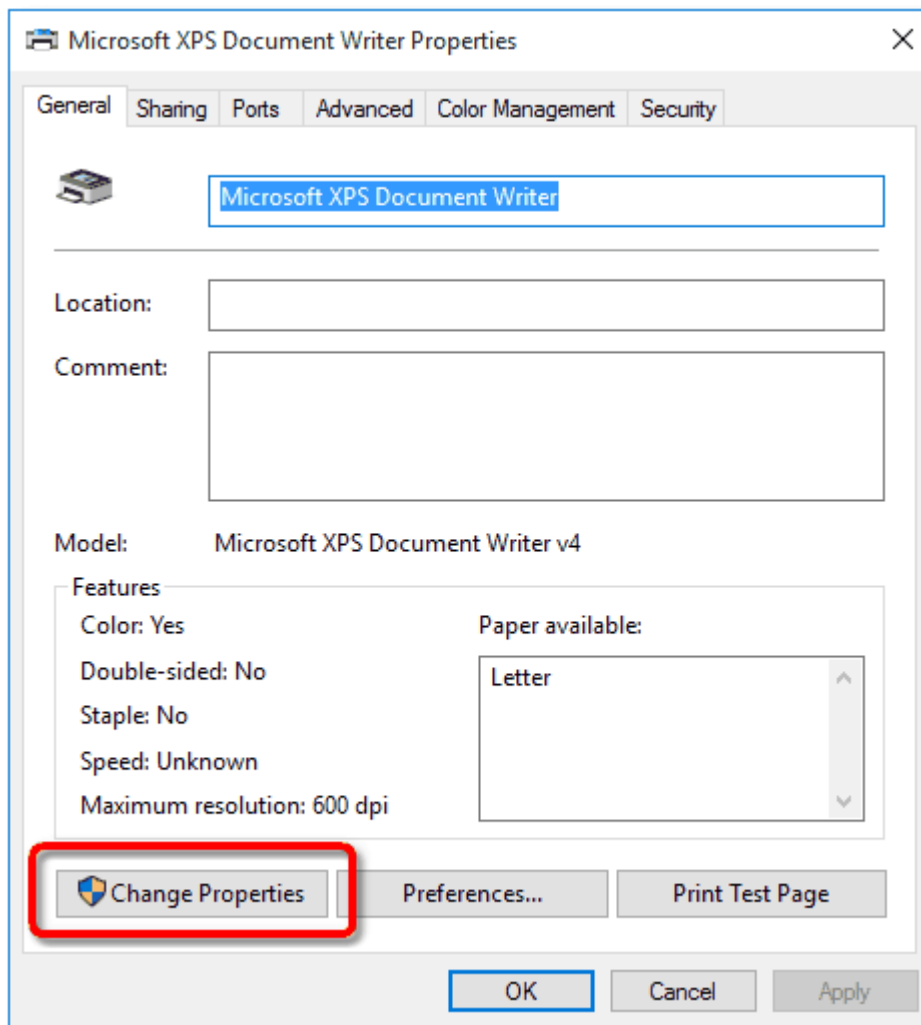
Adding Printer Permissions to Microsoft XPS Document Writer

This is only needed if you are creating vector Adobe PDF files using the profile *Adobe PDF Multipaged*, or the settings listed within.

The instructions below show how to add this permission to the Microsoft XPS Document Writer for the **Everyone** account. You can instead add these permissions for the account that Document Conversion Service is running under; this is often the DCSAdmin account. If you used a different account when installing, add these permissions for that account instead.

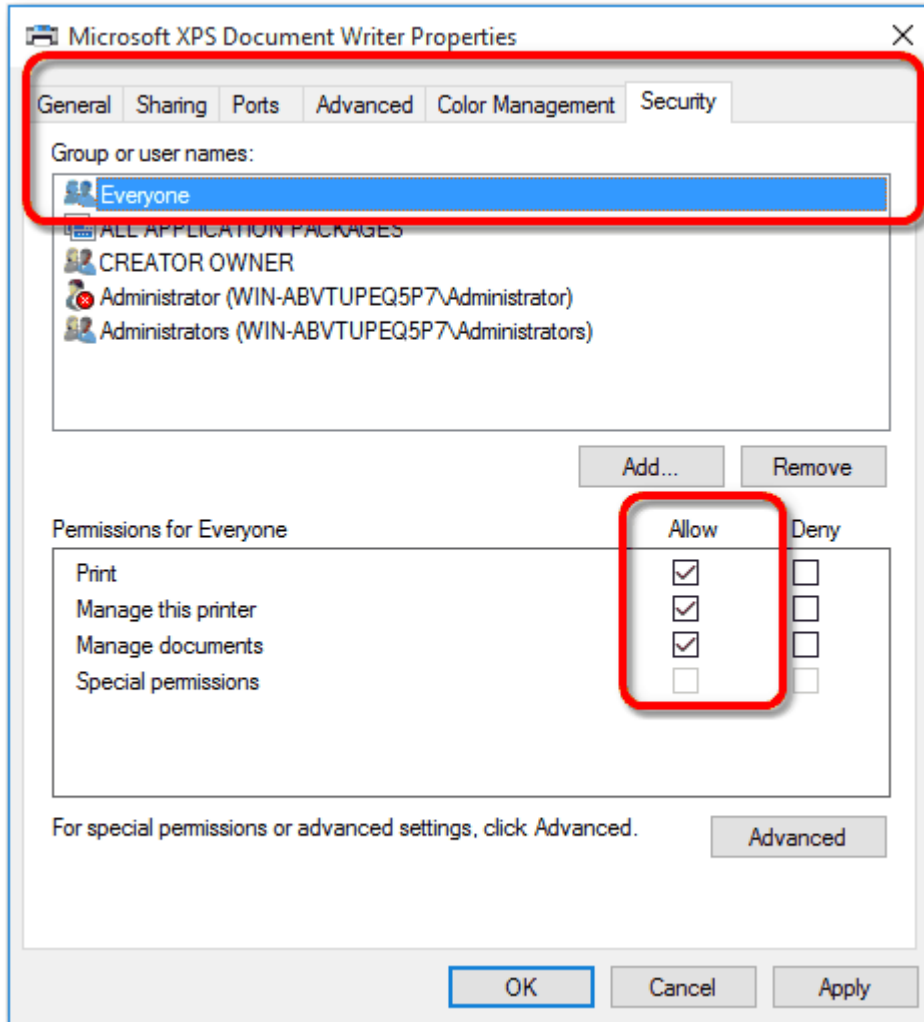
You will need to have Administrative permissions to make these changes.

1. Open the *Devices and Printers* folder by typing *Printers* into the Search field in the Start menu.
2. Right-click on the Microsoft XPS Document Writer printer and select *Printer Properties* from the context menu.
3. On the **General** tab, Click the *Change Properties* button in the lower left.



4. Click on the Security tab, select the *Everyone* account, then make sure that the permissions *Print*, *Manage this printer*, and *Manage documents* are checked.

If you only want to add the permissions for the DCSAdmin account or your own custom account, click the *Add...* button to show the dialog listing all available users. Select DCSAdmin or your custom account to add that user to the list. Once added, select that user in the list and make sure that the permissions *Print*, *Manage this printer*, and *Manage documents* are checked for that user.



Setting the Ghostscript Version

Beginning with version 3.0.014, Document Conversion Service will auto-detect any 32-bit installed versions of Ghostscript and will automatically use the highest version that it finds.

If no 32-bit installs of Ghostscript are found, Document Conversion Service will then use the bundled Ghostscript 7.0.7 included in the Document Conversion Service install. 64-bit installations of Ghostscript are not supported.

As 7.07 is an older version of Ghostscript, there are limitations when converting newer Postscript files and PDF files. If you encounter any issues converting Postscript or PDF files, we recommend upgrading to the latest version of Ghostscript.

Manually Setting the Path to Ghostscript

To use a specific version of Ghostscript, the Ghostscript converter needs to know the path to the installed version of Ghostscript you want so that it can load the needed components. This path will vary depending on where you installed Ghostscript and the version you installed.

The Ghostscript converter is configured in the application configuration file. The configuration file is an XML file that can be edited directly using the DCS Editor, in an XML editor or any text editor such as Notepad or WordPad.



Warning - Ghostscript 9.05 (32-bit) or Higher

Document Conversion Service was tested against various versions of Ghostscript and it is recommended that version 9.05 or later be used. Earlier versions were found to have handle leak issues.

Do not install the 64-bit version available in the latest releases of Ghostscript.

Document Conversion Service only works with the 32-bit version.

Opening the Configuration File

Go to **DCS Dashboard** - DCS-Settings - Edit DCS Configuration to edit the configuration file using a visual GUI.

You can also go to Start - All Programs - PEERNET Document Conversion Service 3.0 - Edit DCS Configuration File to edit the configuration file using the DCS Editor.

The configuration file can also be opened in any XML editor and can be found here:

Configuration file location:

C:\Program Files\Document Conversion Service 3.0\Core\PNJobItemProcessor.exe.config

The paths to your version of Ghostscript should follow the format below. Replace N.NN with your version of Ghostscript.

Standard Ghostscript DLL Path:

C:\Program Files (x86)\gs\gsN.NN\bin\gsdll32.dll

If you have both Ghostscript and GSView installed, you can find the GS_DLL and GS_LIB paths by opening GSView and going to Options - Advanced Configure from the menu. You can also search for the keys GS_DLL and GS_LIB in the HKEY_LOCAL_MACHINE hive of your registry by using the Registry Editor (regedit.exe).

In the configuration section for Ghostscript, uncomment the GS_DLL and GS_LIB settings and set the path to the version of Ghostscript you want to use.



Configuration Section for Ghostscript 10.03.1 on a 64-bit machine

```
<AppFactories>
  <Factories>
    <AppFactory Name="Ghostscript"
      Type="PEERNET.PNDocConv.Applications.PNGhostscriptApplicationFactory"
      Assembly="PNGhostscriptApplicationFactory">
      <Settings>
        <add Name="Enabled" Value="auto"/>
        <add Name="MaxInstances" Value="auto"/>
        <add Name="RecycleThreshold" Value="0"/>
        <add Name="GS_DLL" Value="C:\Program Files (x86)\gs\gs10.03.1\bin\gsdll32.dll" />
        <add Name="GS_LIB" Value="C:\Program Files (x86)\gs\gs10.03.1\bin;
          C:\Program Files (x86)\gs\gs10.03.1\lib" />
      </Settings>
    </AppFactory>
    ...
  </Factories>
  <Settings>
    <!-- Global factory settings -->
    <add Name="MaxInstances" Value="5"/>
    <add Name="RecycleThreshold" Value="100"/>
  </Settings>
</AppFactories>
```

Vector PDF with Office 2007

If you are creating vector (searchable) PDF files and have Office 2007 installed, you will need to download and install the following add-in from Microsoft.

- [2007 Microsoft Office Add-in: Microsoft Save as PDF or XPS](#)

Microsoft Outlook

In order to use Outlook to convert e-mail messages files safely and securely, a private e-mail account is recommended. A private e-mail account cannot send or receive e-mail but does allow Outlook to open and print Message Files (*.msg) and Templates (*.oft). Starting with Document Conversion Service 3.0.029, support for EML mail messages has been added as well.

If you have Email (EML) or Message (MSG) files with attachments, see [The Watch Folder Service Sample](#) for the ability to process these files and also extract and convert any attachments in the files. You will still need to follow the steps below before processing.

If you are using the DCSAdmin account, or have created a new account, Outlook will be automatically configured to use a private e-mail account when Document Conversion Service is first run. If you are running the conversion service under a user account where Outlook that is already configured to use an Exchange server or other mail account no further configuration is necessary.

There are certain conditions in which Document Conversion Service will not launch Outlook. If these conditions are detected, an error message is displayed in the [logging console](#) as well as in the Application log in the Event Viewer.

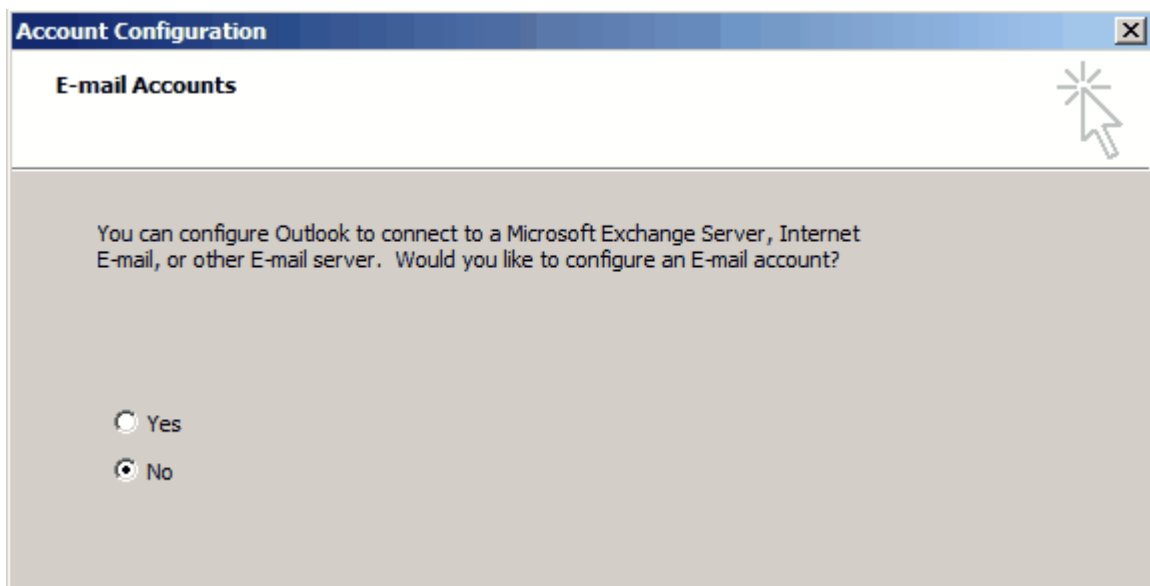
- Outlook has never been run or configured under the user account the Document Conversion Service is running under.
- Printing attachments is enabled. Printing attachments must be disabled for the Microsoft Outlook converter to operate properly. To disable printing attachments do the following:
 - select an e-mail from the list of email (do not open the email)
 - go to File - Print and in the Print dialog uncheck the *Print attached files* checkbox
 - close Outlook to save the changes

Setting up a Private E-Mail Account

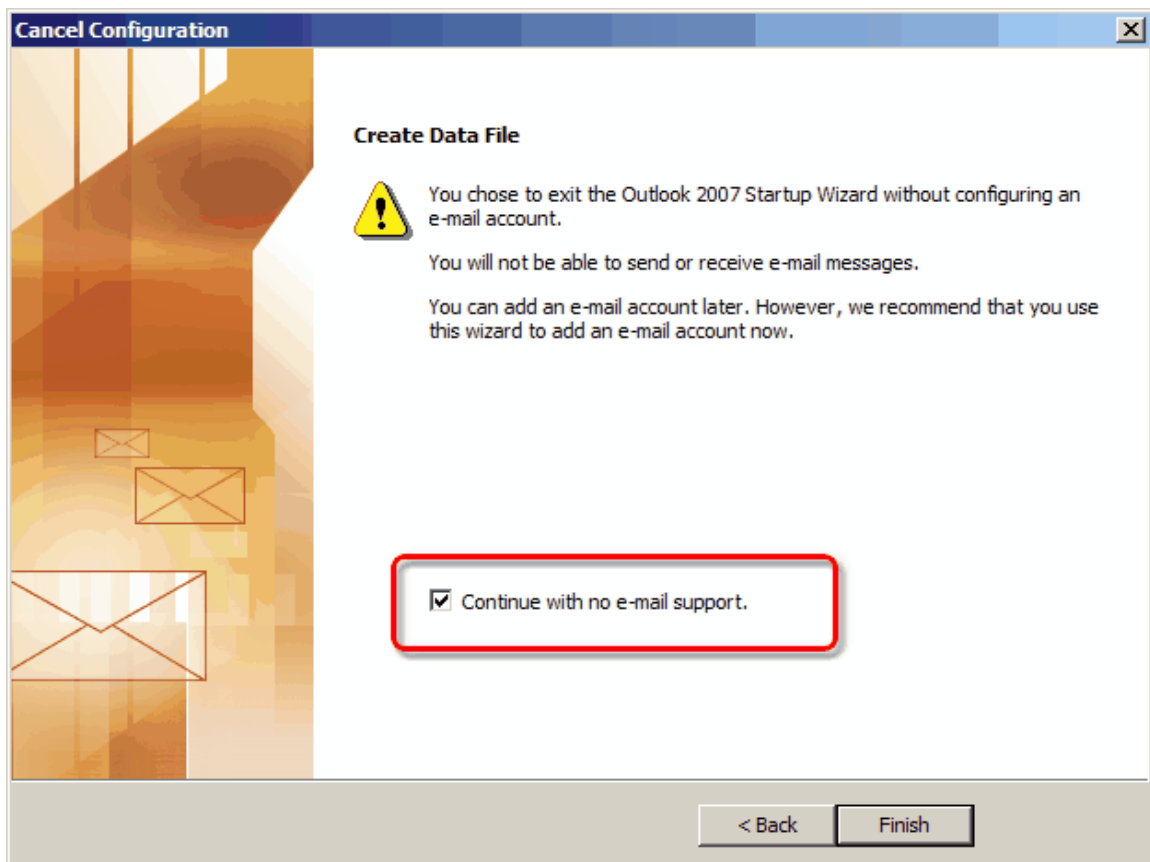
When setting up a private e-mail account on a computer that is not attached to your company domain, this is just a case of selecting "No" when Outlook prompts you to create an e-mail account. If the computer is attached to the domain, and you are running an Exchange server, you will need to configure Outlook to not use the Exchange server.

Microsoft Outlook 2007, 2010 and 2013

- 1. This set up process needs to be done under the user account that the Document Conversion Service will be running under. This is the user account you specified when installing the product, usually DCSAdmin, or a custom account you chose during installation.**
- 2.** When Outlook 2007 or Outlook 2010 or Outlook 2013 are launched for the first time you will be prompted to configure an e-mail account. Choose No on this dialog.



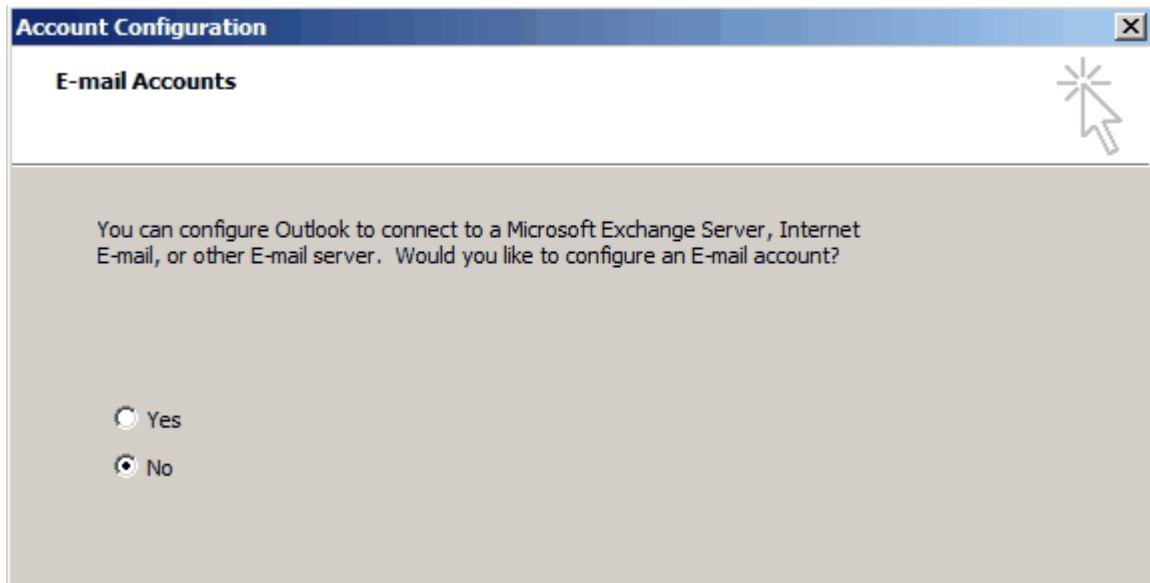
3. On the Cancel Configuration dialog, check the "*Continue with no e-mail support*" option, then select Finish.



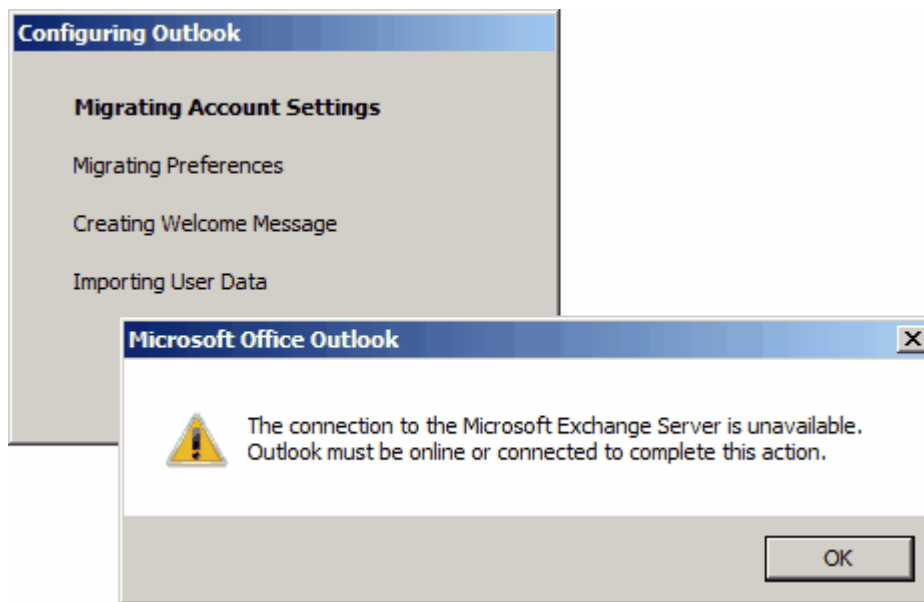
4. Exit the application to save the mailbox settings.
5. Log out and log back in as the Document Conversion Service user.
6. Open Outlook 2007 (or Outlook 2010/2013) again. In some scenarios there can be one last prompt to confirm that you do not want to connect to an Exchange server.
7. Close Outlook and log off the user account.
8. Outlook 2007 (or Outlook 2010/2013) is now configured to run with Document Conversion Service.

Outlook 2003

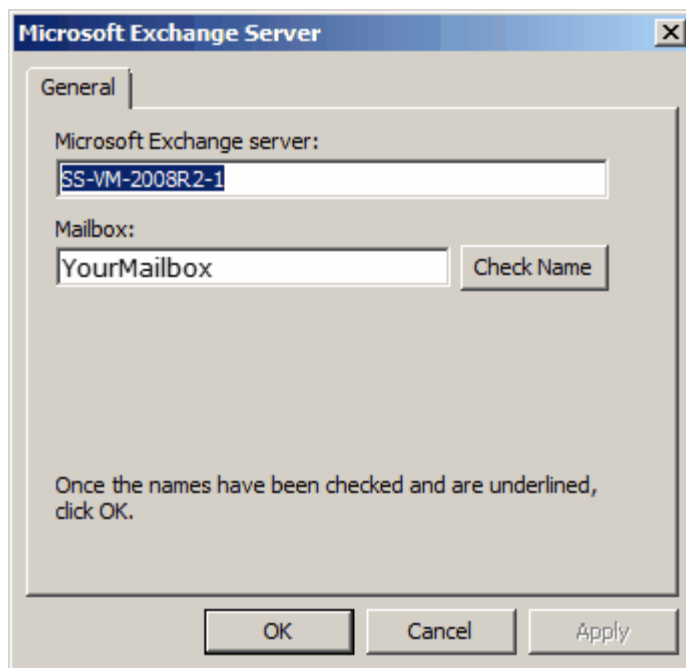
1. **This set up process needs to be done under the user account that the Document Conversion Service will be running under. This is the user account you specified when installing the product, usually DCSAdmin, or a custom account you chose during installation.**
2. When Outlook 2003 is launched for the first time you are prompted to configure an e-mail account. Choose No on this dialog.



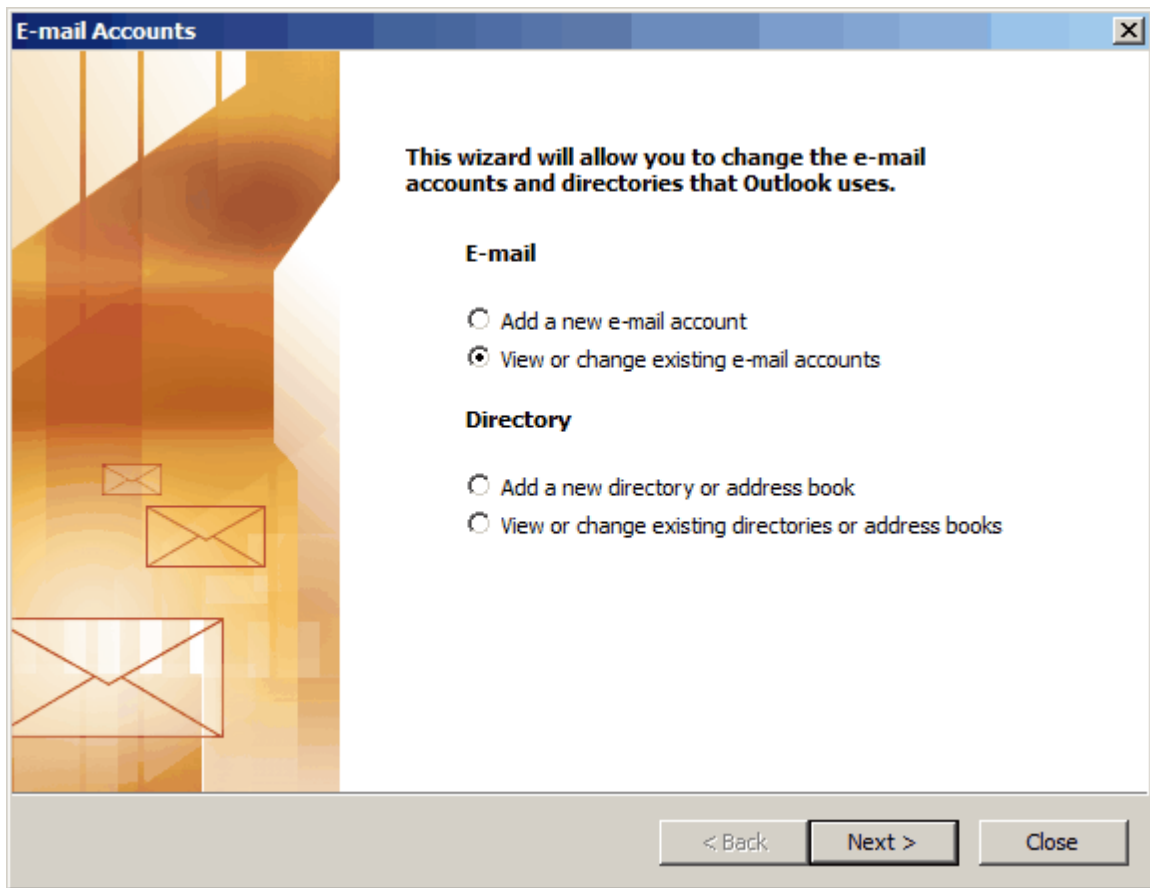
3. Finish the configuration wizard, close Outlook 2003.
4. Log off and log back on as the same user. Do not skip this step or Outlook will not be configured properly!
5. When the computer has rebooted, open Outlook 2003 again. Outlook 2003 will automatically find and try to use an Exchange server at this point. Select OK from the dialog to re-configure the e-mail accounts.



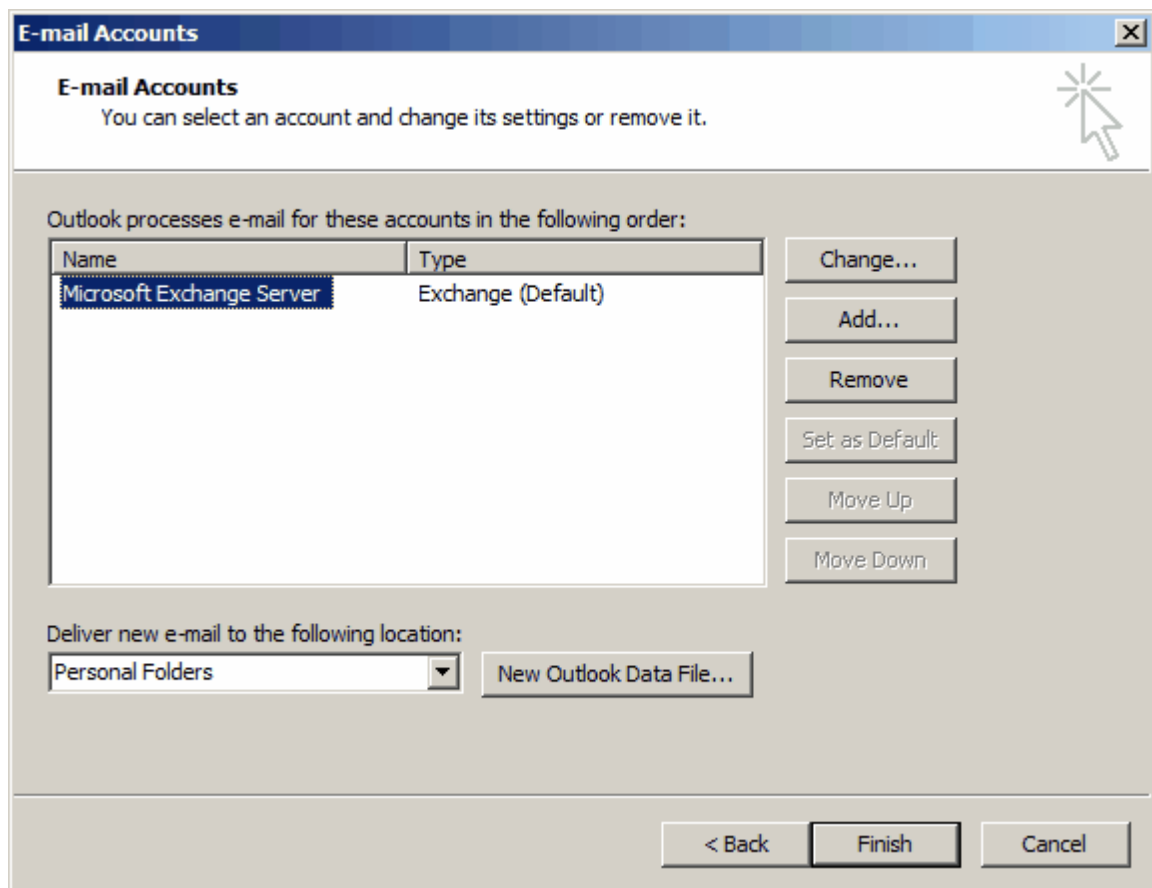
6. In the Microsoft Exchange Server dialog, choose Cancel.



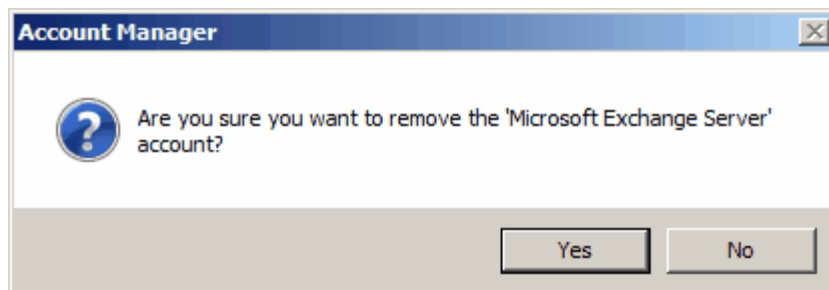
7. In Outlook go to Tools - Account Settings... to edit the e-mail accounts.
 - a. From the E-mail Accounts dialog select "View or change existing e-mail accounts" and select Next.



8. Select the *Microsoft Exchange Server* from the list of e-mail accounts and then click the Remove button to delete the account.



9. Click Yes to confirm.



10. Outlook 2003 is now ready to use with Document Conversion Service.

Animated Images and Movies with FFmpeg.exe

An animated image is a sequence of frames that, when displayed in order, create a short looping animation. A GIF image is an example of an animated image format widely used in social media.

Similarly, MOV, MP4, and other movie formats also consist of a sequence of images or frames. Displaying 24 of these frames per second gives movies their portrayal of motion.

Converting animated images and movies to PDF, TIFF, or other formats creates a *flip-book*-like series of images if you have FFmpeg for Windows installed.

See [How To Install FFmpeg on Windows](#) for instructions on installing FFmpeg. Support for this for the following formats is part of Document Conversion Service beginning with version 3.0.031 and works with the following formats:

- *.mpg
- *.mov
- *.mp4
- *.m4v
- *.avi

Windows Imaging Component (WIC) Add-Ons and Extensions

In addition to the built-in image formats supported by the PEERNET Image Converter, you can add additional image and file formats by installing Windows Imaging Component (WIC) Add-Ons and extensions such as the FastPictureViewer Codec Pack and the DjVu Shell Extension Pack.

These codec packs and shell extensions need to be installed under the same account that Document Conversion Service is running under. In most cases this is the DCSAdmin created as part of the install.

The [DjVu Shell Extension Pack](#) will add support for the DejaVu file format (*.djvu).

See the [FastPictureViewer Codec Pack](#) information page for a complete list of the 45+ image formats and over 500 raw digital camera formats that are supported.

Installing WIC Add-Ons and Extensions

1. Download the WIC Codec Pack or extension.
2. On the computer where Document Conversion Service is installed, log into the DCSAdmin account, or, if you are using a different account, log into that account instead.
 - a. The DCSAdmin account and password is normally created during the install of Document Conversion Service. If you used a different account and password during the install, you will need to log into that account instead.
3. Once logged into the account that Document Conversion Service runs under, install the WIC codec pack or extension.
4. Log out of the DCSAdmin.
5. Restart Document Conversion Service to pick up the added components.

Internet Explorer

Document Conversion Service uses Internet Explorer to convert HTM, HTML and MHT files. When dealing with MHT and HTML files with large images, and older style HTML files formatted for earlier browser versions the options for image scaling and browser emulation may need to be configured to produce the desired output file.

These options are set in the Internet Explorer section of the application configuration file. Changing these options will require a restart of Document Conversion Service for the new settings to take effect.

Opening the Configuration File


Go to **DCS Dashboard** - DCS-Settings - Edit DCS Configuration to edit the configuration file using a visual GUI.

You can also go to Start - All Programs - PEERNET Document Conversion Service 3.0 - Edit DCS Configuration File to edit the configuration file using the DCS Editor.

The configuration file can also be opened in any XML editor and can be found here:

Configuration file location:

C:\Program Files\Document Conversion Service 3.0\Core\PNJobItemProcessor.exe.config



Configuration Section for Internet Explorer

```

<AppFactories>
  <Factories>

    <AppFactory Name="Internet Explorer"
      Type="PEERNET.PNDocConv.Applications.PNInternetExplorerApplicationFactory"
      Assembly="PNInternetExplorerApplicationFactory">
      <Settings>
        <add Name="Enabled" Value="auto"/>
        <add Name="MaxInstances" Value="auto"/>
        <add Name="RecycleThreshold" Value="0"/>
        <add Name="DocumentOpenTimeout" Value="360000"/>

        <!-- Value range 30 - 100 -->
        <add Name="ConverterPlugIn.PNIExplorer.ShrinkToFitScaleMin" Value="30"/>

        <!-- Values: Empty string, IE7, IE8, IE8FORCE, IE9, IE9FORCE, IE10, IE10FORCE, IE11 -->
        <add Name="ConverterPlugIn.PNIExplorer.BrowserEmulation" Value="" />

      </Settings>
    </AppFactory>
    ...
  </Factories>
  <Settings>
    <!-- Global factory settings -->
    <add Name="MaxInstances" Value="auto"/>
    <add Name="RecycleThreshold" Value="0"/>
  </Settings>
</AppFactories>

```

Setting the Minimum Scale For Internet Explorer

HTML files and MHT files such as email messages from Outlook can sometimes have very wide images. By default, these files are always printed with Shrink-to-Fit enabled and a minimum scale factor of 30. This means that the page will shrink to at most 30% of its original size to fit the image contents on the page.

If you need the images to be scaled larger, the setting *ConverterPlugin.PNIExplorer.ShrinkToFitScaleMin* can be adjusted from between 30 to 100 to get the size of image you want.

This option is set at the application level and cannot be changed per file. Changes to this setting require a restart of Document Conversion Service to take effect.

Setting the Browser Emulation for Internet Explorer

In certain cases, older HTML files created for previous versions of Internet Explorer will not convert correctly when printed using the latest version of Internet Explorer. This is because Internet Explorer runs with *Edge compatibility* by default and it is this new compatibility and rendering that has a problem with the older style HTML.

If you have these type of files, the setting *ConverterPlugin.PNIExplorer.BrowserEmulation* can be used to force Internet Explorer to emulate older versions of the browser so that the files are rendered properly based on the older browsers rendering engine.

This option is set at the application level and cannot be changed per file. Changes to this setting require a restart of Document Conversion Service to take effect.

Outside-In AX

In order to use the Outside-In AX converter to convert files, you will need to download and install the 32-bit version of Oracle's Outside In Viewer Technology that you are licensed for. To get this component, visit [Oracle Outside In Technology Downloads](#) and download the appropriate *Outside-In Viewer Technology*.

Starting with version 8.4.1, the Outside-In Viewer needs to be [installed](#) and [registered](#) under the same account that Document Conversion Service is running under. In most cases this is the DCSAdmin created as part of the install.

For versions prior to 8.4.1, the Outside In Viewer can be installed under any user account on the machine running Document Conversion Service.

If you need to create vector Adobe PDF files, you will also need to [Add Printer Permissions](#) to the Microsoft XPS Document Writer printer for the *Everyone* account. Alternatively you can add the permissions for just the account that Document Conversion Service is running under - in most cases this is the *DCSAdmin* created as part of the install.

Installing Outside-In Viewer Technology 8.4.1 or later

1. Download the latest 32-bit version of the Outside In Viewer Technology that you are licensed for. The other versions will not work with Document Conversion Service.
2. On the computer where Document Conversion Service is installed, log into the DCSAdmin account, or, if you are using a different account, log into that account instead.
 - a. The DCSAdmin account and password is normally created during the install of Document Conversion Service. If you used a different account and password during the install, you will need to log into that account instead.
3. Once logged into the account that Document Conversion Service runs under, install the Outside In Viewer by running the downloaded setup.

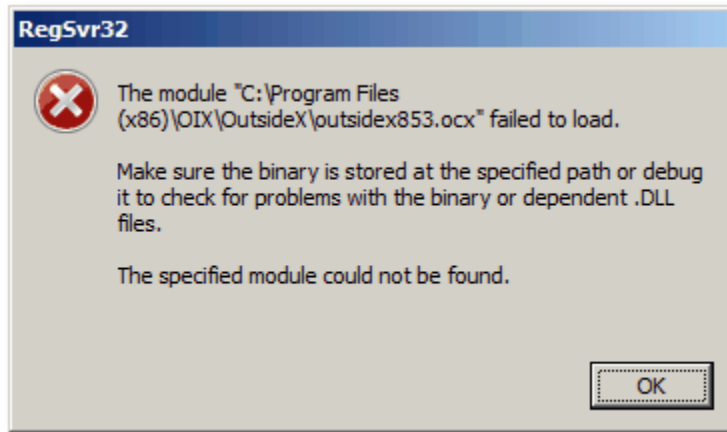
Registering the Outside-In Control

When the install is complete the Outside In Active X control still needs to be registered at an administrative level to work properly with Document Conversion Service.

Open an administrative level command prompt and type the following, replacing the ### with the version of Outside-In AX you have installed.

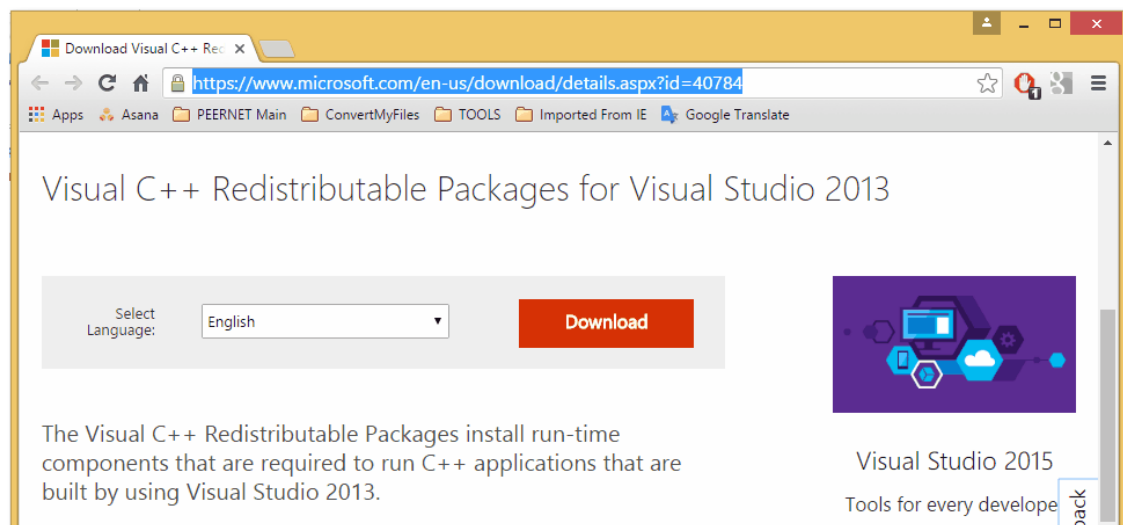
```
C:\Windows\system32\regsvr32.exe "C:\Program Files (x86)\OIX\OutsideX\outsidex###.ocx"
```

Starting with version 8.5.3, the Outside In Active X control has a dependency on the Visual C++ Redistributable packages for Visual Studio 2013. If you see this error message when trying to register the control you will need to download and install the package as per the instructions below. If you don't see the error message then the package has most likely been installed by another piece of software on your computer.

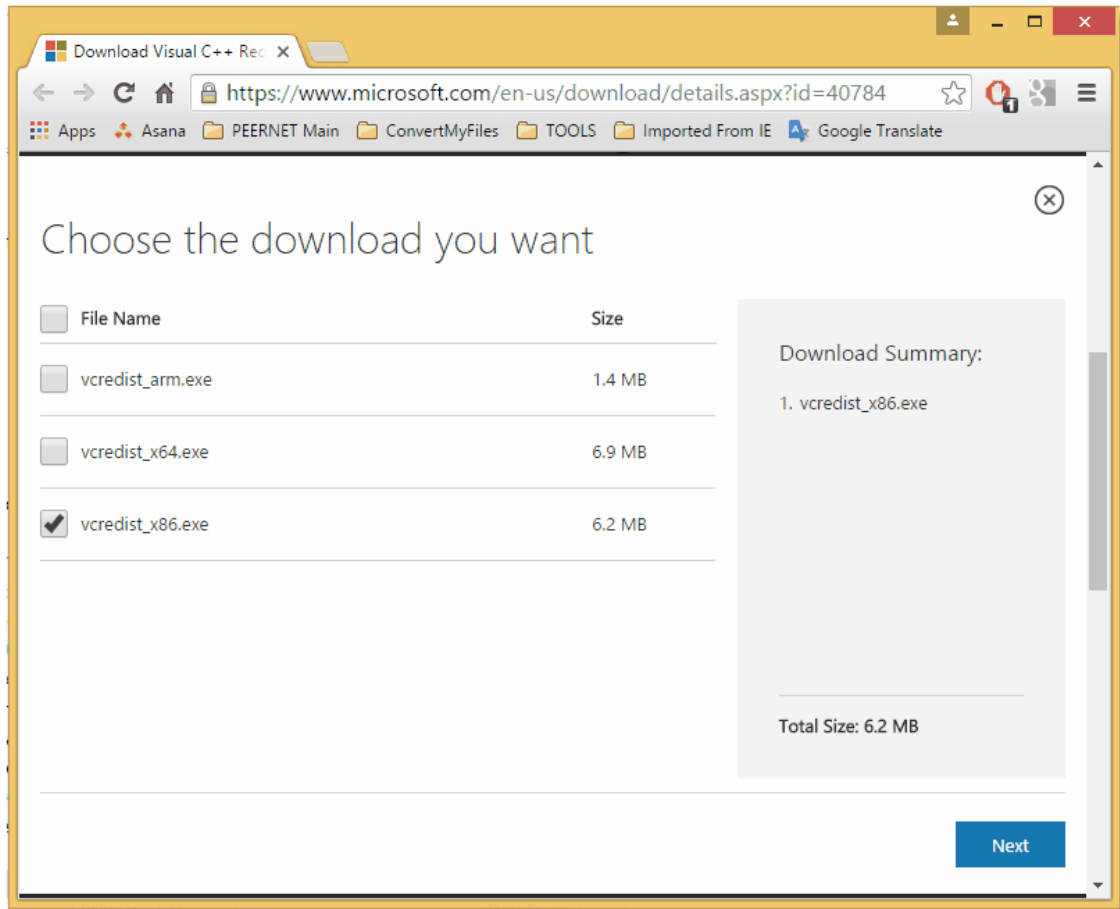


Downloading the Visual C++ Redistributable Package for Visual Studio 2013

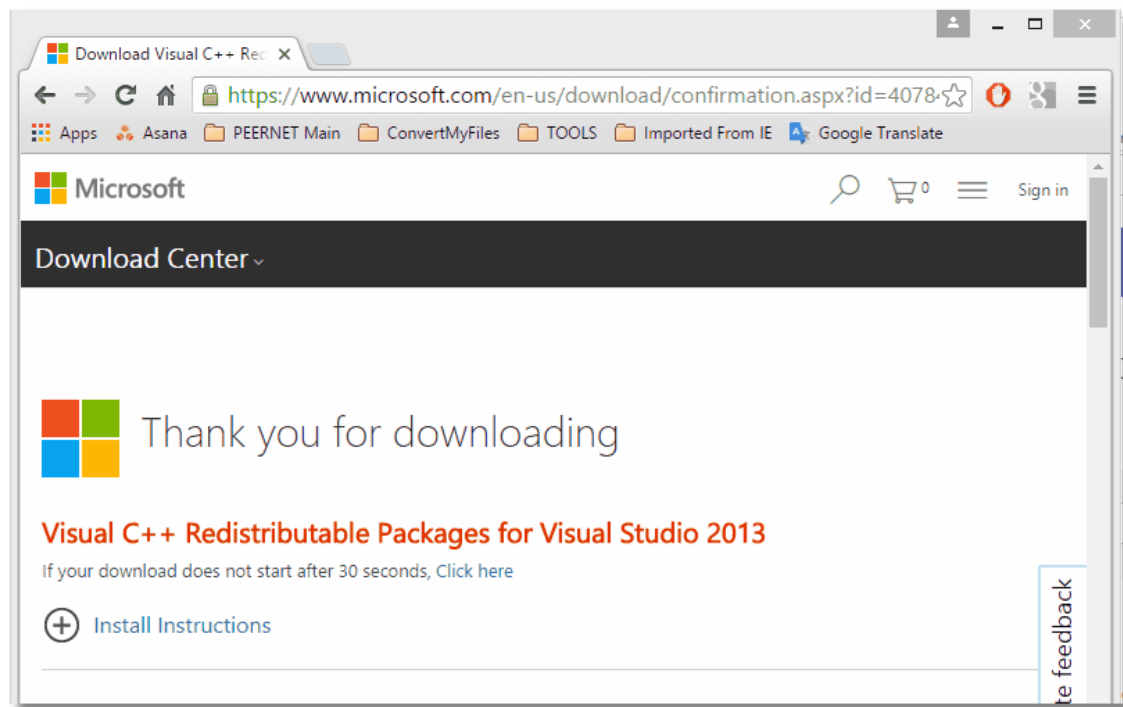
1. Go to the following link: <https://www.microsoft.com/en-us/download/details.aspx?id=40784>
2. Click the red **Download** button on the right-hand side.



3. Choose the **vcredist_x86.exe** download and click the **Next** button. The other downloads will not work with Document Conversion Service.



4. The download should start automatically.



5. Once downloaded, run the **vcredist_x86.exe** setup to install the required dependencies.
6. [Register](#) the Outside In Active X as shown above.

Adding Printer Permissions to Microsoft XPS Document Writer

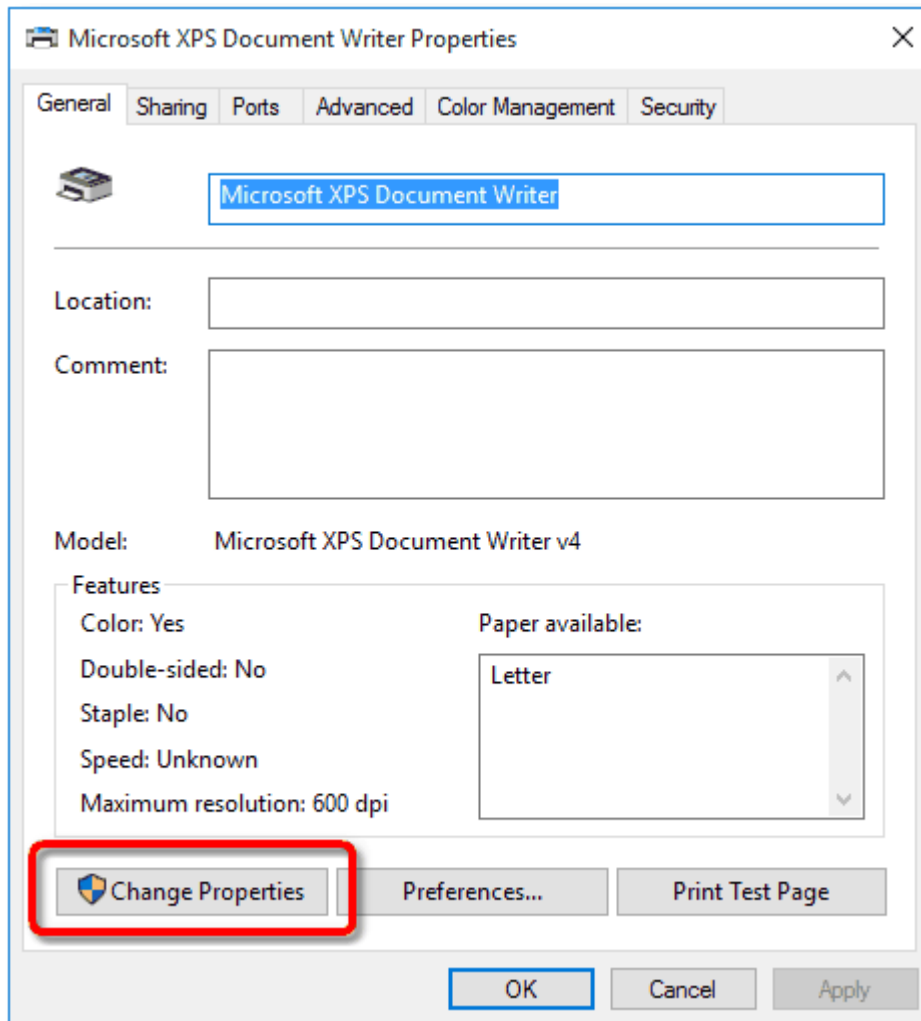
This is only needed if you are creating vector Adobe PDF files using the profile *Adobe PDF Multipaged*, or the settings listed within.

The instructions below show how to add this permission to the Microsoft XPS Document Writer for the **Everyone** account. You can instead add these permissions for the account that Document Conversion Service is running under; this is often the DCSAdmin account. If you used a different account when installing, add these permissions for that account instead.

You will need to have Administrative permissions to make these changes.

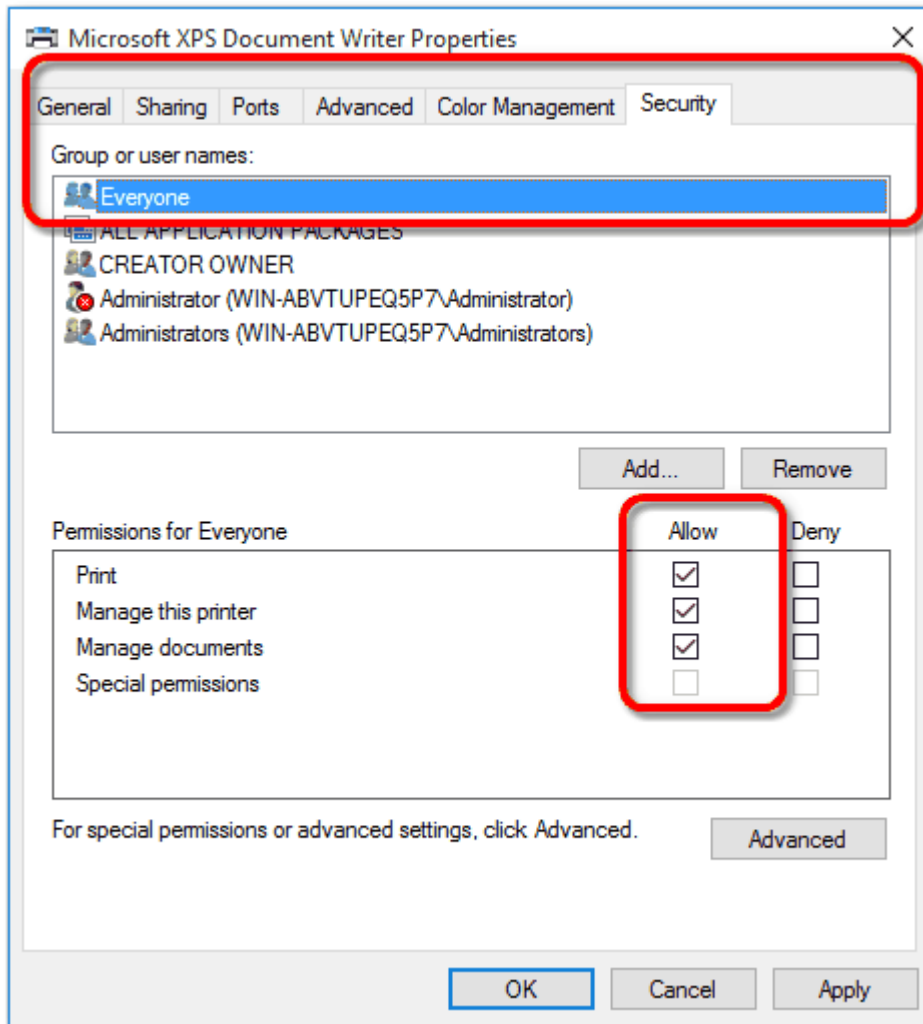
1. Open the *Devices and Printers* folder by typing *Printers* into the Search field in the Start menu.
2. Right-click on the Microsoft XPS Document Writer printer and select *Printer Properties* from the context menu.

3. On the **General** tab, Click the *Change Properties* button in the lower left.



- Click on the Security tab, select the *Everyone* account, then make sure that the permissions *Print*, *Manage this printer*, and *Manage documents* are checked.

If you only want to add the permissions for the DCSAdmin account or your own custom account, click the *Add...* button to show the dialog listing all available users. Select DCSAdmin or your custom account to add that user to the list. Once added, select that user in the list and make sure that the permissions *Print*, *Manage this printer*, and *Manage documents* are checked for that user.



Completing the Changes

Once the above changes have been made, log out of the DCSAdmin account, or the account you are using.

If the Document Conversion Service is running, you will need to stop and restart the conversion service to pick up the added component.

Optical Character Recognition (OCR) with Document Conversion Service

Optical Character Recognition, or **OCR** for short, searches for and recognizes text (characters) on scanned pages or images and extracts it as digital text.

With this digital text, we can create searchable PDF files from images or PDF documents containing scanned pages. A searchable PDF is a PDF file in which you can select and copy the text on the pages and use the search function to look for specific words and phrases in the file.

When recognizing text, the OCR engine has to know which languages to look for on the page. OCR works by analyzing the patterns, shapes, and curves of the text characters on the page and matching them to predefined information for different characters in each language. It assigns a confidence score for each language, with the highest score determining the language chosen.

Outside factors such as image quality, the font used, and any image background on the pages will all affect the validity of the OCR results.

- [Using OCR to Create Searchable PDF Files](#)
- [Adding New Languages for OCR](#)

Using OCR to Create Searchable PDF Files

When you convert images or PDF to editable PDF, the digital text found by the OCR engine gets added as an invisible text layer to each page in the new PDF file, making the file's content searchable. The new PDF contains the original image and an invisible layer of text. It is this layer of text that makes the PDF searchable.

Optical Character Recognition can only be used when creating PDF files. OCR can increase the processing time for file conversion and is supported by the following converters:

- Built-in PDF Converter
- Built-in Image Converter



Caution

This feature is not supported on Microsoft® Windows Server 2008 R2 and Microsoft® Windows 7.

Searchable PDF in the Watch Folder Service

The Watch Folder Service includes a sample conversion folder, **OCR to AdobePDF Watch Folder**, that is already configured for OCR and creates searchable PDF for English, French and Spanish text. See [OCR Images and Scanned PDF Files to Searchable PDF](#).

Creating Searchable PDF Using Profiles

When converting using the desktop conversion tools, [Convert File](#) or [Drop Files Converter](#), the [command line tools](#), and the [PEERNET.ConvertUtility.dll](#), you use a profile to tell Document Conversion Service what type of file to create. A profile is a group of settings stored as a collection of name-value pairs in an XML document.

Document Conversion Service includes a [collection of profiles](#) for converting to various output formats and performing other actions when converting documents, such as e-Discovery and OCR.

Desktop Conversion

The desktop conversion tools use the selected *profile* to determine what type of file to create. The two profiles, **Adobe PDF OCR to Searchable**, and **Adobe PDF OCR to Searchable Serialized**, are already configured for OCR and to create searchable PDF for English text.

To OCR images and scanned PDF files using the desktop tools, choose the above profiles when converting.

Command Line Tools

When you are converting using the [command line tools](#), they too use profiles to determine what type of file to create. On the command line, the desired profiles is specified by passing in the name of the profile XML file, with or without the XML extension. To use the OCR profiles on the command line, you would pass **/P="Adobe PDF OCR to Searchable"** or **/P="Adobe PDF OCR to Searchable Serialized"**.

The PEERNET.ConvertUtility.dll

Like the desktop and command line tools, the PEERNET.ConvertUtility also uses profiles to tell Document Conversion Service what type of file to create. Pass in **Adobe PDF OCR to Searchable**, or **Adobe PDF OCR to Searchable Serialized** to create a searchable PDF file.

OCR Profile Settings

OCR is disabled until these options are added to your profile. The sample profiles already have these options set.

ConverterPlugIn.PNBuiltinsOCRPDF.Enabled - Set this to 1 to enable OCR, 0 to turn it off. Default value is 0.

ConverterPlugIn.PNBuiltinsOCRPDF.FirstPageOnly - Set this to 1 to only OCR the first page of any document. Set it to 0 or do not set it to OCR each page in the document. Default is 0.

ConverterPlugIn.PNBuiltinsOCRPDF.Languages - List which languages you want to try to recognize on the page. To look for multiple languages, list the language code for each language separated by a plus sign. For example, the sample profile only looks for English, "**eng**". To look for English, French, and Spanish, you would use the string "**eng+fra+spa**". The default when this is not supplied is English only, "**eng**". The more languages listed the longer the OCR process will take.

Adding New Languages for OCR

The OCR engine needs to know which languages you want to try to recognize on the page. The languages provided with Document Conversion Service, and their language codes are as follows.

Language	Language Code
Arabic	ara
English	eng
French	fra
German	deu
Hebrew	heb
Hindi	hin
Italian	ita
Spanish	spa

To download individual language files, go to [Tesseract Languages Code and Traineddata Files](#). This link also includes a table listing the language code for each traineddata file for each language. To download complete sets of language files go to [Traineddata Files for Tesseract](#).

To add them to Document Conversion Service, copy the desired *.traineddata files into the following folder:

%PROGRAMDATA%\PEERNET\Document Conversion Service\tessdata

Converting Files with Document Conversion Service

Command Line Utilities

Several [command line utilities](#) for converting files and folders are included with Document Conversion Service. These utilities can be called from the DCS command window, scheduled tasks, from batch files or any program that can call an external program.

Watch Folder Service

This included Windows service can watch multiple folders for any files placed in those folders, and will convert those files using the conversion settings for that folder. Multiple folders can be configured, each with their own conversion settings. This type of approach is often called a *hot folder* or *drop folder*. Files dropped into the folders are converted and the original file can be kept or discarded.

Additional features that are a part of the Watch Folder Service are:

- [Large volume batch conversion](#) is built-in for dealing with existing folder structures with a large number of files.
- Starting with version 3.0.009, [processing Outlook message attachments](#) can optionally be enabled as needed for each folder. This will extract and convert any attachments in Outlook Message files (*.msg), as well as the original email message in the file.
- [Clustering](#) can be enabled on any folder as of version 3.0.010. Clustering allows multiple servers running Document Conversion Service and Watch Folder Service to process files from the same watched folder. This can lead to higher conversion throughput and also allows for fail over should one of the servers have to be taken offline.
- Beginning with version 3.0.010, separate [post-processing](#) for success and failure can be enabled for each folder.
- The [ability to automatically create unique file names and flatten folder structures](#) was added with version 3.0.019.

The Watch Folder Service is also included as one of the open source samples to allow for further custom processing of the converted files when completed.

Desktop Conversion Applications

Two desktop conversion applications are supplied as part of the Document Conversion Service and Document Conversion Service Client Redistributable install.

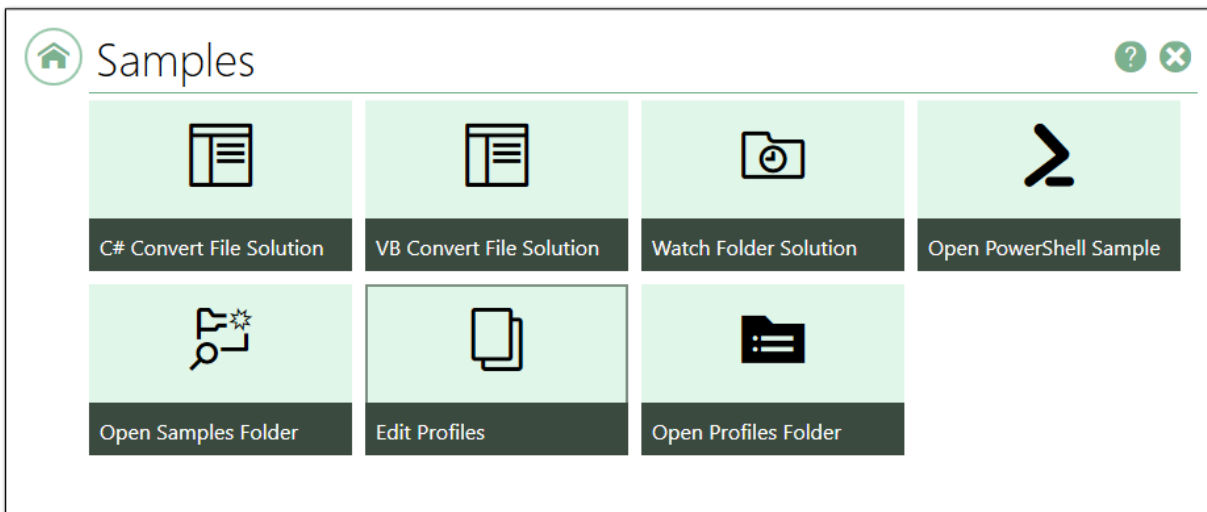
- [The Drop Files Converter Desktop Application](#)- provides a drop area in which to drag and drop files and folders to be converted. The type of file to be created and where it is stored can be customized. Advanced options allow remote conversions and the ability to run a command on each newly created file when the conversion has completed.
- [The Convert File Application](#) - a simple interface for selecting and converting a single file at a time. The type of file being created is determined by the chosen *profile* of settings and remote conversion can also be done. This desktop application is also included as open source sample code.

Sample Programs

Both the Watch Folder Service and the Convert File application are included with Document Conversion Service as sample projects. While these samples can be used on their own, or as a starting point to integrating Document Conversion Service into your own applications, there is no warranty, implied or otherwise, of merchantability or fitness for a particular purpose.

Go to **DCS Dashboard** - Samples to access the Visual Studio solutions for the Watch Folder Service and the Convert File samples. From here you can also access a PowerShell sample.

The Samples folder can be opened from here as well as by going to Start - All Programs - PEERNET Document Conversion Service 3.0 - Samples - Open Samples Folder. The samples can be edited in this location but if you wish to keep the original source code, copy its folder to another location and edit that project.



The Convert File Application

This sample demonstrates using the provided .NET library, PEERNET.ConvertUtility.dll to convert a single file. It is provided in both C#.NET and VB.NET.

Like the command line utilities, the .NET library also uses a conversion profile, an XML file of name-value pairs, to describe the output. The .NET library methods can also take an *IDictionary<String, String>* list of settings instead of the profile. Document Conversion Service includes a set of common conversion profiles; see the section [Conversion Settings](#) for more details, and instructions on creating your own custom profiles.

The Watch Folder Service Sample

This is an advanced sample in C#.NET that demonstrates using the PNDocConvQueueServiceLib COM object from a service in a multithreaded environment. It demonstrates a service watching a *drop folder* for files and converting those files on demand using Document Conversion Service.

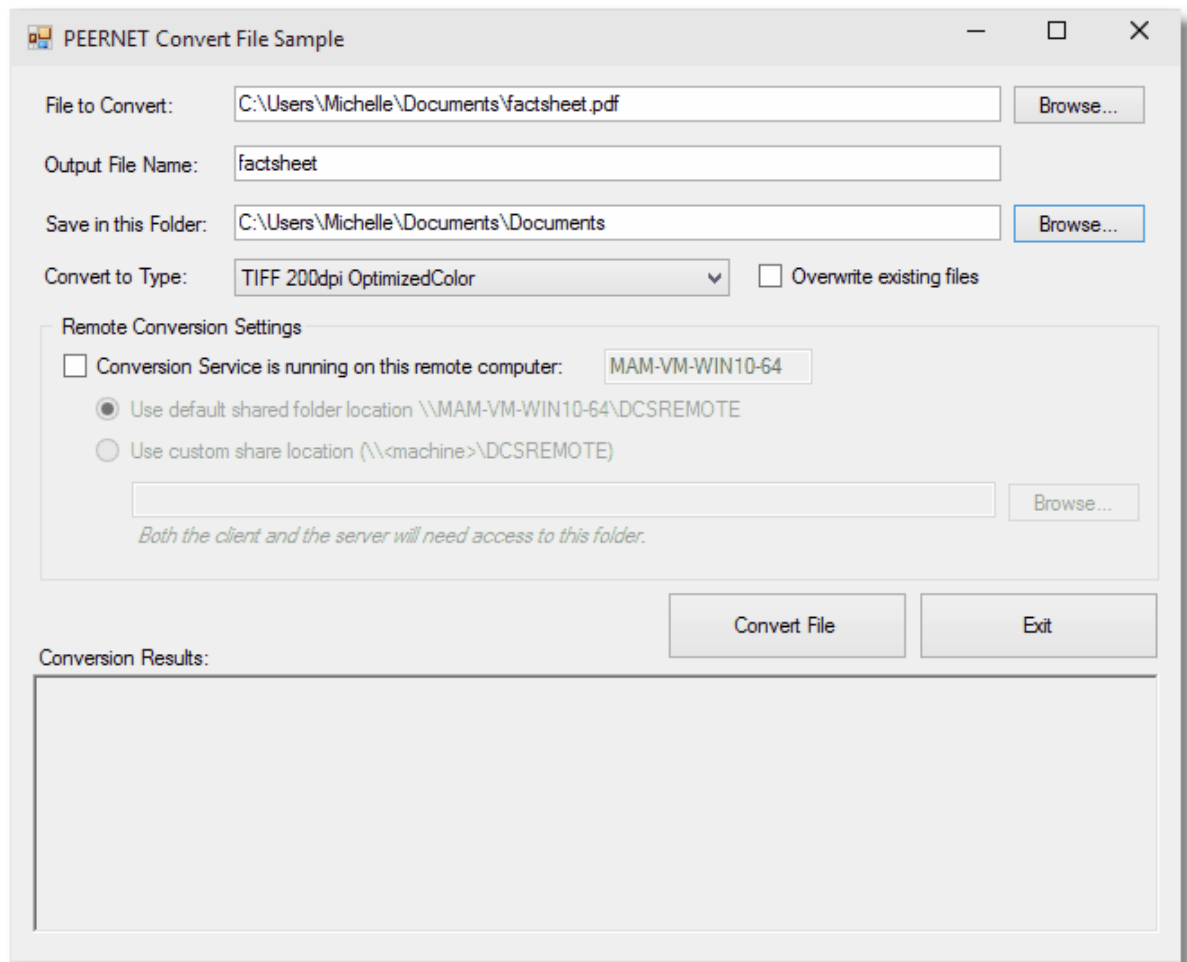
The PowerShell Sample

This PowerShell script demonstrates how to set conversion options and convert a file using the PEERNET.ConvertUtility.dll. After conversion, it shows how to read the returned results for the created file list or conversion errors.

The Convert File Application

The Convert File application is a simple application that converts a single, chosen file using Document Conversion Service and a selected conversion profile. It also includes the ability to convert the file remotely on another computer.

This application is also provided as a Visual Studio project in both VB.NET and C#.NET and shows how to convert a file using the provided .NET library, PEERNET.ConvertUtility.dll.



The type of output created is based on the conversion profile chosen. A selection of common conversion profiles are included with the Document Conversion Service install. See [Creating and Customizing Profiles](#) for more information about the contents of the profiles, a list of profiles included with Document Conversion Service, and how to create your own.

The application uses the file extension of the source file to determine what converter to use to convert the file. The default file extension to converter mapping provided through the PEERNET.ConvertUtility.dll is used. As with profiles, this file extension mapping can be customized, but rarely needs to be. See the section [File Extension to Converter Mapping](#) for details.

You can also use this program to test remote document conversion by following the steps in [Setting up Client-Server Conversion](#).

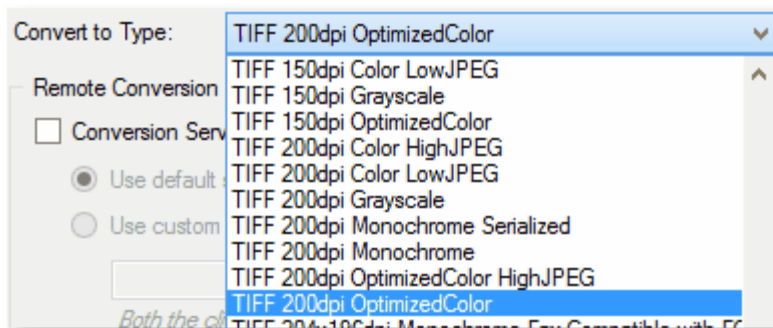
Running the Convert File Application



Before you begin...

Before running the application, follow the steps in [Starting and Stopping the Service](#) to start the Document Conversion Service. If the service is not started, an error message will display when you try to convert documents.

1. Open the application by going to the **DCS Dashboard - Desktop Conversion - Convert a File**.
 - a. You can also open this sample by going to Start - All Programs - PEERNET Document Conversion Service 3.0 – Convert a File....
2. Choose a file to convert using the **Browse** button or typing in the file name. The *Output File Name* field will be populated from the chosen file name.
3. Choose a folder in which to save the output file.
4. Use the **Convert to Type** drop down list to select your output format from the list of available profiles.



5. Click Convert to convert the chosen file. When the conversion process is finished, the results are displayed in the listbox at the bottom.

Inside the Sample Code - Calling the PEERNET.ConvertUtility.dll Methods

The conversion process itself happens in the *Click* event handler of the "**Convert File**" button. Below is a simplified version of the C# version of that event. Field checking and error reporting is stripped out for brevity.

Go to Start - All Programs - PEERNET Document Conversion Service 3.0 – Samples - Open Samples Folder to see the C# or VB.NET sample code for the full function in the language of your choice.

Code Sample - Click Event Handler for Convert File in C#

```
using PEERNET.ConvertUtility;

private void btnConvert_Click(object sender, EventArgs e)
{
    // conversion results returned, use to find files created or errors
    PNConversionItem resultItem = null;

    try
    {
        lbResults.Items.Add("Converting...");

        // This is the single call needed to convert a file
        resultItem =
            PNConverter.ConvertFile(tbInputFile.Text,
                                   tbSaveFolder.Text,
                                   tbOutputFileName.Text,
                                   cbOverwriteExisting.Checked,
                                   false,
                                   false,
                                   cmbBoxFileTypes.Text,
                                   String.Empty,
                                   String.Empty,
                                   cbUseDCOM.Checked ? tbDCOMName.Text : String.Empty,
                                   String.Empty,
                                   String.Empty);
    }
    catch (Exception ex)
    {
        String errMsg = String.Format("An error occurred during conversion. {0}",
                                      ex.ToString());

        lbResults.Items.Add(errMsg);
        MessageBox.Show(this, errMsg, this.Text);
    }
    finally
    {
        DisplayResultsItems(resultItem);
    }
}
```

The Drop Files Converter Desktop Application

The Drop Files Converter desktop application is a simple utility that provides an area in which to drag and drop files, and if enabled, folders, to be converted. The type of file to be created and where it is stored can be customized. It includes advanced options to allow remote conversions and the ability to run a command on each newly created file when the conversion has completed.

This utility is provided as part of Document Conversion Service.

Running the Drop Files Converter Application

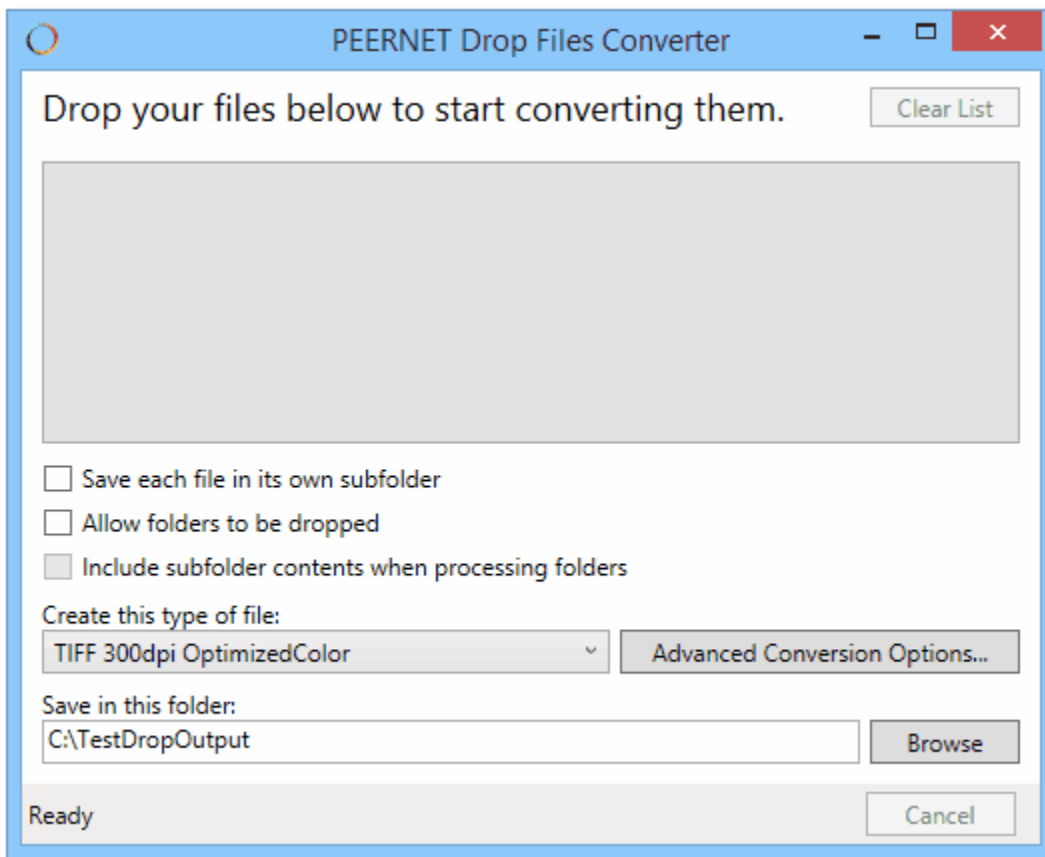


Before you begin...

Before running the application, follow the steps in [Starting and Stopping the Service](#) to start the Document Conversion Service. If the service is not started, a message stating "Waiting for service" will display when you try to convert documents.

Open the application by going to the **DCS Dashboard - Desktop Conversion - Convert Files and Folders**.

You can also open the application by going to **Start - All Programs - PEERNET Document Conversion Service 3.0 – Drop Files Converter**.

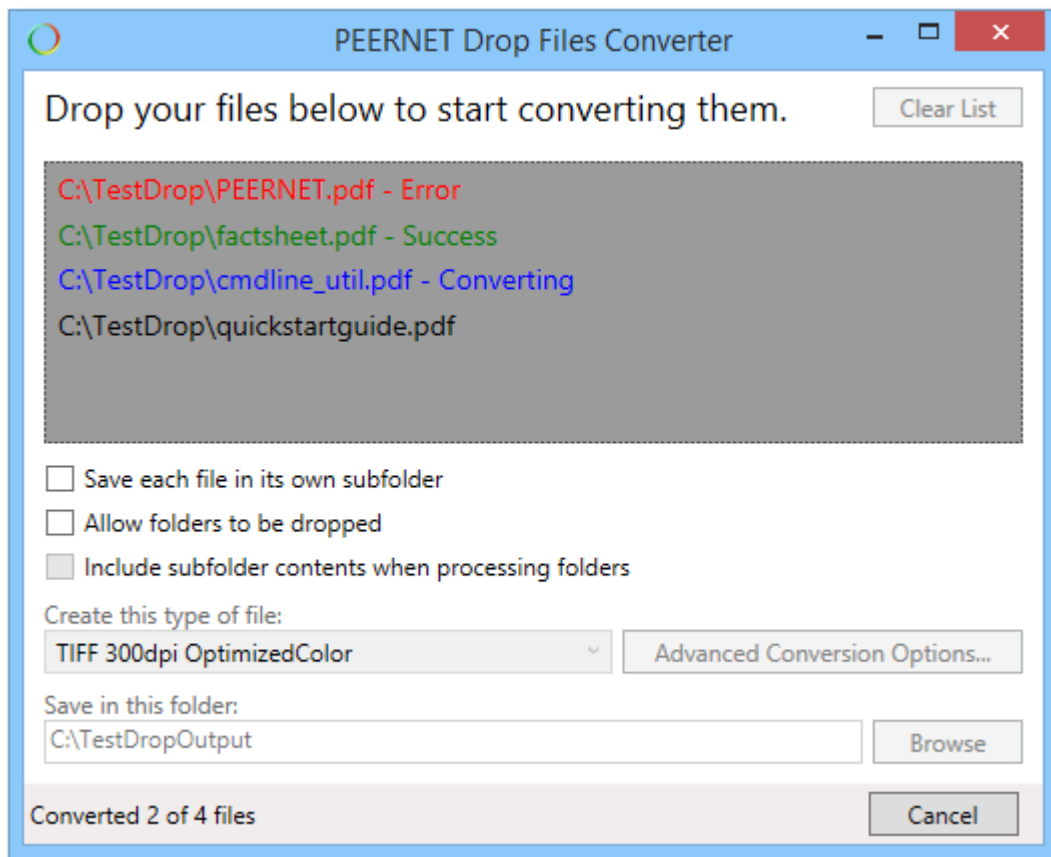


File and folders can be converted by dragging and dropping them onto the light gray drop area. This area changes to a darker gray color to reflect that a file or folder can be dropped.

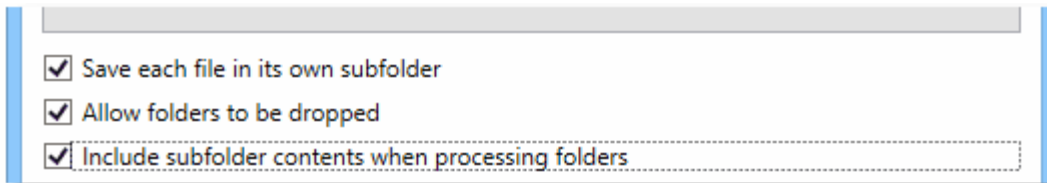
Once a file or a collection of files is dropped, the drop area stays dark gray and the conversion will start immediately and no more files can be dropped until the current collection has been converted. Conversion options are disabled and the Cancel button enabled. The Cancel button lets you stop converting a group of files, but cannot cancel the currently running conversion. The cancel action will take place after the currently converting file is finished.

As files are processed, information about their state is displayed in the drop area, and their color changed to reflect their status.

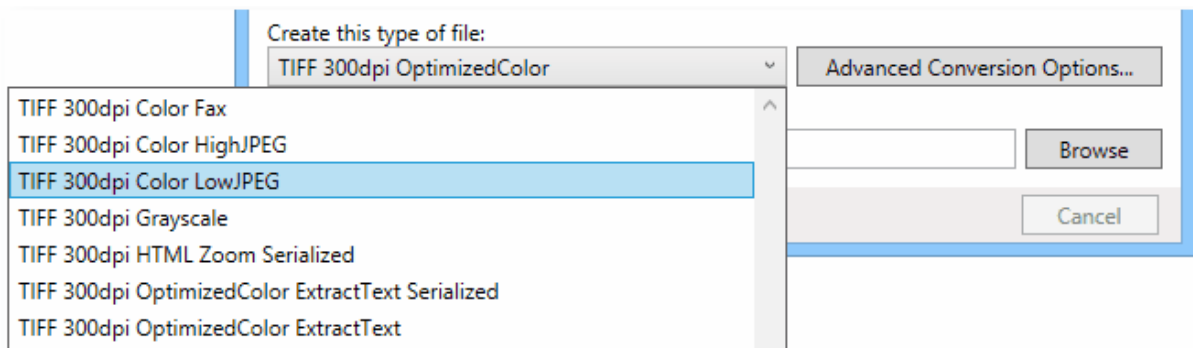
When the collection of files has been converted, the list of files can be cleared using the **Clear List** button in the upper right.



By default folders and subfolders are not processed, this option can be enabled if needed. Optionally, each file can be created in its own subfolder under the output folder.



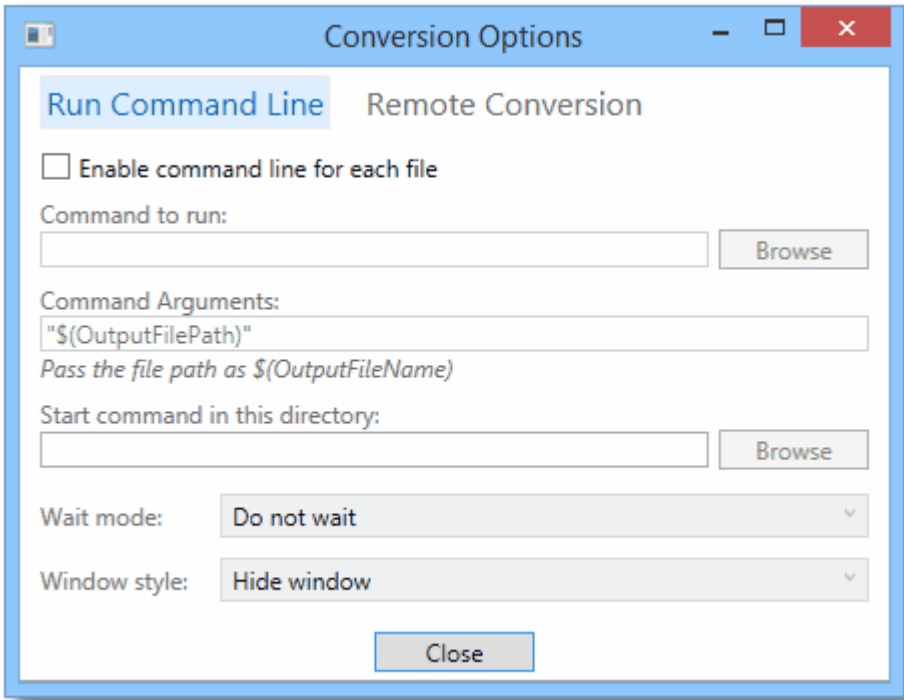
The type of output created is based on the conversion profile chosen. A selection of common conversion profiles are included with the Document Conversion Service install. See [Creating and Customizing Profiles](#) for more information about the contents of the profiles, a list of profiles included with Document Conversion Service, and how to create your own.



The application uses the file extension of the source file to determine what converter to use to convert the file. The default file extension to converter mapping provided through the PEERNET.ConvertUtility.dll is used. As with profiles, this file extension mapping can be customized, but rarely needs to be. See the section [File Extension to Converter Mapping](#) for details.

Advanced Conversion Options

The application includes some advanced options for running a command on each successfully converted file as well as allowing remote conversion. You can switch between these options by clicking on the text at the top of the window.



Run Command Line

When enabled, the command line will only run for successfully converted files. A command is normally another executable, batch file or other command line program. Type in your command or use **Browse** button to select it.

To pass the path of the created file into your command, use the macro name `$(OutputFilePath)`. If needed, command arguments should be enclosed with quotation marks, especially if they have spaces in them.

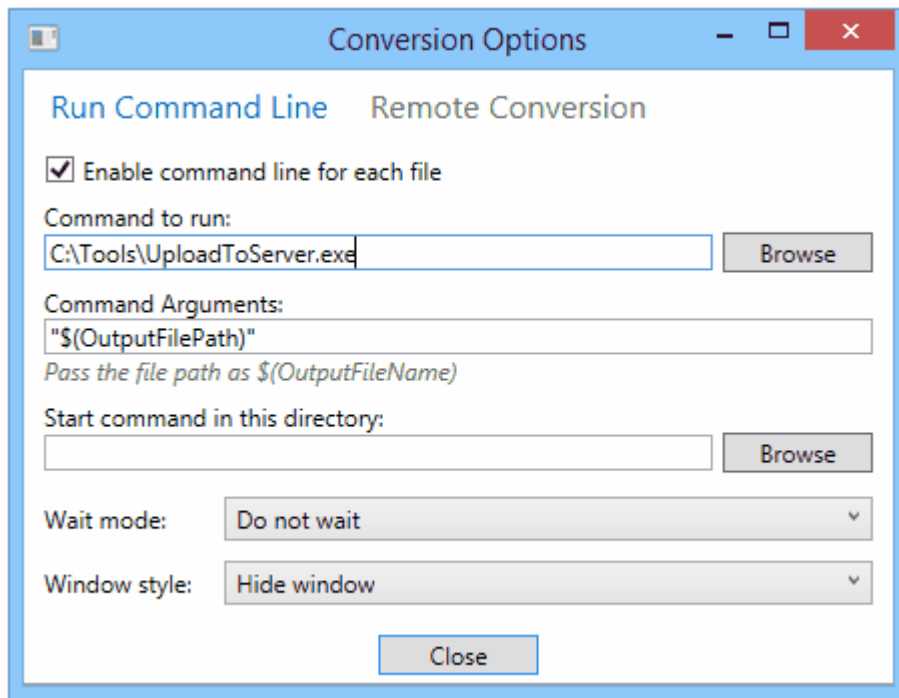
When running a command there are options guiding whether or not to wait for the command to complete before moving onto the next file, and controlling how the command window is displayed, if at all.

Wait Mode	Wait for command to complete before continuing - wait for the command to complete before continuing on to the next file. Wait for command to complete and return error code - waits for the command to complete before continuing on to the next file and shows the exit code in the file status. Do not wait - does not wait for the command to complete. (Default)
Window Style	Normal - display the window in its normal state. Min - display the window minimized to the taskbar Max - display the window maximized.

Hidden - do not show the window. (Default)

The default settings for this are to hide the command window and to not wait for the command to complete. When first adding a command, it can be helpful to display the window and wait for the command to complete to ensure that it is working as expected and that arguments are being passed correctly. Once this has been determined, it can be set back to hidden and to not wait.

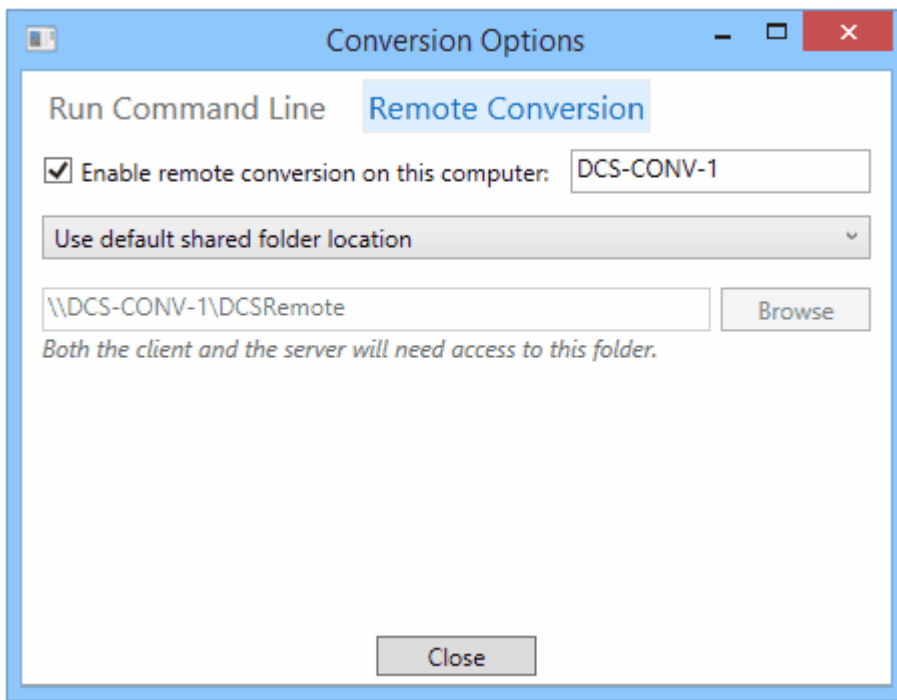
If the command line settings have been enabled, the file status in the drop area will change to reflect that a command is being run as part of the conversion process.



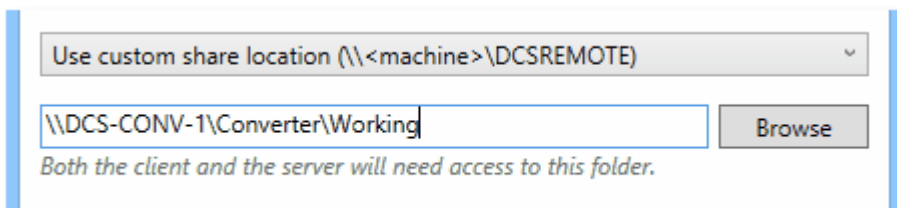
Remote Conversion

You can also use this sample program to test remote document conversion where Document Conversion Service is installed on a different computer. See the steps in [Setting up Client-Server Conversion](#) to learn more about setting Document Conversion Service up in this environment.

For remote conversion, you will need to know the name of the server where Document Conversion Service is installed and running, and a temporary conversion folder that is accessible to both the client and the server is required. A network shared folder named DCSREMOTE is automatically created on the server as part of the Document Conversion Service installation and can be used as this temporary conversion folder, or a custom remote folder chosen as needed.



If you do need to use a custom share folder, select the second option from the drop down list and provide the path to the folder.



Command Line Utilities

Utility	Descriptions
DCSConvertFile	Converts a file using Document Conversion Service. The file can be converted in place or saved in a different location.
DCSConvertFileList	Given a text file containing a list of files to convert, or a list of files provided on the command line, converts all files using Document Conversion Service.
DCSConvertFolder	Walks the given folder and converts all files, or all files matching a provided search filter, using Document Conversion Service. The utility can optionally also process all subfolders under the starting folder as well.
DCSCombineFiles	Given a text file containing a list of files, and/or a list of files provided on the command line, this utility will convert the files using Document Conversion Service and combine all of the files together into a single multipaged file or a collection of serialized pages. The files are appended together in the order in which they are received.
DCSCombineFolder	Walks a folder and combines all files, or all files matching a search filter, together into a single file or a collection of serialized pages using Document Conversion Service. The utility can optionally also process all subfolders under the starting folder as well.
DCSExtractResults	Each of the above utilities can create a results log file containing a complete snapshot of the conversion information for each file converted, in both a success and failure case. This utility can be used to extract information from the results log files, such as all files created, or any errors that occurred.
DCSCreateFileList	Searches a folder, and optionally any subfolders and lists of files matching the search filter specified.
DCSLicenseDaysLeft	Echos to the command line the current license level and how many days remain in the subscription.

When using the above command line utilities the type of output created is controlled by the settings passed in through the conversion profile, a XML file of name-value settings. A selection of common conversion profiles are included with Document Conversion Service. See the section [Conversion Settings](#) for more details, and instructions on creating your own custom profiles.

The command line utilities all return the following error codes:

- 0 – success
- 1 – failed
- 2 – invalid parameters, when the command line tool has parameters

The results of each command line utility are sent to standard out, while any errors that may have happened are sent to standard error. To capture this information you can:

- use the output redirection operators `>` and `>>` to save or append the standard output results in a file
- use the pipe operator `(|)` to send the standard output to another program as input
- redirect only the standard output to a file with the operator `1>C:\DCS\files.txt`
- redirect only the standard error to a file with the operator `2>C:\DCS\err.txt`

DCSConvertFile

A command line utility to convert a file using Document Conversion Service. The Document Conversion Service must be running, either locally or on a remote computer for the file to be converted. If it is not running the command will return immediately with an error.

```
DCSConvertFile /P=profile [/S=save location] [/N=output name] [/O] [/NE] [/L]
               [/D="name:value"] [/E=extension map]
               [/W=wait time] [/FAIL=failed results log file location]
               [/SIL=conversion log file path]
               [/C=remote computer name;remote scratch folder]
               [/T=alternate temp folder]
               sourcefile
```

Sample Command Lines

Convert a single file to a TIFF:

```
DCSConvertFile /P="TIFF 200dpi Monochrome" "C:\Test\Document.doc"
```

Send the file C:\Test\Document.doc to Document Conversion Service to be converted using the settings contained in the conversion profile *TIFF 200dpi Monochrome.xml*.

The converted file, *Document.doc.tif*, is saved in C:\Test\, the same location as the source file.

If a file of the same name already existed, this conversion would fail and a *.failed* folder would be created in the same location as the source document, C:\Test. The results log would be named *Document.doc.failed.dcsresults* and saved to a new subfolder under the *.failed* folder. The subfolder is named using the date and time of the conversion to keep subsequent runs separate.

To overwrite an existing file the */O* switch would need to be added to the above command. If you did not want the source file extension as part of your file name, the */NE* switch would need to be added.

Convert a single file to a TIFF with a specific output name, overwrite existing files:

```
DCSConvertFile /O /N="Opt_Document" /P="TIFF 300dpi OptimizedColor"
               "C:\Test\Document.doc"
```

Send the file C:\Test\Document.doc to Document Conversion Service to be converted using the settings contained in the conversion profile *TIFF 300dpi OptimizedColor.xml*.

The converted file will be named *Opt_Document.tif* and saved in the same location as the source file. If a file of the same name already existed, this file would be overwritten with the new file.

Convert a single file to a TIFF in a specific location:

```
DCSConvertFile /S="C:\Output" /P="TIFF 200dpi Monochrome" "C:\Test\Document.doc"
```

Send the file C:\Test\Document.doc to Document Conversion Service to be converted using the settings contained in the conversion profile *TIFF 200dpi Monochrome.xml*.

The converted file will be named *Document.doc.tif* and placed in the folder named C:\Output.

If a file of the same name already existed, this conversion would fail and a *.failed* folder would be created in the same location as the source document, C:\Test. The results log would be named *Document.doc.failed.dcsresults* and saved to a new subfolder under the *.failed* folder. The subfolder is named using the date and time of the conversion to keep subsequent runs separate.

Convert a single file to a TIFF in a specific location, wait up to 5 minutes for the conversion service to start:

```
DCSConvertFile /S="C:\Output" /P="TIFF 200dpi Monochrome" /W=300
"C:\Test\Document.doc"
```

Send the file C:\Test\Document.doc to Document Conversion Service to be converted using the settings contained in the conversion profile *TIFF 200dpi Monochrome.xml*. If Document Conversion Service is not running, wait up to 5 minutes (300 seconds) for the conversion service to be available.

The converted file will be named *Document.doc.tif* and placed in the folder named C:\Output.

If a file of the same name already existed, this conversion would fail and a *.failed* folder would be created in the same location as the source document, C:\Test. The results log would be named *Document.doc.failed.dcsresults* and saved to a new subfolder under the *.failed* folder. The subfolder is named using the date and time of the conversion to keep subsequent runs separate.

Convert a single file to a vector PDF document, remove source extension and save the conversion results log file:

```
DCSConvertFile /P="Adobe PDF Multipage" /L /NE "C:\Test\Document.doc"
/FAIL="C:\Test\FailedLogs\\" /D="UseDateTimeInFailedFolder:FALSE"
```

Send the file C:\Test\Document.doc to Document Conversion Service to be converted using the settings contained in the conversion profile *Adobe PDF Multipage.xml*. This profile creates vector PDF where possible.

The converted file will be named *Document.doc.pdf* and saved in the same location as the source file. A conversion results log file, *Document.doc.succeeded.dcsresults* will also be saved in same location as the source file.

If a file of the same name already existed, this conversion would fail. The results log would be named *Document.doc.failed.dcsresults* and would be saved in the *C:\Test\FailedLogs* folder. The */D* setting *UseDateTimeInFailedFolder* disables the date and time subfolder creation under the failed logs folder.

Note: The double ending backslash used when specifying the folder for the */FAIL* switch is required for the command line path to be parsed correctly.

To overwrite an existing file the */O* flag would need to be added to the above command.

Convert a single file to a vector PDF document, save the output to a specific location and save the conversion results log file:

```
DCSConvertFile /S="C:\Output" /N="NewFileName" /L /O  
/P="Adobe PDF Multipage" "C:\Test\Document.doc"
```

Creates a PDF document from the source file C:\Test\Document.doc. The type of PDF created is controlled by the settings in the conversion profile *Adobe PDF Multipage.xml*. This profile creates vector PDF where possible.

The converted file will be named *NewFileName.pdf* and saved in the C:\Output folder. If a file of the same name already exists in C:\Output\ this file would be overwritten with the new file.

A conversion results log file, *Document.doc.succeeded.dcsresults* will also be created in the C:\Output folder.

If the conversion did not succeed, the results log would be named *Document.doc.failed.dcsresults* and a *.failed* folder would be created in the same location as the source document, C:\Test. The results log would be named *Document.doc.failed.dcsresults* and saved to a new subfolder under the *.failed* folder. The subfolder is named using the date and time of the conversion to keep subsequent runs separate.

Convert a single file to a PDF document, save both the output and any failed conversion results log file to custom locations :

```
DCSConvertFile /S="C:\Output" /N="NewFileName" /L /O  
/P="PDF A-1b 300dpi OptimizedColor"  
/FAIL="C:\FailedResults\\" "C:\Test\Document.doc"
```

Creates a PDF document from the source file C:\Test\Document.doc. The type of PDF created is controlled by the settings in the conversion profile *PDF A-1b 300dpi OptimizedColor.xml*.

The converted file will be named *NewFileName.pdf* and saved in the C:\Output folder. If a file of the same name already exists in C:\Output\ this file would be overwritten with the new file as specified by the /O argument.

A conversion results log file, *Document.doc.succeeded.dcsresults* will be created in the C:\FailedResults folder if the conversion does not succeed. The results log is named *Document.doc.failed.dcsresults* and placed in a subfolder named with the current date and time created under the specified folder.

Note: The double ending backslash used when specifying the folder for the /FAIL switch is required for the command line path to be parsed correctly.

Use the command line argument *D="UseDateTimeInFailedFolder:FALSE"* to store the results log file directly in the folder C:\FailedResults.

Command Line Arguments

Command line switches are not case-sensitive and can be entered in either upper or lower case.

/S - The Save Location

Pass in the full path to the folder in which to save the new file. If the save location is not specified the new file is created in the same folder as the source file.

- If the path includes spaces it must be enclosed in quotes.
- If the path doesn't exist, the conversion will fail.
- If a file of the same name already exists in the save file location, the conversion will fail.
The /O option can be used to enable file overwriting, which is off by default.

Example:

```
/S="C:\Converted Files\Test"
```

/N - Output File Name

The name to use for the output file. The default file extension for the type of file being created will always be added to the name provided here.

If this argument is not specified the name of the source file, including the extension is used. This prevents name collision when you have two different files with the same base name, such as abc.doc and abc.pdf. If you were converting to multipaged TIFF images the resulting converted files would be named abc.doc.tif and abc.pdf.tif.

If you do not want the original file name extension as part of your file name, use the /NE switch to remove the file extension.

If serialized files, such as JPEG images, are being created, the base name will be appended with the page number, SampleDocument_0001.jpg, SampleDocument_002.jpg, etc.

Example:

```
/N="SampleDocument_06_15_2012"
```

/O - Overwrite Always

Enables overwrite mode so that existing files of the same name are overwritten with the new file. When not specified the conversion will fail if a file of the same name already exists in the output folder.

/NE - No Extension

If you do not want the original file name extension as part of your output file name, use this switch to remove the file extension. If you have provided an output name with the /N switch above, this argument is ignored.

/L - Results Log

The results log file is an XML file containing a complete snapshot of the conversion information. Normally only saved for failed conversions, the /L argument enables creation of the results log file when the conversion succeeds.

The name of the results log file is based on the name of the original file and also indicates the conversion status. For example, when converting *Document.doc*, a successful conversion will create a log file named *Document.doc.succeeded.dcsresults.*, while a failed conversion would be named *Document.doc.failed.dcsresults.*

The results log file for a successful conversion is always copied to the output location with the converted files when this flag is used.

In the case of a failed conversion, the log file is always created. See the /FAIL switch to control the location and creation of the failed results log files.

The result log files can later be passed to the [DCSExtractResults](#) command line utility to extract information such as all files created or any errors encountered during conversion. You can search a folder for the results log files using the [DCSCreateFileList](#) utility.

/FAIL - Failed Results Log File Location

In the case of a failed conversion, the conversion results log file is always created. The default behavior is to create a *.failed* folder in the same location as the source file and save the conversion results log file to a new subfolder under the *.failed* folder. The subfolder is named using the date and time of the conversion to keep subsequent runs separate.

This argument allows you to override the default use of the *.failed* folder and to provide a specific folder in which to store the failed results log files. The name of the results log file is based on the name of the original file and also indicates the conversion status. For example, when converting *Document.doc*, a failed conversion would be named *Document.doc.failed.dcsresults.*

You can suppress the use of the date and time subfolder by passing the *UseDateTimeInFailedFolder* setting using the /D switch.

If you do not want to create the failed results log files at all, you can use the /D switch to pass the *KeepFailedItemResultsFiles* setting as *false*. These settings can also be added to any conversion profile you are using.

The result log files can later be passed to the [DCSExtractResults](#) command line utility to extract information such the source file used or any errors encountered during conversion. You can search a folder for the results log files using the [DCSCreateFileList](#) utility.

Note: The double ending backslash used when specifying the folder for the /FAIL switch is required for the command line path to be parsed correctly.

Examples:

```
/FAIL="C:\ConvertedFiles\Failed\\" /D="UseDateTimeInFailedFolder:FALSE"
```


/P - Conversion Profile

This is a required argument. The type of file created is controlled by supplying a conversion profile using this switch. The profiles are referenced by passing in the name of the profile XML file, with or without the XML extension. See [Creating and Customizing Profiles](#) for more information about the contents of the profiles, a list of profiles included with Document Conversion Service, and how to create your own.

Examples:

```
/P="TIFF 300dpi Color Fax"  
/P="TIFF 204x196dpi Monochrome Fax.xml"
```

/D - Define Setting

Individual conversion and profile settings can be supplied on the command line using this switch. This switch can be specified multiple times for separate settings and any settings passed here will override the settings in the profile.

Any name-value pair that can be written in a profile can be passed through this parameter. This includes options to control the conversion settings as well as the behavior of the individual converters as well. See [Creating and Customizing Profiles](#) for more information about the name-value pairs that can be used.

Examples:

These first two are settings that control the converter options, such as what pages to print, and the output that PowerPoint will print.

```
/D="PrintRange:1-5"  
/D="PowerPoint.PrintOptionsOutputType:PrintOutputNotesPages"
```

These two settings control the output file creation options, and would override or add to the settings in the conversion profile passed using the /P switch.

```
/D="Image Options;Fax Resolution:3"  
/D="TIFF File Format;BW compression:Group3-2D"
```

These two settings control the where the failed results log files are created and are most often used along with the /FAIL switch to control where the results log files are saved.

```
/D="KeepFailedItemResultsFiles:TRUE"  
/D="UseDateTimeInFailedFolder:FALSE"
```

/E - File Extension Mapping

A file extension mapping profile uses the extension of the source file to determine what converter will be used to convert the file. Like the conversion profiles, this file is also an XML file. This switch is optional and an internal default mapping is provided. You would only need to provide this file if you wanted to override the default file extension to converter mappings provided.

Examples:

```
/E="Custom Extension To Converter Map"
```

/W - Wait Time

Use this switch to wait to the specified number of seconds for the Document Conversion Service to be running and available to convert documents. If Document Conversion Service is already running the command executes immediately. If the Document Conversion Service is not running in the timeout period specified, the command will return with an error.

If this argument is not specified the command will return immediately with an error if Document Conversion Service is not running.

Example:

```
/W=300
```

/C - Convert on a Remote Computer (DCOM)

If Document Conversion Service is running on a different computer, use this switch to pass the name of the remote computer and the path of a shared location that both computers have access to. Separate the name of the remote computer and the path to the shared folder location with a semi-colon.

When converting remotely, the client redistributable, PNDocConvClientSetup_3.0.exe, must be installed on the computer running this command line utility. The client setup install program is included as part of the Document Conversion Service install and can be found in the **\Samples\Redist** folder in your product installation folder.

Examples:

```
/C="DOCCONV_SERVER;\\DOCCONV_SERVER\DCSREMOTE"
```

/SIL - Smart Inspect Logging File

Smart Inspect Log files are a tracing of the entire conversion process and are not the same as the conversion results log files created when a conversion fails. These logs can be viewed using the SmartInspect Redistributable Console included with Document Conversion Service.

These log files are automatically deleted when conversion succeeds. To keep the log files on success use the custom setting *AlwaysKeepProcessingLoggingFiles* as shown below.

The default location for this file is the TEMP folder. Each logging file is assigned a unique date, time and thread prefix followed by "_PNConvertFile.sil", such as *2014_09_11_2_38_00_PM_4_PNConvertFile.sil*.

Use this argument to specify a custom path and optional file name for the SmartInspect logging file (*.sil) created by this utility. The /SIL switch can take a folder, or a path to a filename. If a path without a trailing backslash is provided, the last part of the path is assumed to be a filename.

Note: The double ending backslash used when specifying a folder for the /SIL switch is required for the command line path to be parsed correctly.

/SIL=	Is interpreted as...
"C:\Test\LogFile"	Create the SmartInspect log file as C:\Test\LogFile.sil.
"C:\Test\LogFile\\"	Create the SmartInspect log file as C:\Test\LogFile\datetime_PNConvertFile.sil
"C:\Test\LogFile\ConvertFileCustom.sil"	Create the SmartInspect log file as C:\Test\LogFile\ConvertFileCustom.sil

The following settings can be used to control the creation and naming of the logging file. These settings are all passed using the /D switch.

Custom Setting	Description
RemoveDateTimePrefixOnProcessingLoggingFiles	Pass <i>True</i> to disable the adding of the unique date, time and thread prefix when a custom file name has not been specified in the <i>ConvertFileProcessLoggingPath</i> parameter.
KeepFailedProcessingLoggingFiles	Pass as <i>False</i> to disable the automatic creation of SmartInspect logging files when conversion fails. This setting can be overridden by <i>AlwaysKeepProcessingLoggingFiles</i> .
AlwaysKeepProcessingLoggingFiles	When set to <i>True</i> , the SmartInspect logging files are always created in the %TEMP% or other specified folder for both successful and failed conversions. If set to <i>False</i> , no logging files are created. This setting will override the <i>KeepFailedProcessingLoggingFiles</i> setting.

Examples:

Pass a custom folder and remove the prefix, each run will overwrite the log file C:\PEERNET\Logs\PNConvertFile.sil.

```
/SIL="C:\PEERNET\Logs\\" /D="RemoveDateTimePrefixOnProcessingLoggingFiles:TRUE"
```

Pass a custom folder and log file name and remove the prefix. Each run will overwrite the logging file C:\PEERNET\Logs\MyLogFile.sil.

```
/SIL="C:\PEERNET\Logs\MyLogFile" /D="RemoveDateTimePrefixOnProcessingLoggingFiles:TRUE"
```

Don't save any SmartInspect log files at all.

```
/D="AlwaysKeepProcessingLoggingFiles:FALSE"
```

/T - Alternate Temp Folder

This is an advanced setting that should not be needed in most cases. When converting a file, the conversion tool copies the file and performs the conversion in temporary *staging* and *working* folders created on demand in the default Windows temp folder. When dealing with long path and file names the default folders created can occasionally cause path names that are too long to process. When this happens this switch can be used to set the temporary folder to a shorter path to allow processing.

This setting is overridden if the /C option for remote conversion is being used with its own path to a shared location for conversion.

Examples:

```
/T="C:\PNTemp\\"
```

/? - Display Help

When passed as the only argument this switch will display help for this command.

Source File

The full path to the file to convert.

- If the path to the file includes spaces it must be enclosed in quotes.
- If the file doesn't exist, the conversion will fail.

DCSConvertFileList

A command line that accepts a text file containing a list of files to convert, or a list of files provided on the command line, and converts all files using Document Conversion Service. The Document Conversion Service must be running, either locally or on a remote computer for the file to be converted. If it is not running the command will return immediately with an error.

```
DCSConvertFileList /P=profile [/S=save location] [/O] [/NE] [/L]
                  [/E=extension map]
                  [/C=remote computer name;remote scratch folder]
                  [/D="name:value"] [/W=wait time]
                  [/FAIL=failed results log file location] [/SIL=conversion log file path]
                  [/I=input text file path]
                  [/T=alternate temp folder]
                  "file[;save location]" "file[;save location]"...
```

Sample Command Lines

Convert all files on command line to TIFF images:

```
DSCConvertFileList /P="TIFF 200dpi Monochrome.xml"  
"C:\Input\File1.doc" "C:\Input2\File2.doc"
```

Sends the files C:\Input\File1.doc and C:\Input\File2.doc to Document Conversion Service to be converted using the settings contained in the conversion profile *TIFF 200dpi Monochrome.xml*.

The converted files, *File1.doc.tif* and *File2.doc.tif*, will each be saved in the same location as their source file.

If a file with the same name already exists, that file conversion would fail. The results log file, named based on the source file and ending with *.doc.failed.dcsresults* would be placed in a folder named *.failed* created in the same location as the source document. This can be controlled with the */FAIL* switch.

To overwrite an existing file the */O* switch would need to be added to the above command. If you did not want the source file extension as part of your file name, the */NE* switch would need to be added.

Convert all files on command line to TIFF images in their own directory:

```
DSCConvertFileList /P="TIFF 200dpi Monochrome.xml" "C:\Input\File1.doc;C:\Output1"  
"C:\Input2\File2.doc;C:\Output2"
```

Sends the files C:\Input\File1.doc and C:\Input\File2.doc to Document Conversion Service to be converted using the settings contained in the conversion profile *TIFF 200dpi Monochrome.xml*.

The converted file, *File1.doc.tif* will be saved to the directory *C:\Output1* and *File2.doc.tif* will be saved in the directory *C:\Output2*.

If the output directory does not exist, or if a file with the same name already exists in either directory, that file conversion will fail. The results log file, named based on the source file and ending with *.doc.failed.dcsresults* would be placed in a folder named *.failed* created in the same location as each source document. This can be controlled with the */FAIL* switch.

To overwrite an existing file the */O* switch would need to be added to the above command. If you did not want the source file extension as part of your file name, the */NE* switch would need to be added.

Convert all files in the input file to TIFF images in a specific location:

```
DCSConvertFileList /P="TIFF 300dpi OptimizedColor.xml" /S="C:\Test\Output"
/I="C:\Test\Files.txt"
```

Sends the files listed in the text file *Files.txt* to Document Conversion Service to be converted using the settings contained in the conversion profile *TIFF 300dpi OptimizedColor.xml*.

Upon successful conversion each output file is placed under the C:\Test\Output folder.

If a file with the same name already exists, that file conversion would fail. The results log file, named based on the source file and ending with *.doc.failed.dcsresults* would be placed in a folder named *.failed* created in the same location as the source document.

Convert a list of files to vector PDF, strip off the source extension and save the output and results log files to a specific location:

```
DCSConvertFileList /P="Adobe PDF Multipage.xml" /NE /S="C:\Test\Output" /L
/FAIL="C:\Test\FailedLogs\\" /D="UseDateTimeInFailedFolder:FALSE"
/I="C:\Test\Files.txt"
```

Creates a PDF file from each file listed in the input file *Files.txt*. The PDF created is a vector PDF as controlled by the settings in the conversion profile *Adobe PDF Multipage.xml*.

Upon successful conversion each output file is placed under the C:\Test\Output folder along with the conversion results log file. The name of the results log file is based on the original source file and also indicates the conversion status. For example, if the source file name was *SampleDocument.doc*, a results log file, *SampleDocument.doc.succeeded.dcsresults*, will be created if the conversion succeeds.

The */NE* flag causes the output file to be named using the base name of the original file, plus the extension of the file type you are creating. If a file of that name already exists in the folder the conversion will fail. If the conversion fails a results log file named based on the source file and ending with *.failed.dcsresults* is placed into the folder *C:\Test\FailedLogs* specified by the */FAIL* parameter. The */D* setting *UseDateTimeInFailedFolder* disables the date and time subfolder creation under the failed logs folder.

Note: The double ending backslash used when specifying the folder for the */FAIL* switch is required for the command line path to be parsed correctly.

Command Line Arguments

Command line switches are not case-sensitive and can be entered in either upper or lower case.

/S - The Save Location

Pass in the full path to the folder in which to save the new files. If the save location is not specified the new file is created in the same folder as the source file. If the files listed in the input file text file specified with the /I switch also include save locations, those locations will be used instead.

- If the path includes spaces it must be enclosed in quotes.
 - If the path doesn't exist, the conversion will fail.
 - If a file of the same name already exists in the save file location, the conversion will fail.
- The /O option can be used to enable file overwriting, which is off by default.

Example:

```
/S="C:\Converted Files\Test"
```

/O - Overwrite Always

Enables overwrite mode so that existing files of the same name are overwritten with the new file. If the overwrite switch is not specified, the conversion of that file in the list of files will fail if a file of the same name already exists in the output folder.

/NE - No Extension

If you do not want the original file name extension as part of your output file name, use this switch to remove the file extension.

/L - Results Log

The results log file is an XML file containing a complete snapshot of the conversion information. Normally only saved for failed conversions, the /L argument enables creation of the results log file when the conversion succeeds. The results log file is placed in the same location as the converted files.

The name of the results log file is based on the name of the original file and also indicates the conversion status. For example, when converting *Document.doc*, a successful conversion will create a log file named *Document.doc.succeeded.dcsresults*, while a failed conversion would be named *Document.doc.failed.dcsresults*.

The results log file for a successful conversion is always copied to the output location with the converted files when this flag is used.

In the case of a failed conversion, the log file is always created. See the /FAIL switch to control the location and creation of the failed results log files.

The result log files can later be passed to the [DCSExtractResults](#) command line utility to extract information such as all files created or any errors encountered during conversion. You can search a folder for the results log files using the [DCSCreateFileList](#) utility.

/FAIL - Failed Results Log File Location

In the case of a failed conversion, the conversion results log file is always created. The default behavior is to create a *.failed* folder in the same location as the source file and save the conversion results log file to a new subfolder under the *.failed* folder. The subfolder is named using the date and time of the conversion to keep subsequent runs separate.

This argument allows you to override the default use of the *.failed* folder and to provide a specific folder in which to store the failed results log files. The name of the results log file is based on the name of the original file and also indicates the conversion status. For example, when converting *Document.doc*, a failed conversion would be named *Document.doc.failed.dcsresults*.

You can suppress the use of the date and time subfolder by passing the *UseDateTimeInFailedFolder* setting using the /D switch.

Note: The double ending backslash used when specifying the folder for the /FAIL switch is required for the command line path to be parsed correctly.

Examples:

```
/FAIL="C:\ConvertedFiles\Failed\\" /D="UseDateTimeInFailedFolder:FALSE"
```

If you do not want to create the failed results log files at all, you can use the /D switch to pass the *KeepFailedItemResultsFiles* setting as *false*.

On the command line:

```
/D="KeepFailedItemResultsFiles:False"
```

In a conversion profile:

```
<add Name="KeepFailedItemResultsFiles" Value="False"/>
```

The result log files can later be passed to the [DCSExtractResults](#) command line utility to extract information such the source file used or any errors encountered during conversion. You can search a folder for the results log files using the [DCSCreateFileList](#) utility.

/P - Conversion Profile

This is a required argument. The type of file created is controlled by supplying a conversion profile using this switch. The profiles are referenced by passing in the name of the profile XML file, with or without the XML extension. See [Creating and Customizing Profiles](#) for more information about the contents of the profiles, a list of profiles included with Document Conversion Service, and how to create your own.

Examples:

```
/P="TIFF 300dpi Color Fax"  
/P="TIFF 204x196dpi Monochrome Fax.xml"
```

/D - Define Setting

Individual conversion and profile settings can be supplied on the command line using this switch. This switch can be specified multiple times for separate settings and any settings passed here will override the settings in the profile.

Any name-value pair that can be written in a profile can be passed through this parameter. This includes options to control the conversion settings as well as the behavior of the individual converters as well. See [Creating and Customizing Profiles](#) for more information about the name-value pairs that can be used.

Examples:

These first two are settings that control the converter options, such as what pages to print, and the output that PowerPoint will print.

```
/D="PrintRange:1-5"
```

```
/D="PowerPoint.PrintOptionsOutputType:PrintOutputNotesPages"
```

These two settings control the output file creation options, and would override or add to the settings in the conversion profile passed using the /P switch.

```
/D="Image Options;Fax Resolution:3"
```

```
/D="TIFF File Format;BW compression:Group3-2D"
```

These two settings control where the failed results log files are created and are most often used along with the /FAIL switch to control where the results log files are saved.

```
/D="KeepFailedItemResultsFiles:TRUE"
```

```
/D="UseDateTimeInFailedFolder:FALSE"
```

/E - File Extension Mapping

A file extension mapping profile uses the extension of the source file to determine what converter will be used to convert the file. Like the conversion profiles, this file is also an XML file. This switch is optional and an internal default mapping is provided. You would only need to provide this file if you wanted to override the default file extension to converter mappings provided.

Examples:

```
/E="Custom Extension To Converter Map"
```

/W - Wait Time

Use this switch wait to the specified number of seconds for the Document Conversion Service to be running and available to convert documents. If Document Conversion Service is already running the command executes immediately. If the Document Conversion Service is not running in the timeout period specified, the command will return with an error.

If this argument is not specified the command will return immediately with an error if Document Conversion Service is not running.

Example:

```
/W=300
```

/SIL - Smart Inspect Logging File

Smart Inspect Log files are a tracing of the entire conversion process and are not the same as the conversion results log files created when a conversion fails. These logs can be viewed using the SmartInspect Redistributable Console included with Document Conversion Service.

These log files are automatically deleted when conversion succeeds. To keep the log files on success use the custom setting *AlwaysKeepProcessingLoggingFiles* as shown below.

The default location for this file is the TEMP folder. Each logging file is assigned a unique date, time and thread prefix followed by "_PNConvertFileList.sil", such as
`2014_09_11_2_38_00_PM_4_PNConvertFileList.sil`.

Use this argument to specify a custom path and optional file name for the SmartInspect logging file (*.sil) created by this utility. The /SIL switch can take a folder, or a path to a filename. If a path without a trailing backslash is provided, the last part of the path is assumed to be a filename.

Note: The double ending backslash used when specifying a folder for the /SIL switch is required for the command line path to be parsed correctly.

/SIL=	Is interpreted as...
"C:\Test\LogFile"	Create the SmartInspect log file as C:\Test\LogFile.sil.
"C:\Test\LogFile\\"	Create the SmartInspect log file as C:\Test\LogFile\datetime_PNConvertFileList.sil
"C:\Test\LogFile\ConvertFileCustom.sil"	Create the SmartInspect log file as C:\Test\LogFile\ConvertFileCustom.sil

The following settings can be used to control the creation and naming of the logging file. These settings are all passed using the /D switch.

Custom Setting	Description
RemoveDateTimePrefixOnProcessingLoggingFiles	Pass <i>True</i> to disable the adding of the unique date, time and thread prefix when a custom file name has not been specified in the <i>ConvertFileProcessLoggingPath</i> parameter.
KeepFailedProcessingLoggingFiles	Pass as <i>False</i> to disable the automatic creation of SmartInspect logging files when conversion fails. This setting can be overridden by <i>AlwaysKeepProcessingLoggingFiles</i> .
AlwaysKeepProcessingLoggingFiles	When set to <i>True</i> , the SmartInspect logging files are always created in the %TEMP% or other specified folder for both successful and failed conversions. If set to <i>False</i> , no logging files are created. This setting will override the <i>KeepFailedProcessingLoggingFiles</i> setting.

Examples:

Pass a custom folder and remove the prefix, each run will overwrite the log file C:\PEERNET\Logs\PNConvertFileList.sil.

```
/SIL="C:\PEERNET\Logs\" /D="RemoveDateTimePrefixOnProcessingLoggingFiles:TRUE"
```

Pass a custom folder and log file name and remove the prefix. Each run will overwrite the logging file C:\PEERNET\Logs\MyLogFile.sil.

```
/SIL="C:\PEERNET\Logs\MyLogFile" /D="RemoveDateTimePrefixOnProcessingLoggingFiles:TRUE"
```

Don't save any SmartInspect log files at all.

```
/D="AlwaysKeepProcessingLoggingFiles:FALSE"
```

/I - Input text file path

The collection of files to be converted can be passed as a text file containing a list of files, one each per line. Optionally you can specify individual save locations for each file by listing the file and directory, separated by a semi-colon(;) on each line. The full path or a UNC path to the source file and optional directory must be given for the files listed in the input text file and as command line arguments; relative paths are not supported.

The input text file should follow the following format:

```
C:\Input\WordFiles\File1.doc
C:\Input\WordFiles\File2.docx;C:\OutputPath\WordFiles\
C:\Input\PDF\File3.pdf;C:\OutputPath\PDFFiles\
\\server\share\input\scans\scan1.tif
```

/C - Convert on a Remote Computer (DCOM)

If Document Conversion Service is running on a different computer, use this switch to pass the name of the remote computer and the path of a shared location that both computers have access to. Separate the name of the remote computer and the path to the shared folder location with a semi-colon.

When converting remotely, the client redistributable, PNDocConvClientSetup_3.0.exe, must be installed on the computer running this command line utility. The client setup install program is included as part of the Document Conversion Service install and can be found in the **Samples\Redist** folder in your product installation folder.

Examples:

```
/C="DOCCONV_SERVER;\\DOCCONV_SERVER\DCSREMOTE"
```

/T - Alternate Temp Folder

This is an advanced setting that should not be needed in most cases. When converting files, the conversion tool copies each file and performs the conversion in temporary *staging* and *working* folders created on demand in the default Windows temp folder. When dealing with long path and file names the default folders created can occasionally cause path names that are too long to process. When this happens this switch can be used to set the temporary folder to a shorter path to allow processing.

This setting is overridden if the /C option for remote conversion is being used with its own path to a shared location for conversion.

Examples:

```
/T= "C:\PNTemp\\" "
```

/? - Display Help

When passed as the only argument this switch will display help for this command.

file[:save location]

The full path to the file to convert. You can list more than one on the command line. Like the input text file, you can pass in a semi-colon(;) separated file-directory pair here as well.

- If the path to the file includes spaces it must be enclosed in quotes.
- If the file doesn't exist, the conversion will fail.

DCSConvertFolder

A command line utility to walk a folder and convert all files, or all files matching a search filter, using Document Conversion Service. The utility can optionally also process all subfolders under the starting folder as well. The Document Conversion Service must be running, either locally or on a remote computer for the files to be converted. If it is not running the command will return immediately with an error.

Any sorting options applied only control the order in which the files are picked up from the directory. Sorting does not guarantee the order the files are processed in, only that files sorted to the top of the list are submitted for conversion first. A smaller file further down the list might finish before a larger file that was first in the list.

```
DCSConvertFolder /P=profile [/R] [/F=filter] [/X=exclude filter] [/S=save location]
[/O] [/NE] [/L] [/D="name:value"] [/E=extension map]
[/FAIL=failed results log file location] [/SIL=conversion log file path]
[/W=wait time] [/C=remote computer name;remote scratch folder]
[/T=alternate temp folder]
[/NAME=sort files by name] [/CREATED=sort files by creation date]
[/MODIFIED=sort files by name] [/DESC=sort files in descending order (Z-A, 9-0)]
folder
```

Sample Command Lines

Convert all files in a folder to TIFF images:

```
DCSConvertFolder /P="TIFF 200dpi Monochrome" "C:\Test\Input"
```

Sends all files in the folder C:\Test\Input to Document Conversion Service to be converted using the settings contained in the conversion profile *TIFF 200dpi Monochrome.xml*. Any folders under C:\Test\Input are not processed.

Upon successful conversion each output file is placed in a folder named *.converted* created under the C:\Test\Input folder. Each output file is named using the base name and file extension of the original file, plus the extension of the file type you are creating.

If a file of that name already exists in the *.converted* folder the conversion will fail and a *.failed* folder will be created under the C:\Test\Input folder. A results log file, ending with *.failed.dcsresults*, is created for each failed file and saved to a new subfolder under the *.failed* folder. The subfolder is named using the date and time of the conversion to keep subsequent runs separate.

Convert all files in a folder to TIFF images, wait up to 5 minutes for the conversion service to start:

```
DCSConvertFolder /P="TIFF 200dpi Monochrome" /W=300 "C:\Test\Input"
```

Sends all files in the folder C:\Test\Input to Document Conversion Service to be converted using the settings contained in the conversion profile *TIFF 200dpi Monochrome.xml*. Any folders under C:\Test\Input are not processed.

If Document Conversion Service is not running, wait up to 5 minutes (300 seconds) for the conversion service to be available.

Convert all files in a folder, including subfolders, to TIFF images in a specific location:

```
DCSConvertFolder /R /P="TIFF 300dpi OptimizedColor" /S="C:\Test\Output"
"C:\Test\Input"
```

Walks the folder C:\Test\Input and any folders underneath and sends all the files found to Document Conversion Service to be converted using the settings contained in the conversion profile *TIFF 300dpi OptimizedColor.xml*.

Upon successful conversion each output file is placed in the C:\Test\Output folder in a directory structure that mirrors the source folder directory structure.

If a file does not convert, a subfolder named *.failed* is created in the same location as the input file. A results log named by appending *.failed.dcsresults* to the input file name is created and saved to a new subfolder under the *.failed* folder. The new subfolder is named using the date and time of the conversion to keep subsequent runs separate.

To store all of the failed file information in a separate location, see the */FAIL* option.

Convert all Word and Excel files in the folder, including subfolders, to vector PDF documents in a specific location:

```
DCSConvertFolder /R /F="*.doc|*.docx|*.xls|*.xlsx" /X="12-01*" /S="C:\Test\Output\
/P="C:\Test\Adobe PDF Multipage.xml" "C:\Test\Input"
```

Walks the folder C:\Test\Input and any folders underneath and sends all Word files ending in *.doc* and *.docx* and all Excel files ending in *.xls* and *.xlsx* to Document Conversion Service to be converted to vector PDF using the settings contained in the conversion profile *Adobe PDF Multipage.xml*. Any files or folders that begin with "12-01" are excluded.

Upon successful conversion each output file is placed under the C:\Test\Output folder in a directory structure that mirrors the source folder directory structure.

Failed conversion results logs are saved in a *.failed* folder created in the same location as the source file. A results log named by appending *.failed.dcsresults* to the input file name is created and saved to a new subfolder under the *.failed* folder. The new subfolder is named using the date and time of the conversion to keep subsequent runs separate.

Convert a folder of documents to vector PDF, overwrite any existing files, and save the results log:

```
DCSConvertFolder /P="Adobe PDF Multipage" /O /L "C:\Test\Input"
```

Sends all files in the folder C:\Test\Input to Document Conversion Service to be converted to vector PDF using the settings contained in the conversion profile *Adobe PDF Multipage.xml*. Any folders under C:\Test\Input are not processed.

Upon successful conversion each output file is placed in a folder named *.converted* created under the C:\Test\Input folder. Any files of the same name that already exist in that folder are overwritten.

A conversion results log file for each file converted will be also be saved in the *.converted* folder. The name of the results log file is based on the name of the original source file appended with *.succeeded.dcsresults*.

Failed conversion results logs are saved in a *.failed* folder created in the same location as the source file. A results log named by appending *.failed.dcsresults* to the input file name is created and saved to a new subfolder under the *.failed* folder. The new subfolder is named using the date and time of the conversion to keep subsequent runs separate.

Convert a folder of documents to vector PDF, strip off the source extension and save the output and results log files to a specific location:

```
DCSConvertFolder /R /P="Adobe PDF Multipage" /NE /S="C:\Test\Output"  
/L /FAIL="C:\Test\FailedResults\\" "C:\Test\Input"
```

Walks the folder C:\Test\Input and any folders underneath and creates vector PDF files from all documents found. The type of PDF created is controlled by the settings in the conversion profile *Adobe PDF Multipage.xml*.

Upon successful conversion each output file is placed under the C:\Test\Output folder in a directory structure that mirrors the source folder directory structure. The /NE flag causes the output file to be named using the base name of the original file, plus the extension of the file type you are creating. If a file of that name already exists in the folder the conversion will fail.

A conversion results log file for each file will be also be saved in the C:\Test\Output folder in the same mirrored directory structure. The name of the results log file is based on the original source file and appended with *.succeeded.dcsresults* to indicate its conversion status.

If the conversion did not succeed, the results log is named by appending *.failed.dcsresults* to the input file name and placing this file into a subfolder named with the current date and time created under the specified folder *C:\Test\FailedResults*.

Note: The double ending backslash used when specifying the folder for the /FAIL switch is required for the command line path to be parsed correctly.

Use the command line argument *D="UseDateTimeInFailedFolder:FALSE"* to store the results log file directly in the folder *C:\FailedResults*.

Controlling the Number of Documents Processed in Parallel

When converting a folder of files, the number of documents that can be passed in parallel (*at the same time*) to Document Conversion Service to be converted is automatically determined based on the number of CPU's and cores on your system multiplied by 1.5. We recommend that you allow this value to be determined automatically, but if needed, you can specify exactly how many documents you want to process in parallel by adding the following line into the profile you are using.

Please note that this value is completely separate from the value of the same name used by the Document Conversion Service configuration. Also, keep in mind that setting this to a value that is too high for the capabilities of the computer can cause the computer to work very slowly.

```
<add Name="NumberOfDocumentsInParallel" Value="10" />
```

You can also pass this directly on the command line using the /D option.

```
DSCConvertFolder /P="TIFF 200dpi Monochrome.xml" /D="NumberOfDocumentsInParallel:6"  
"C:\Test\Input"
```

Command Line Arguments

Command line switches are not case-sensitive and can be entered in either upper or lower case.

/R - Include Subfolders (Recurse)

Use this switch to also search any subfolders under the source folder when building the list of files to be passed to Document Conversion Service to be converted.

/F - File Filter

A filter can be provided using this switch to only process certain types of files. Multiple file filters can be combined using the pipe (|) character. Hidden and system files are ignored, and the search pattern filters files based on a regular expression match of the long name of a file.

When this switch is not specified all files in the folder are (*.*) passed to Document Conversion Service to be processed.

Examples:

Convert PDF only: `/F="*.pdf"`
Convert Word, Excel and PDF only: `/F="*.doc|*.docx|*.xls|*.xlsx|*.pdf"`
Convert all Word files starting with MEMO: `/F="MEMO*.doc"`

/X - Exclude File Filter

A exclude file filter can be provided to take the returned file list gathered using the /F file filter and exclude any files that match a pattern. Multiple patterns can be combined using the pipe (|) character. By default no files are excluded.

Examples:

Exclude Word and Excel 2010 documents: `/X="*.docx|*.xlsx"`
Exclude all files starting with "Draft": `/X="Draft*.*"`

/S - The Save Location

Pass in the full path to the folder in which to save the new files. If the /R switch is used the original directory structure is maintained.

- If the path includes spaces it must be enclosed in quotes.
 - If the path is specified but doesn't exist, the conversion will fail.
 - If a file of the same name already exists in the save file location, the conversion will fail.
- The /O option can be used to enable file overwriting, which is off by default.

If this argument is not specified, a *.converted* folder is created in the same location as each source file and all output files are saved there. On subsequent processing of the same folder with the /R switch enabled, any *.converted* folders are ignored.

Example:

`/S="C:\Converted Files\Test"`

/O - Overwrite Always

Enables overwrite mode so that existing files of the same name are overwritten with the new file. If the overwrite switch is not specified, the conversion of that file in the list of files will fail if a file of the same name already exists in the output folder.

/NE - No Extension

Specify this option if you do not want the original file name extension as part of your output file name. Normally the name of the each output file is created using the base name and file extension of the original file to prevent name collision when you have two files in the folder with the same base name.

/L - Results Log

The results log file is an XML file containing a complete snapshot of the conversion information for each file converted. Normally only saved for failed conversions, the /L argument enables creation of the results log file when the conversion succeeds. The results log file is placed in the same location as the converted files.

The name of the results log file is based on the name of the original file and also indicates the conversion status. For example, when converting *Document.doc*, a successful conversion will create a log file named *Document.doc.succeeded.dcsresults*, while a failed conversion would be named *Document.doc.failed.dcsresults*.

The results log file for a successful conversion is always copied to the output location with the converted files when this flag is used.

In the case of a failed conversion, the log file is always created. See the /FAIL switch to control the location and creation of the failed results log files.

The result log files can later be passed to the [DCSEExtractResults](#) command line utility to extract information such as all files created or any errors encountered during conversion. You can search a folder for the results log files using the [DCSCreateFileList](#) utility.

/FAIL - Failed Results Log File Location

In the case of a failed conversion, the conversion results log file is always created. The default behavior is to create a *.failed* folder in the same location as the source file and save the conversion results log file to a new subfolder under the *.failed* folder. The subfolder is named using the date and time of the conversion to keep subsequent runs separate.

This argument allows you to override the default use of the *.failed* folder and to provide a specific folder in which to store the failed results log files. The name of the results log file is based on the name of the original file and also indicates the conversion status. For example, when converting *Document.doc*, a failed conversion would be named *Document.doc.failed.dcsresults*.

You can suppress the use of the date and time subfolder by passing the *UseDateTimeInFailedFolder* setting using the /D switch.

Note: The double ending backslash used when specifying the folder for the /FAIL switch is required for the command line path to be parsed correctly.

Examples:

```
/FAIL="C:\ConvertedFiles\Failed\\" /D="UseDateTimeInFailedFolder:FALSE"
```

If you do not want to create the failed results log files at all, you can use the /D switch to pass the *KeepFailedItemResultsFiles* setting as *false*.

On the command line:

```
/D="KeepFailedItemResultsFiles:False"
```

In a conversion profile:

```
<add Name="KeepFailedItemResultsFiles" Value="False"/>
```

The result log files can later be passed to the [DCSExtractResults](#) command line utility to extract information such the source file used or any errors encountered during conversion. You can search a folder for the results log files using the [DCSCreateFileList](#) utility.

/P - Conversion Profile

This is a required argument. The type of file created is controlled by supplying a conversion profile using this switch. The profiles are referenced by passing in the name of the profile XML file, with or without the XML extension. See [Creating and Customizing Profiles](#) for more information about the contents of the profiles, a list of profiles included with Document Conversion Service, and how to create your own.

Examples:

```
/P="TIFF 300dpi Color Fax"  
/P="TIFF 204x196dpi Monochrome Fax.xml"
```

/D - Define Setting

Individual profile settings can be supplied on the command line using this switch. This switch can be specified multiple times for separate settings and any settings passed here will override the settings in the profile.

Any name-value pair that can be written in a profile can be passed through this parameter. This includes options to control the conversion settings as well as the behavior of the individual converters as well. See [Creating and Customizing Profiles](#) for more information about the name-value pairs that can be used.

Examples:

These first two are settings that control the converter options, such as what pages to print, and the output that PowerPoint will print.

```
/D="PrintRange:1-5"
```

```
/D="PowerPoint.PrintOptionsOutputType:PrintOutputNotesPages"
```

These two settings control the output file creation options, and would override or add to the settings in the conversion profile passed using the /P switch.

```
/D="Image Options;Fax Resolution:3"
```

```
/D="TIFF File Format;BW compression:Group3-2D"
```

These two settings control where the failed results log files are created and are most often used along with the /FAIL switch to control where the results log files are saved.

```
/D="KeepFailedItemResultsFiles:TRUE"
```

```
/D="UseDateTimeInFailedFolder:FALSE"
```

/E - File Extension Mapping

A file extension mapping profile uses the extension of the source file to determine what converter will be used to convert the file. Like the conversion profiles, this file is also an XML file. This switch is optional and an internal default mapping is provided. You would only need to provide this file if you wanted to override the default file extension to converter mappings provided.

Examples:

```
/E="Custom Extension To Converter Map"
```

/W - Wait Time

Use this switch wait to the specified number of seconds for the Document Conversion Service to be running and available to convert documents. If Document Conversion Service is already running the command executes immediately. If the Document Conversion Service is not running in the timeout period specified, the command will return with an error.

If this argument is not specified the command will return immediately with an error if Document Conversion Service is not running.

Example:

```
/W=300
```

/C - Convert on a Remote Computer (DCOM)

If Document Conversion Service is running on a different computer, use this switch to pass the name of the remote computer and the path of a shared location that both computers have access to. Separate the name of the remote computer and the path to the shared folder location with a semi-colon.

When converting remotely, the client redistributable, PNDocConvClientSetup_3.0.exe, must be installed on the computer running this command line utility. The client setup install program is included as part of the Document Conversion Service install and can be found in the **\Samples\Redist** folder in your product installation folder.

Examples:

```
/C="DOCCONV_SERVER;\\DOCCONV_SERVER\DCSREMOTE"
```

/SIL - Smart Inspect Logging File

Smart Inspect Log files are a tracing of the entire conversion process and are not the same as the conversion results log files created when a conversion fails. These logs can be viewed using the SmartInspect Redistributable Console included with Document Conversion Service.

These log files are automatically deleted when conversion succeeds. To keep the log files on success use the custom setting *AlwaysKeepProcessingLoggingFiles* as shown below.

The default location for this file is the TEMP folder. Each logging file is assigned a unique date, time and thread prefix followed by "_PNConvertFolder.sil", such as *2014_09_11_2_38_00_PM_4_PNConvertFolder.sil*.

Use this argument to specify a custom path and optional file name for the SmartInspect logging file (*.sil) created by this utility. The /SIL switch can take a folder, or a path to a filename. If a path without a trailing backslash is provided, the last part of the path is assumed to be a filename.

Note: The double ending backslash used when specifying a folder for the /SIL switch is required for the command line path to be parsed correctly.

/SIL=	Is interpreted as...
"C:\Test\LogFile"	Create the SmartInspect log file as C:\Test\LogFile.sil.
"C:\Test\LogFile\\"	Create the SmartInspect log file as C:\Test\LogFile\datetime_PNConvertFolder.sil
"C:\Test\LogFile\ConvertFileCustom.sil"	Create the SmartInspect log file as C:\Test\LogFile\ConvertFileCustom.sil

The following settings can be used to control the creation and naming of the logging file. These settings are all passed using the /D switch.

Custom Setting	Description
RemoveDateTimePrefixOnProcessingLoggingFiles	Pass <i>True</i> to disable the adding of the unique date, time and thread prefix when a custom file name has not been specified in the <i>ConvertFileProcessLoggingPath</i> parameter.
KeepFailedProcessingLoggingFiles	Pass as <i>False</i> to disable the automatic creation of SmartInspect logging files when conversion fails. This setting can be overridden by <i>AlwaysKeepProcessingLoggingFiles</i> .
AlwaysKeepProcessingLoggingFiles	When set to <i>True</i> , the SmartInspect logging files are always created in the %TEMP% or other specified folder for both successful and failed conversions. If set to <i>False</i> , no logging files are created. This setting will override the <i>KeepFailedProcessingLoggingFiles</i> setting.

Examples:

Pass a custom folder and remove the prefix, each run will overwrite the log file C:\PEERNET\Logs\PNConvertFolder.sil.

```
/SIL="C:\PEERNET\Logs\\" /D="RemoveDateTimePrefixOnProcessingLoggingFiles:TRUE"
```

Pass a custom folder and log file name and remove the prefix. Each run will overwrite the logging file C:\PEERNET\Logs\MyLogFile.sil.

```
/SIL="C:\PEERNET\Logs\MyLogFile" /D="RemoveDateTimePrefixOnProcessingLoggingFiles:TRUE"
```

Don't save any SmartInspect log files at all.

```
/D="AlwaysKeepProcessingLoggingFiles:FALSE"
```

/T - Alternate Temp Folder

This is an advanced setting that should not be needed in most cases. When converting files, the conversion tool copies each file and performs the conversion in temporary *staging* and *working* folders created on demand in the default Windows temp folder. When dealing with long path and file names the default folders created can occasionally cause path names that are too long to process. When this happens this switch can be used to set the temporary folder to a shorter path to allow processing.

This setting is overridden if the /C option for remote conversion is being used with its own path to a shared location for conversion.

Examples:

```
/T="C:\PNTemp\\"
```

/NAME - Sort Files by Name

The file list picked up from the folder is sorted by file name in ascending order (0-9, A-Z).

/CREATED - Sort Files by Creation Date

The file list picked up from the folder is sorted by the files creation date file name in ascending order (0-9, A-Z).

/MODIFIED - Sort Files by Modified Date

The file list picked up from the folder is sorted by the files last modified date file name in ascending order (0-9, A-Z).

/DESC - Sort Files in Descending Order

Use with /NAME, /CREATED or /MODIFIED. Sorts the descending order (Z-A, 9-0).

/? - Display Help

When passed as the only argument this switch will display help for this command.

Folder

The folder containing the files to convert.

- If the path to the folder includes spaces it must be enclosed in quotes.
- If the folder doesn't exist, the conversion will fail.

DCSCombineFiles

A command line utility that accepts a text file containing a list of files, and/or a list of files provided on the command line, and combines all of the files together into a single file or a collection of serialized pages using Document Conversion Service. The files are appended together in the order in which they are received.

The Document Conversion Service must be running, either locally or on a remote computer for the files to be combined. If it is not running the command will return immediately with an error.

```
DCSCombineFiles /P=profile /S=save location /N=output name
                [/O] [/L] [/E=extension map]
                [/C=remote computer name;remote scratch folder]
                [/FAIL=failed results log file location] [/SIL=conversion log file path]
                [/D="name:value"] [/W=wait time]
                [/I=input text file path]
                [/T=alternate temp folder]
                "file" "file" ...
```

Sample Command Lines

Combine all files on command line into a single TIFF image:

```
DCSCombineFiles /S="C:\Test\Output" /N="CombinedFiles" /P="TIFF 200dpi Monochrome"
                "C:\Input\File1.doc" "C:\Input2\File2.doc"
```

Sends the files C:\Input\File1.doc and C:\Input2\File2.doc to Document Conversion Service to be converted using the settings contained in the conversion profile *TIFF 200dpi Monochrome*.

The converted files are saved as a single TIFF image, named *CombinedFiles.tif* in the output folder C:\Test\Output. The files are combined in the order in which they are returned by the underlying file system.

If an output file with the same name already exists, or if one of the files in the combine set fails to convert, the combine will fail and a results log file will be placed in a folder named *.failed* created in the save location. The results log file name will be *PNCombineFiles_*, followed by a date and time stamp and ending with *failed.dcsresults*. This can be controlled with the */FAIL* switch.

To overwrite an existing file the */O* switch would need to be added to the above command.

Convert all files in the input file to a multipage PDF:

```
DCSCombineFiles /S="C:\Test\Output" /N="CombinedFiles"
                /P="PDF 300dpi OptimizedColor"
                /I="C:\Test\Files.txt"
```

Sends the files listed in the text file *C:\Test\Files.txt* to Document Conversion Service to be converted using the settings contained in the conversion profile *PDF 300dpi OptimizedColor*.

The converted files are saved as a multipaged PDF file named *CombinedFiles.pdf* in the output folder *C:\Test\Output*. The files are combined in the order they are listed in the input file.

Upon successful conversion each output file is placed under the *C:\Test\Output* folder.

If a file with the same name already exists, or if one of the files in the combine set fails to convert, the conversion will fail and a results log file will be placed in a folder named *.failed* created in the save location. Where this file is saved can be controlled with the */FAIL* switch.

The results log file name starts with *PNCombineFiles*, contains a date and time stamp and ends with *.failed.dcsresults*.

Convert all files in the input file and command line to a multipage PDF in that order:

```
DCSCombineFiles /S="C:\Test\Output" /N="CombinedFiles"
                /P="PDF 300dpi OptimizedColor" /I="C:\Test\Files.txt"
                "C:\Test\EndOfCombine.doc"
```

Sends the files listed in the text file *"C:\Test\Files.txt"*, followed by the file *"C:\Test\EndOfCombine.doc"* specified on the command line, in that order, to Document Conversion Service to be converted using the settings contained in the conversion profile *PDF 300dpi OptimizedColor*.

The converted files are saved as a multipaged PDF file named *CombinedFiles.pdf* in the output folder *C:\Test\Output*. The files are combined in the order they are listed in the input file.

Upon successful conversion each output file is placed under the *C:\Test\Output* folder.

If a file with the same name already exists, or if one of the files in the combine set fails to convert, the combine will fail and a results log file will be placed in a folder named *.failed* created in the save location. Where this file is saved can be controlled with the */FAIL* switch.

The results log file name starts with *PNCombineFiles*, contains a date and time stamp and ends with *.failed.dcsresults*.

Combine all files on command line into a single TIFF image, save conversion results logs to a specific location:

```
DSCCombineFiles /L /S="C:\Test\Output" /N="CombinedFiles"  
/P="TIFF 200dpi Monochrome" /FAIL="C:\Test\Output\Failed\\"  
"C:\Input\File1.doc" "C:\Input2\File2.doc"
```

Sends the files C:\Input\File1.doc and C:\Input\File2.doc to Document Conversion Service to be converted using the settings contained in the conversion profile *TIFF 200dpi Monochrome*.

The converted files are saved as single TIFF image, named *CombinedFiles.tif* in the output folder C:\Test\Output. The files are combined in the order given on the command line - File1.doc followed by File2.doc.

If a file with the same name already exists, or if one of the files in the combine set fails to convert, the combine will fail and a conversion results log file will be placed into the folder C:\Test\Output\Failed.

The conversion results log file name starts with *PNCombineFiles*, contains a date and time stamp and ends with *failed.dcsresults*. You can use the /D parameter *UseDateTimeInFailedFolder* to remove the date and time stamp from the results log file name.

To overwrite an existing file the /O switch would need to be added to the above command.

Command Line Arguments

Command line switches are not case-sensitive and can be entered in either upper or lower case.

/S - The Save Location

This is a required argument. Pass in the full path to the folder in which to save the new files.

- If the path includes spaces it must be enclosed in quotes.
- If the path doesn't exist, the conversion will fail.
- If a file of the same name already exists in the save file location, the conversion will fail. To enable file overwriting, use the /O option.

Example:

```
/S="C:\Converted Files\Test"
```

/N - Output File Name

This is a required argument and specifies the name to use for the output file. The default file extension for the type of file being created will always be added to the name provided here.

Example:

```
/N="CombinedOutput_06_15_2012"
```

/O - Overwrite Always

Enables overwrite mode so that existing files of the same name are overwritten with the new file. If overwrite is not specified the combine action will fail if a file of the same name already exists in the save location.

/L - Results Log

The results log file is an XML file containing a complete snapshot of the combine request. Normally only saved for failed conversions, the /L argument enables creation of the results log file when the conversion succeeds.

All results log files for this command line utility start with *PNCombineFiles_*, contain a date and time stamp and end with the conversion status.

When the combine has succeeded, the results log file is placed in the same folder as the output (specified using the /S switch) and would have a name similar to the following:

```
PNCombineFiles_2013_05_31_2_50_05_PM_3.succeeded.dcsresults
```

The bold text in the name will change for each file and is based on the date and time of the run and an internal counter. You can suppress the use of the date and time information in the file name by passing *false* for the *UseDateTimeInFailedFolder* setting using the /D switch.

In the case of a failed conversion, the log file is always created. See the /FAIL switch to control the location and creation of the failed results log files.

The result log files can later be passed to the [DCSExtractResults](#) command line utility to extract information such as all files created or any errors encountered during conversion. You can search a folder for the results log files using the [DCSCreateFileList](#) utility.

/FAIL - Combine Results Log File Location

In the case of a failed combine, the combine results log file is always created. When the combine does not succeed, a *.failed* folder is created in the save folder location (provided by the /S switch) and the results log files are stored there.

The name of the results log when the combine does not succeed will be similar to the following:

```
PNCombineFiles_2013_05_31_2_50_05_PM_3.failed.dcsresults
```

The bold text in the name will change each time a combine command is run and is based on the date and time of the run and an internal counter.

This argument allows you to override the default use of the *.failed* folder and to provide a specific folder in which to store the failed results log file. You can suppress the use of the date and time information in the file name by passing *false* for the *UseDateTimeInFailedFolder* setting using the /D switch.

Note: The double ending backslash used when specifying the folder for the /FAIL switch is required for the command line path to be parsed correctly.

Examples:

```
/FAIL="C:\ConvertedFiles\Failed\\" /D="UseDateTimeInFailedFolder:FALSE"
```

If you do not want to create the failed results log files at all, you can use the /D switch to pass the *KeepFailedItemResultsFiles* setting as *false*.

On the command line:

```
/D="KeepFailedItemResultsFiles:False"
```

In a conversion profile:

```
<add Name="KeepFailedItemResultsFiles" Value="False"/>
```

The result log files can later be passed to the [DCSExtractResults](#) command line utility to extract information such the source file used or any errors encountered during conversion. You can search a folder for the results log files using the [DCSCreateFileList](#) utility.

/P - Conversion Profile

This argument is required. The type of file created is controlled by supplying a conversion profile using this switch. The profiles are referenced by passing in the name of the profile XML file, with or without the XML extension. See [Creating and Customizing Profiles](#) for more information about the contents of the profiles, a list of profiles included with Document Conversion Service, and how to create your own.

Examples:

```
/P="TIFF 300dpi Color Fax"  
/P="TIFF 204x196dpi Monochrome Fax.xml"
```

/D - Define Setting

Individual profile settings can be supplied on the command line using this switch. This switch can be specified multiple times for separate settings and any settings passed here will override the settings in the profile.

Any name-value pair that can be written in a profile can be passed through this parameter. This includes options to control the conversion settings as well as the behavior of the individual converters as well. See [Creating and Customizing Profiles](#) for more information about the name-value pairs that can be used.

Examples:

These first two are settings that control the converter options, such as what pages to print, and the output that PowerPoint will print.

```
/D="PrintRange:1-5"
```

```
/D="PowerPoint.PrintOptionsOutputType:PrintOutputNotesPages"
```

These two settings control the output file creation options, and would override or add to the settings in the conversion profile passed using the /P switch.

```
/D="Image Options;Fax Resolution:3"
```

```
/D="TIFF File Format;BW compression:Group3-2D"
```

These two settings control the where the failed results log files are created and are most often used along with the /FAIL switch to control where the results log files are saved.

```
/D="KeepFailedItemResultsFiles:TRUE"
```

```
/D="UseDateTimeInFailedFolder:FALSE"
```

/E - File Extension Mapping

A file extension mapping profile uses the extension of the source file to determine what converter will be used to convert the file before combining them together. Like the conversion profiles, this file is also an XML file. This switch is optional and an internal default mapping is provided. You would only need to provide this file if you wanted to override the default file extension to converter mappings provided.

Examples:

```
/E="Custom Extension To Converter Map"
```

/W - Wait Time

Use this switch wait to the specified number of seconds for the Document Conversion Service to be running and available to convert and combine documents. If Document Conversion Service is already running the command executes immediately. If the Document Conversion Service is not running in the timeout period specified, the command will return with an error.

If this argument is not specified the command will return immediately with an error if Document Conversion Service is not running.

Example:

```
/W=300
```


/I - Input text file path

The collection of files to be combined can be passed as a text file containing a list of files, one each per line. The full path or a UNC path to the source file must be given for the files listed in the input text file; relative paths are not supported.

- If the path to the file includes spaces it must be enclosed in quotes.
- If the file doesn't exist, the conversion will fail.

The files are combined together in the order in which they are listed in the folder. Any files were specified directly on the command line before this switch are combined before adding the files in the input text file. Any source files specified on the command line after this switch are combined after the files in the input text file.

The input text file should follow the following format:

```
C:\Input\WordFiles\File1.doc  
C:\Input\WordFiles\File2.docx;C:\OutputPath\WordFiles\  
C:\Input\PDF\File3.pdf;C:\OutputPath\PDFFiles\  
\\server\share\Input\scans\scan1.tif
```

/C - Convert on a Remote Computer (DCOM)

If Document Conversion Service is running on a different computer, use this switch to pass the name of the remote computer and the path of a shared location that both computers have access to. Separate the name of the remote computer and the path to the shared folder location with a semi-colon.

When combining remotely, the client redistributable, PNDocConvClientSetup_3.0.exe, must be installed on the computer running this command line utility. The client setup install program is included as part of the Document Conversion Service install and can be found in the **\Samples\Redist** folder in your product installation folder.

Examples:

```
/C="DOCCONV_SERVER;\\DOCCONV_SERVER\DCSREMOTE"
```

/SIL - Smart Inspect Logging File

Smart Inspect Log files are a tracing of the entire conversion process and are not the same as the conversion results log files created when a conversion fails. These logs can be viewed using the SmartInspect Redistributable Console included with Document Conversion Service.

These log files are automatically deleted when conversion succeeds. To keep the log files on success use the custom setting *AlwaysKeepProcessingLoggingFiles* as shown below.

The default location for this file is the TEMP folder. Each logging file is assigned a unique date, time and thread prefix followed by "_PNCombineFiles.sil", such as
`2014_09_11_2_38_00_PM_4_PNCombineFiles.sil`.

Use this argument to specify a custom path and optional file name for the SmartInspect logging file (*.sil) created by this utility. The /SIL switch can take a folder, or a path to a filename. If a path without a trailing backslash is provided, the last part of the path is assumed to be a filename.

Note: The double ending backslash used when specifying a folder for the /SIL switch is required for the command line path to be parsed correctly.

/SIL=	Is interpreted as...
"C:\Test\LogFile"	Create the SmartInspect log file as C:\Test\LogFile.sil.
"C:\Test\LogFile\\"	Create the SmartInspect log file as C:\Test\LogFile\datetime_PNCombineFiles.sil
"C:\Test\LogFile\ConvertFileCustom.sil"	Create the SmartInspect log file as C:\Test\LogFile\ConvertFileCustom.sil

The following settings can be used to control the creation and naming of the logging file. These settings are all passed using the /D switch.

Custom Setting	Description
RemoveDateTimePrefixOnProcessingLoggingFiles	Pass <i>True</i> to disable the adding of the unique date, time and thread prefix when a custom file name has not been specified in the <i>ConvertFileProcessLoggingPath</i> parameter.
KeepFailedProcessingLoggingFiles	Pass as <i>False</i> to disable the automatic creation of SmartInspect logging files when conversion fails. This setting can be overridden by <i>AlwaysKeepProcessingLoggingFiles</i> .
AlwaysKeepProcessingLoggingFiles	When set to <i>True</i> , the SmartInspect logging files are always created in the %TEMP% or other specified folder for both successful and failed conversions. If set to <i>False</i> , no logging files are created. This setting will override the <i>KeepFailedProcessingLoggingFiles</i> setting.

Examples:

Pass a custom folder and remove the prefix, each run will overwrite the log file C:\PEERNET\Logs\PNCombineFiles.sil.

```
/SIL="C:\PEERNET\Logs\\" /D="RemoveDateTimePrefixOnProcessingLoggingFiles:TRUE"
```

Pass a custom folder and log file name and remove the prefix. Each run will overwrite the logging file C:\PEERNET\Logs\MyLogFile.sil.

```
/SIL="C:\PEERNET\Logs\MyLogFile" /D="RemoveDateTimePrefixOnProcessingLoggingFiles:TRUE"
```

Don't save any SmartInspect log files at all.

```
/D="AlwaysKeepProcessingLoggingFiles:FALSE"
```

/T - Alternate Temp Folder

This is an advanced setting that should not be needed in most cases. When converting files, the conversion tool copies each file and performs the conversion in temporary *staging* and *working* folders created on demand in the default Windows temp folder. When dealing with long path and file names the default folders created can occasionally cause path names that are too long to process. When this happens this switch can be used to set the temporary folder to a shorter path to allow processing.

This setting is overridden if the /C option for remote conversion is being used with its own path to a shared location for conversion.

Examples:

```
/T="C:\PNTemp\\"
```

/? - Display Help

When passed as the only argument this switch will display help for this command.

File

The full path to the files to combine. You can list more than one on the command line.

The files are combined together in the order in which they listed on the command line. If any files were specified in an input text file using the /I switch before these files, the files listed in the input text file are combined before adding the files from the command line. If you specify the input text file after the files on the command line, the command line files are combined first, then the files listed in the input text file.

- If the path to the file includes spaces it must be enclosed in quotes.
- If the file doesn't exist, the conversion will fail.

DCSCombineFolder

A command line utility to walk a folder and combines all files, or all files matching a search filter, into a single file or a collection of serialized pages using Document Conversion Service. The utility can optionally also process all subfolders under the starting folder as well.

The order of the files in the combined file cannot be guaranteed and is dependent on the file system. Most cases they are alphabetical but can also be by creation time. Files from the root of the input folder are listed first, then all files from the subfolders when enabled. Subfolders are listed in alphabetical or creation time order, again dependent on the file system.

The Document Conversion Service must be running, either locally or on a remote computer for the files to be combined. If it is not running the command will return immediately with an error.

```
DCSCombineFolder /P=profile /S=save location /N=output name
                  [/R] [/F=filter] [/X=exclude filter]
                  [/O] [/L] [/E=extension map]
                  [/C=remote computer name;remote scratch folder]
                  [/FAIL=failed results log file location] [/SIL=conversion log file path]
                  [/D="name:value"] [/W=wait time]
                  [/T=alternate temp folder]
                  [/NAME=sort files by name] [/CREATED=sort files by creation date]
                  [/MODIFIED=sort files by name] [/DESC=sort files in descending order (Z-A, 9-0)]
                  "folder"
```

Sample Command Lines

Combine all files in the folder into a single TIFF image:

```
DCSCombineFolder /S="C:\Test\Output" /N="CombinedFromFolder"
                  /P="TIFF 200dpi Monochrome"
                  "C:\Input\"
```

Searches for and send all files from the root of the C:\Input\ folder to Document Conversion Service to be converted using the settings contained in the conversion profile *TIFF 200dpi Monochrome*.

The converted files are combined and saved as single TIFF image, named *CombinedFromFolder.tif* in the output folder *C:\Test\Output*.

If the file *C:\Test\Output\CombinedFromFolder.tif* already exists, or if one of the files in the combine set fails to convert, the combine will fail and a results log file will be placed in a folder named *.failed* created in the save location. The results log file name will be *"PNCombineFolder_"*, followed by a date and time stamp and ending with *failed.dcsresults*. This can be controlled with the */FAIL* switch.

To overwrite an existing file the */O* switch would need to be added to the above command.

Convert all files in the folder to a multipage PDF:

```
DCSCombineFolder /S="C:\Test\Output" /N="CombinedFromFolder"
/P="PDF 300dpi OptimizedColor"
"C:\Input\"
```

Searches for and sends all files from the root of C:\Input\ folder to Document Conversion Service to be converted using the settings contained in the conversion profile *PDF 300dpi OptimizedColor*.

The converted files are combined and saved as a multipaged PDF file named *CombinedFromFolder.pdf* in the output folder *C:\Test\Output*.

If the file *C:\Test\Output\CombinedFromFolder.pdf* already exists, or if one of the files in the combine set fails to convert, the conversion will fail and a results log file will be placed in a folder named *.failed* created in the save location. Where this file is saved can be controlled with the */FAIL* switch.

The results log file name starts with *PNCombineFolder*, contains a date and time stamp and ends with *.failed.dcsresults*.

Convert all Word documents in a folder and all subfolders to a multipage PDF:

```
DCSCombineFolder /S="C:\Test\Output" /N="CombinedFromFolder" /R /F="*.doc|*.docx"
/P="PDF 300dpi OptimizedColor"
"C:\Input\"
```

Searches C:\Input\ and any subfolders for Word documents to send to Document Conversion Service to be converted using the settings contained in the conversion profile *PDF 300dpi OptimizedColor*.

The converted files are combined and saved as a multipaged PDF file named *CombinedFromFolder.pdf* in the output folder *C:\Test\Output*.

If the file *C:\Test\Output\CombinedFromFolder.pdf* already exists, or if one of the files in the combine set fails to convert, the conversion will fail and a results log file will be placed in a folder named *.failed* created in the save location. Where this file is saved can be controlled with the */FAIL* switch.

The results log file name starts with *PNCombineFolder*, contains a date and time stamp and ends with *.failed.dcsresults*.

Combine all files except PDF to a new PDF file, save conversion results logs to a specific location:

```
DCSCombineFolder /R /L /S="C:\Test\Output" /N="CombinedFromFolder"  
/P="PDF 300dpi OptimizedColor" /X="*.pdf"  
/FAIL="C:\Test\Output\Failed\"  
"C:\Input\"
```

Searches C:\Input\ and any subfolders (/R) for all files except PDF documents (/X="*.pdf") to send to Document Conversion Service to be converted using the settings contained in the conversion profile *PDF 300dpi OptimizedColor*.

The converted files are saved as single PDF file, *CombinedFromFolder.pdf*, in the output folder *C:\Test\Output*.

If a file with the same name already exists, or if one of the files in the combine set fails to convert, the combine will fail and a conversion results log file will be placed into the folder *C:\Test\Output\Failed*.

The conversion results log file name starts with *PNCombineFolder*, contains a date and time stamp and ends with *failed.dcsresults*. You can use the /D parameter *UseDateTimeInFailedFolder* to remove the date and time stamp from the results log file name.

To overwrite an existing file the /O switch would need to be added to the above command.

Command Line Arguments

Command line switches are not case-sensitive and can be entered in either upper or lower case.

/R - Include Subfolders (Recurse)

Use this switch to also search any subfolders under the source folder when building the list of files to be passed to Document Conversion Service to be converted.

/F - File Filter

A filter can be provided using this switch to only process certain types of files. Multiple file filters can be combined using the pipe (|) character. Hidden and system files are ignored, and the search pattern filters files based on a regular expression match of the long name of a file.

When this switch is not specified all files in the folder are (*.*) passed to Document Conversion Service to be processed.

Examples:

Convert PDF only: /F="*.pdf"

Convert Word, Excel and PDF only: /F="*.doc|*.docx|*.xls|*.xlsx|*.pdf"

Convert all Word files starting with MEMO: /F="MEMO*.doc"

/X - Exclude File Filter

A exclude file filter can be provided to take the returned file list gathered using the /F file filter and exclude any files that match a pattern. Multiple patterns can be combined using the pipe (|) character. By default no files are excluded.

Examples:

Exclude Word and Excel 2010 documents: /X="*.docx|*.xlsx"

Exclude all files starting with "Draft": /X="Draft*.*"

/S - The Save Location

This is a required argument. Pass in the full path to the folder in which to save the new files.

- If the path includes spaces it must be enclosed in quotes.
- If the path doesn't exist, the conversion will fail.
- If a file of the same name already exists in the save file location, the conversion will fail. To enable file overwriting, use the /O option.

Example:

/S="C:\Converted Files\Test"

/N - Output File Name

This is a required argument and specifies the name to use for the output file. The default file extension for the type of file being created will always be added to the name provided here.

Example:

```
/N="CombinedOutput_06_15_2012"
```

/O - Overwrite Always

Enables overwrite mode so that existing files of the same name are overwritten with the new file. If overwrite is not specified the combine action will fail if a file of the same name already exists in the save location.

/L - Results Log

The results log file is an XML file containing a complete snapshot of the combine request. Normally only saved for failed conversions, the /L argument enables creation of the results log file when the conversion succeeds.

All results log files for this command line utility start with *PNCombineFolder_*, contain a date and time stamp and end with the conversion status.

When the combine has succeeded, the results log file is placed in the same folder as the output (specified using the /S switch) and would have a name similar to the following:

```
PNCombineFolder_2013_05_31_2_50_05_PM_3.succeeded.dcsresults
```

The bold text in the name will change for each file and is based on the date and time of the run and an internal counter. You can suppress the use of the date and time information in the file name by passing *false* for the *UseDateTimeInFailedFolder* setting using the /D switch.

In the case of a failed conversion, the log file is always created. See the /FAIL switch to control the location and creation of the failed results log files.

The result log files can later be passed to the [DCSExtractResults](#) command line utility to extract information such as all files created or any errors encountered during conversion. You can search a folder for the results log files using the [DCSCreateFileList](#) utility.

/FAIL - Combine Results Log File Location

In the case of a failed combine, the combine results log file is always created. When the combine does not succeed, a *.failed* folder is created in the save folder location (provided by the /S switch) and the results log files are stored there.

The name of the results log when the combine does not succeed will be similar to the following:

```
PNCombineFolder_2013_05_31_2_50_05_PM_3.failed.dcsresults
```

The bold text in the name will change each time a combine command is run and is based on the date and time of the run and an internal counter.

This argument allows you to override the default use of the *.failed* folder and to provide a specific folder in which to store the failed results log file. You can suppress the use of the date and time information in the file name by passing *false* for the *UseDateTimeInFailedFolder* setting using the /D switch.

Note: The double ending backslash used when specifying the folder for the /FAIL switch is required for the command line path to be parsed correctly.

Examples:

```
/FAIL="C:\ConvertedFiles\Failed\\" /D="UseDateTimeInFailedFolder:FALSE"
```

If you do not want to create the failed results log files at all, you can use the /D switch to pass the *KeepFailedItemResultsFiles* setting as *false*.

On the command line:

```
/D="KeepFailedItemResultsFiles:False"
```

In a conversion profile:

```
<add Name="KeepFailedItemResultsFiles" Value="False"/>
```

The result log files can later be passed to the [DCSExtractResults](#) command line utility to extract information such the source file used or any errors encountered during conversion. You can search a folder for the results log files using the [DCSCreateFileList](#) utility.

/P - Conversion Profile

This argument is required. The type of file created is controlled by supplying a conversion profile using this switch. The profiles are referenced by passing in the name of the profile XML file, with or without the XML extension. See [Creating and Customizing Profiles](#) for more information about the contents of the profiles, a list of profiles included with Document Conversion Service, and how to create your own.

Examples:

```
/P="PDF 300dpi OptimizedColor"  
/P="TIFF 204x196dpi Monochrome Fax.xml"
```

/D - Define Setting

Individual profile settings can be supplied on the command line using this switch. This switch can be specified multiple times for separate settings. Any settings passed here will override the settings in the profile.

Any name-value pair that can be written in a profile can be passed through this parameter. This includes options to control the conversion settings as well as the behavior of the individual converters as well. See [Creating and Customizing Profiles](#) for more information about the name-value pairs that can be used.

Examples:

These first two are settings that control the converter options, such as what pages to print, and the output that PowerPoint will print.

```
/D="PrintRange:1-5"
```

```
/D="PowerPoint.PrintOptionsOutputType:PrintOutputNotesPages"
```

These two settings control the output file creation options, and would override or add to the settings in the conversion profile passed using the /P switch.

```
/D="Image Options;Fax Resolution:3"
```

```
/D="TIFF File Format;BW compression:Group3-2D"
```

These two settings control the where the failed results log files are created and are most often used along with the /FAIL switch to control where the results log files are saved.

```
/D="KeepFailedItemResultsFiles:TRUE"
```

```
/D="UseDateTimeInFailedFolder:FALSE"
```

/E - File Extension Mapping

A file extension mapping profile uses the extension of the source file to determine what converter will be used to convert the file before combining them together. Like the conversion profiles, this file is also an XML file. This switch is optional and an internal default mapping is provided. You would only need to provide this file if you wanted to override the default file extension to converter mappings provided.

Examples:

```
/E="Custom Extension To Converter Map"
```

/W - Wait Time

Use this switch wait to the specified number of seconds for the Document Conversion Service to be running and available to convert and combine documents. If Document Conversion Service is already running the command executes immediately. If the Document Conversion Service is not running in the timeout period specified, the command will return with an error.

If this argument is not specified the command will return immediately with an error if Document Conversion Service is not running.

Example:

```
/W=300
```

/C - Convert on a Remote Computer (DCOM)

If Document Conversion Service is running on a different computer, use this switch to pass the name of the remote computer and the path of a shared location that both computers have access to. Separate the name of the remote computer and the path to the shared folder location with a semi-colon.

When combining remotely, the client redistributable, PNDocConvClientSetup_3.0.exe, must be installed on the computer running this command line utility. The client setup install program is included as part of the Document Conversion Service install and can be found in the **\Samples\Redist** folder in your product installation folder.

Examples:

```
/C="DOCCONV_SERVER;\\DOCCONV_SERVER\DCSREMOTE"
```

/SIL - Smart Inspect Logging File

Smart Inspect Log files are a tracing of the entire conversion process and are not the same as the conversion results log files created when a conversion fails. These logs can be viewed using the SmartInspect Redistributable Console included with Document Conversion Service.

These log files are automatically deleted when conversion succeeds. To keep the log files on success use the custom setting *AlwaysKeepProcessingLoggingFiles* as shown below.

The default location for this file is the TEMP folder. Each logging file is assigned a unique date, time and thread prefix followed by "_PNCombineFolder.sil", such as
`2014_09_11_2_38_00_PM_4_PNCombineFolder.sil`.

Use this argument to specify a custom path and optional file name for the SmartInspect logging file (*.sil) created by this utility. The /SIL switch can take a folder, or a path to a filename. If a path without a trailing backslash is provided, the last part of the path is assumed to be a filename.

Note: The double ending backslash used when specifying a folder for the /SIL switch is required for the command line path to be parsed correctly.

/SIL=	Is interpreted as...
"C:\Test\LogFile"	Create the SmartInspect log file as C:\Test\LogFile.sil.
"C:\Test\LogFile\\"	Create the SmartInspect log file as C:\Test\LogFile\datetime_PNCombineFolder.sil
"C:\Test\LogFile\ConvertFileCustom.sil"	Create the SmartInspect log file as C:\Test\LogFile\ConvertFileCustom.sil

The following settings can be used to control the creation and naming of the logging file. These settings are all passed using the /D switch.

Custom Setting	Description
RemoveDateTimePrefixOnProcessingLoggingFiles	Pass <i>True</i> to disable the adding of the unique date, time and thread prefix when a custom file name has not been specified in the <i>ConvertFileProcessLoggingPath</i> parameter.
KeepFailedProcessingLoggingFiles	Pass as <i>False</i> to disable the automatic creation of SmartInspect logging files when conversion fails. This setting can be overridden by <i>AlwaysKeepProcessingLoggingFiles</i> .
AlwaysKeepProcessingLoggingFiles	When set to <i>True</i> , the SmartInspect logging files are always created in the %TEMP% or other specified folder for both successful and failed conversions. If set to <i>False</i> , no logging files are created. This setting will override the <i>KeepFailedProcessingLoggingFiles</i> setting.

Examples:

Pass a custom folder and remove the prefix, each run will overwrite the log file C:\PEERNET\Logs\PNCombineFiles.sil.

```
/SIL="C:\PEERNET\Logs\" /D="RemoveDateTimePrefixOnProcessingLoggingFiles:TRUE"
```

Pass a custom folder and log file name and remove the prefix. Each run will overwrite the logging file C:\PEERNET\Logs\MyLogFile.sil.

```
/SIL="C:\PEERNET\Logs\MyLogFile" /D="RemoveDateTimePrefixOnProcessingLoggingFiles:TRUE"
```

Don't save any SmartInspect log files at all.

```
/D="AlwaysKeepProcessingLoggingFiles:FALSE"
```

/T - Alternate Temp Folder

This is an advanced setting that should not be needed in most cases. When converting files, the conversion tool copies each file and performs the conversion in temporary *staging* and *working* folders created on demand in the default Windows temp folder. When dealing with long path and file names the default folders created can occasionally cause path names that are too long to process. When this happens this switch can be used to set the temporary folder to a shorter path to allow processing.

This setting is overridden if the /C option for remote conversion is being used with its own path to a shared location for conversion.

Examples:

```
/T="C:\PNTemp\"
```

/NAME - Sort Files by Name

The file list picked up from the folder is sorted by file name in ascending order (0-9, A-Z).

/CREATED - Sort Files by Creation Date

The file list picked up from the folder is sorted by the files creation date file name in ascending order (0-9, A-Z).

/MODIFIED - Sort Files by Modified Date

The file list picked up from the folder is sorted by the files last modified date file name in ascending order (0-9, A-Z).

/DESC - Sort Files in Descending Order

Use with /NAME, /CREATED or /MODIFIED. Sorts the files in descending order (Z-A, 9-0).

/? - Display Help

When passed as the only argument this switch will display help for this command.

Folder

The folder containing the files to convert.

- If the path to the folder includes spaces it must be enclosed in quotes.
- If the folder doesn't exist, the conversion will fail.

DCSExtractResults

A command line utility to extract information from the results log files. One of the switch arguments must always be specified. If more than one switch is found, the first one is always used. The information extracted is sent to standard out.

```
DCSExtractResults [/s] [/C] [/E] file
```

The results log files are created by the the following command line utilities:

[DCSConvertFile](#)
[DCSConvertFileList](#)
[DCSConvertFolder](#)
[DCSCombineFiles](#)
[DCSCombineFolder](#)

Sample Command Lines

Extract a list of all files created to standard out:

```
DCSExtractResults /C "Document.doc.succeeded.dcsresults"
```

Extract a list of all files created from the *Document.doc.succeeded.dcsresults* log file and sends the information to the console through standard output.

Extract a list of all errors into a text file:

```
DCSExtractResults /E "C:\Test\Output\Document.doc.failed.dcsresults" > "C:\Test\Errors.txt"
```

Extract a list of any errors from the *Document.doc.failed.dcsresults* log file and saves them in the text file C:\Test\Errors.txt.

Extract the source file name of a failed conversion result file:

```
DCSExtractResults /S "C:\Test\Output\Document.doc.failed.dcsresults" >> "C:\Test\Failed.txt"
```

Extracts the source file name from the *Document.doc.failed.dcsresults* file and appends it into the text file C:\Test\CreatedFiles.txt.

Command Line Arguments

Command line switches are not case-sensitive and can be entered in either upper or lower case.

/S - Extract the source file names

Extracts the source file information from the conversion results log file. For [DCSConvertFileList](#) and [DCSCombineFiles](#) this can be more than one file.

/C - Extract the created file names

Extracts the list of created files, if any, from the conversion results log file.

/E - Extract the errors

Extracts the list of errors, if any, from the conversion results log file.

/? - Display Help

When passed as the only argument this switch will display help for this command.

file

The full path to the file to the conversion results log file

DCSCreateFileList

A command line utility to search a folder, and optionally any subfolders and return a list of files matching the search filter specified. The information extracted is sent to standard out.

```
DCSCreateFileList [/R] [F=filter]
                  [/NAME=sort files by name] [/CREATED=sort files by creation date]
                  [/MODIFIED=sort files by name]
                  [/DESC=sort files in descending order (Z-A, 9-0)]
                  searchfolder
```

This utility can be used to search folders for files to send to the command line utilities, or to find the the results log files created by any of command line utilities. The folder search is optimized for speed and efficiency and will return all files that match the filter provided.

Hidden and system files are ignored, and the search pattern filters files based on a regular expression match of the long name of a file. This is different from the Microsoft .Net System.IO.Directory.GetFiles method which returns files based on a check against file names with both the 8.3 file name format and the long file name format, which can cause unexpected file names to be returned.

The file list returned can optionally be sorted by name, date created or date modified. Results are always in ascending order (0-9, A-Z). Pass /DESC to return the results in descending (9-0, Z-A) order.

Sample Command Lines

Searching a folder based on a single file type:

```
DCSCreateFileList /F="*.tif" "C:\Test\Output"
DCSCreateFileList /R /F="*.tif" "C:\Test\Output"
```

Searches the folder C:\Test\Output for all files ending in the pattern *.tif*. Only files with the three letter extension *.tif* will be returned.

The second example will recursively search the folder C:\Test\Output and all subfolders for all files ending on *.tif*.

Search a folder for more than one file type, sort by name:

```
DCSCreateFileList /R /F="*.doc|*.pdf" /NAME "C:\Test\Input"
DCSCreateFileList /R /F="*.doc|*.pdf" /NAME "C:\Test\Input" > C:\Test\InputFileList.txt
```

To search for more than one file type, separate the filter patterns using the pipe (|) character.

This example recursively searches the folder C:\Test\Input for all files ending in the *.doc* or *.pdf* extension. The complete path to all files with only the three letter extension *.doc* and *.pdf* will be returned and sent to the console through standard output.

The second command line shown uses the redirection operator > to redirect the console's standard output into a text file located at C:\Test\InputFileList.txt.

Search the folder C:\Test\Output for all succeeded result log files:

```
DCSCreateFileList /F="*.succeeded.dcsresults" "C:\Test\Output"
```

```
DCSCreateFileList /R /F="*.succeeded.dcsresults" "C:\Test\Output" > "C:\Test\CompletedResults.txt"
```

Searches the folder C:\Test\Output for all results log files that represent completed conversions. The full path to each matching file found is sent to the console through standard output.

The second example does the same as the first except that it recursively searches the folder C:\Test\Output and any subfolders for all results log files that represent completed conversions, not just the root folder. It also uses the redirection operator > to redirect the output into a text file located at C:\Test\CompletedResults.txt.

Search the folder C:\Test\Output for all failed results log files:

```
DCSCreateFileList /F="*.failed.dcsresults" "C:\Test\Output"
```

```
DCSCreateFileList /R /F="*.failed.dcsresults" "C:\Test\Output" > "C:\Test\FailedResults.txt"
```

Searches the folder C:\Test\Output for all results log files that represent failed conversions. The full path to each matching file found is sent to the console through standard output.

The second example does the same as the first except that it recursively searches the folder C:\Test\Output and any subfolders for all results log files that represent failed conversions, not just the root folder. It also uses the redirection operator > to redirect the output into a text file located at C:\Test\FailedResults.txt.

Command Line Arguments

Command line switches are not case-sensitive and can be entered in either upper or lower case.

/R - Include Subfolders (Recurse)

If this switch is used, the subfolders under the folder are included when searching for the list of files that match the filter pattern.

/F - File Filter

Defines the filter that determines what files can be returned, such as using *.pdf to only process PDF files. When this switch is not specified all files (*.*) in the folder are will be returned. Hidden and system files are ignored, and the search pattern filters files based on a regular expression match of the long name of a file.

Multiple filters are combined using the pipe (|) character, such as *.doc|*.pdf to process only Word and PDF files.

/NAME - Sort Files by Name

The file list picked up from the folder is sorted by file name in ascending order (0-9, A-Z).

/CREATED - Sort Files by Creation Date

The file list picked up from the folder is sorted by the files creation date file name in ascending order (0-9, A-Z).

/MODIFIED - Sort Files by Modified Date

The file list picked up from the folder is sorted by the files last modified date file name in ascending order (0-9, A-Z).

/DESC - Sort Files in Descending Order

Use with /NAME, /CREATED or /MODIFIED. Sorts the descending order (Z-A, 9-0).

searchfolder

The full path to the folder in which to start searching.

/? - Display Help

When passed as the only argument this switch will display help for this command.

DCSLicenseDaysLeft

A command line utility to echo to the command line the current license level and the number of days remaining in the subscription period. The information extracted is sent to standard out. This command line tool needs no arguments or parameters.

The command line tool reports as follows:

```
Level Edition; # days
```

A Level I license with 200 days remaining would see the following:

```
Level I Edition; 200 days
```

An expired license reports its expired state:

```
Level I Edition; 0 days [EXPIRED]
```

The Watch Folder Service

The Watch Folder Service is a Windows service for converting files from "hot" folders. These are sometimes also called "drop" folders. The service will watch one or more folders at a time and convert any files dropped into those folders to the format specified for that folder. This gives you the freedom to do such tasks as watch two separate folders and create black and white TIFF images out of the files dropped in the first folder, and color TIFF images from files dropped in the other folder. This can be expanded to watch as many folders as needed.

The service is installed as part of the Document Conversion Service install and is configured to use the same user account as the PEERNET Document Conversion Service Monitor 1.0 specified during installation. This service requires a privileged user account to be able to access shared volumes and to allow for remote conversion.

This application is also provided as a Visual Studio project in C#.NET and demonstrates using PNDocConvQueueServiceLib from a service in a multithreaded environment.

High Performance Clustering and Failover (DCS 3.0.010)

Clustering allows more than one computer to process against the same group of files. This group of servers all working on the same set of files is called a *cluster*. This type of configuration allows for an increase in conversion speed and provides fail over support if a server in the cluster needs to be restarted. The other servers in the cluster will continue to convert files. This is explained in more detail in [High Performance Clustering and Fail Over Conversion](#).

OCR - Optical Character Recognition (DCS 3.0.031)

Beginning with Document Conversion Service 3.0.031, you can now convert scanned PDF documents and images to searchable PDF using **Optical Character Recognition**, or **OCR**. These options apply when using the PEERNET built-in converters *Adobe PDF - Builtin*, *Image - Builtin* and *Cadd - Builtin* that are installed with Document Conversion Service. These settings only apply when creating PDF files, they do not apply to any other output format. The sample watch folder, **OCR to AdobePDF Watch Folder**, has OCR enabled and creates searchable PDF for English, French, and Spanish.

The PDF files created using OCR consist of each page embedded as an image, with the page text as an invisible layer over the top of the image. The invisible text layer can be searched and text content can be copied from the PDF if permissions permit. See [OCR Images and Scanned PDF Files to Searchable PDF](#).

Large Volume Batch Conversion Using Clustering (DCS 3.0.010)

Extremely large folders of files can be processed efficiently using clustering on a single computer. The same technique that allows multiple computers to process from a shared location also allows a single computer to efficiently work its way through a large folder of documents. See [Large Volume Batch Conversion Using Clustering](#) for details.

Outlook and EML Message Archives (DCS 3.0.009, 3.0.029)

Each folder section can be configured to extract and convert all attachments in Outlook Message (*.msg) archives as well as the message itself. See [Processing Outlook and EML Mail Messages and Attachments](#) for full details.

Post Conversion Processing (DCS 3.0.010)

Each watch folder can optionally run a command at the end of the conversion process. Commands are run on each created file for successful conversions, and on the original source file in the case of a failed conversion. See [Post-Conversion Processing](#) for more information.

Folder Flattening and Unique File Names (DCS 3.0.019)

Each watch folder's default behavior is to maintain any folder structure found in the input folder when creating the converted files in the Output folder. If desired, this input folder structure can be ignored, or flattened, when creating files in the output folder. To prevent filename collision and overwrite issues, options to automatically create unique filenames using Globally Unique Identifiers (GUIDs) can be used. See [Unique File Naming and Flat Folder Structures](#) for an example.

Creating Done Files to Signal Completion

Added in version 3.0.025, this allows for the creation of a *done* file for each file converted, or a *failed* file if the conversion fails. The file lists the path to the input file that was converted, followed by a list of the files created. This file is not created until conversion is complete or failed and can be used to signal completed to other processes. See [Creating Done Files to Signal Completion](#) for more details.

Picking up Files in Sort Order

Added in version 3.0.027, this allows for the files be picked and converted in order by *Name* (default), *DateModified* or *DateCreated* and *None*. If the sorting mode is set to *None* then no ordering is applied to the list of files returned from the underlying file system. The sort order can be *Ascending* (default) or *Descending*. See [Control Sort Order on File Pickup](#) for more details.

Files in the root of the input folder are picked up and sorted first. Then, if enabled, any subfolders, in alphabetical order, are searched for files to convert. The file list for each subfolder is returned in the requested sort mode and added to the list of files to convert.

Sample Conversion Folders

The Watch Folder Service is pre-configured to create several example conversion folders for the following common conversion types and scenarios:

ConvertToTIFF	Creates 300 DPI Optimized TIFF images.
ConvertToFaxTIFF	Creates 204x196 DPI Monochrome faxable TIFF images.
ConvertToAdobePDF	Creates, where possible, vector (searchable) Adobe PDF files. If you need to keep the hyperlinks in your documents when creating PDF files, you'll want to use this folder.
OCR to AdobePDF Watch Folder	Uses Optical Character Recognition (OCR) to create searchable PDF files from scanned PDF or images. Works only with the PEERNET built-in converters <i>Adobe PDF - Builtin</i> , <i>Image - Builtin</i> and <i>Cadd - Builtin</i> that are installed with Document Conversion Service. This folder is pre-configured for clustered processing using a shared folder that is created as part of the installation.
ConvertToRasterPDF	Creates PDF files where each page is an image, similar to a scanned image. Good for archiving as the page content cannot be changed.
ConvertToJPG	Creates color JPG images at 300 DPI. One image is created for each page of each document.
LargeBatchTIFF	<p>This folder is configured to allow dropping a large number of files at once into it's input folder. The files are then picked in small batches of up to 10 files until all files in the folder have been converted.</p> <p>For a more in-depth explanation of converting an existing folder containing a very large number of files, look at Large Volume Batch Conversion with Watch Folder Service.</p>
Clustered ConvertToTIFF	This folder is pre-configured for clustered processing using a shared folder that is created as part of the installation. It creates 300 DPI Optimized TIFF images. See High Performance Clustering and Fail Over Conversion .

Running the Watch Folder Service

- [Watch Folder Service Overview](#)
- [Starting and Stopping the Watch Folder Service](#)
- [Configure the Watch Folder Service](#)
- [High Performance Clustering and Fail Over Conversion](#)
- [OCR Images and Scanned PDF Files to PDF](#)
- [Long Path Name Support](#)
- [Skipping Files with the Passthrough Converter](#)
- [Processing Outlook and EML Mail Messages and Attachments](#)
- [Creating Done Files to Signal Completion](#)
- [Control Sort Order on File Pickup](#)
- [Post-Conversion Processing](#)
- [Large Volume Batch Conversion Using Clustering](#)
- [Large Volume Batch Conversion Using Synchronous File Pickup](#)

Watch Folder Service Overview

The following is an overview of how the Watch Folder Service works.

For each folder watched, the service uses the following:

- an *enabled* flag that determines if the folder will be monitored or not
- an *input folder* to collect the files to be converted based on a search pattern and optional sub-directory inclusion
- a *staging folder* to hold the files being processed
- a *working folder* that holds the output files during creation
- an *output folder*, which is the final destination of the created files; they are copied into this folder when conversion is complete
- a *failed folder* that contains a copy of any file which failed to be converted
- an optional *completed folder* to hold a copy of all input files that have been processed
- options to control the number of files picked up at a time and if batches are run synchronously
- options to extract and process attachments from Outlook MSG archive files
- how files are stored in the *completed* and *failed* folders

The input folders are polled on a customizable time interval looking for files or folders to convert. If any files or folders of files are dropped into the input folder that meet the criteria of what files you want to convert, these files, or the number of files allowed, are moved into a uniquely named folder (based on date and time) under the *staging* folder.

When a folder is dropped into the input location, it searches for files that match the criteria. If any matching files are found in the folder, the folder's structure is mirrored under the new folder in the staging location and the files copied for conversion. During all subsequent steps of copying to the output folder, failed folder or completed folder the folder's structure is kept intact.

Once under the *staging* folder, the files are passed to Document Conversion Service to be converted using the output format settings provided for that watch folder. Putting files in this *staging* folder prevents file name collisions if another file of the same name is dropped into the folder by another user.

Converted files are stored first in the *working* folder while they are being created. Once complete, they are copied into the *output* folder. If any file should fail to convert a folder under the *failed* folder (using the same date and time stamped folder name as was created under the *staging* folder) is created and the failed file is copied there.

If the *completed* folder is specified, all source files are copied to a new folder (using the same date and time stamped folder name as was created under the *staging* folder) under the completed folder. If the *completed* folder is not specified, the source files are deleted.

If you do not want your completed and failed files copied into subfolders under their respective folders, this behavior can be disabled to copy the files directly into the folders provided without creating the subfolder. Take note that with this behavior existing files with the same name will be overwritten.

The sample uses the file extension of the file chosen to determine what converters PNDocConvQueueServiceLib will try and use when converting the files.

You can provide a single converter name or a semi-colon separated list of converter names to use. If you pass a list of names the first matching converter name that has a running converter in Document Conversion Service will be used. See [What Files Can I Convert?](#) for a list of converter names to use.

Starting and Stopping the Watch Folder Service





Before you begin...



If Document Conversion Service is not running, the Watch Folder Service sample can be started but it will not process any files until Document Conversion Service is also running. Follow the steps in [Starting and Stopping the Service](#) to start the Document Conversion Service.

Starting and Stopping the Watch Folder Service from the DCS Dashboard

At the top of the DCS Dashboard you have easy access to starting and stopping Watch Folder Service.

DCS Service:   Stopped

Watch Folder Service:   Stopped

1. To start the Watch Folder Service, click the green *play* icon (). The status will change to **Starting...**, and move to **Running** once the service is running.
2. To stop the service, click the green *stop* icon (). The status will change to **Stopping...** and go to **Stopped** when the service has stopped running.

Starting and Stopping the Watch Folder Service from Start Menu

1. Start the Watch Folder Service by going to Start - All Programs - PEERNET Document Conversion Service 3.0 - Watch Folder - Start Watch Folder Service.
2. A message box will appear when the service has been started or if it has failed to start.
3. To stop the Watch Folder Service by going to Start - All Programs - PEERNET Document Conversion Service 3.0 - Watch Folder - Stop Watch Folder Service.
4. A message box will appear when the service has been stopped.

Starting and Stopping the Watch Folder Service from the Services Panel

1. Open the Services panel by going to Start - Control Panel - System and Security - Administrative Tools - Services (or type "*Services*" into the search field on the Start menu to open the Services panel).
2. In the Service control panel applet locate the service PEERNET Watch Folder Service.
3. Select Start from the left hand side to start the service, or **Stop** to stop it if it is running.

Configure the Watch Folder Service

The Watch Folder Service application configuration file (an XML file) contains the custom configuration section `<WatchFoldersSection>`. This section contains the `<WatchFolders>` section at the top of the file and a global `<Settings>` section at the bottom.

The `<WatchFolders>` section at the top contains, to start, 6 individual `<WatchFolder>` sections, one for each sample conversion folder provided. You can modify these sample sections as needed to meet your requirements, or you can copy them and edit as needed to add your own Watch Folders.

Each `<WatchFolder>` section consists of a `<Settings>` collection of name-value pairs. These settings can be grouped into two sections: the [folder settings](#) and the [output file settings](#).

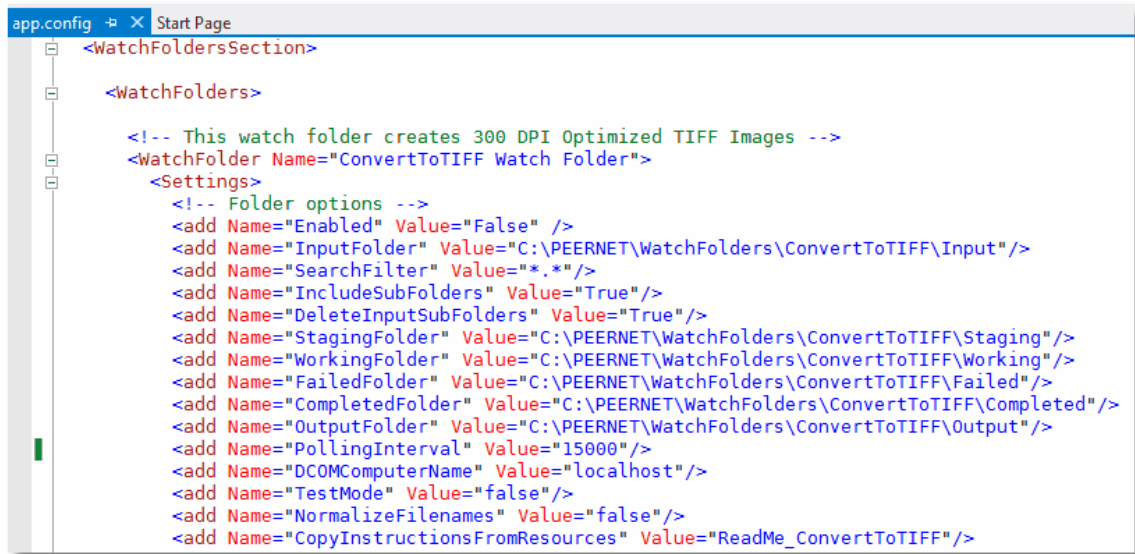
The `<Settings>` section at the bottom of the file contains the [file extension to converter mapping](#) that is used by Watch Folder Service to determine what converter(s) to use for each file type. You can provide a single converter name, or a semi-colon separated list of converter names for each unique file extension. If you pass a list of names the first converter that is found and is running in Document Conversion Service will be used.

Changing the Watch Folder Service Configuration

A copy of the Watch Folder Service sample is installed as a Windows service when Document Conversion Service is installed.

To modify its configuration you need to change the service's application configuration file.

1. Open the Watch Folder Service configuration file by going to the **DCS Dashboard - Watch Folder Settings -Edit Configuration**. You can also open this file by going to Start - All Programs - PEERNET Document Conversion Service 3.0 - Watch Folder - Configure Watch Folder Settings.
2. This opens the configuration file in the [DCS Editor](#) where you can edit the current folders, or copy and paste to make new ones.
3. In the editor, scroll to find the `<WatchFolder>` section for the conversion folder that you need to change, or use the [Find and Replace tool](#). If you are creating a new section, copy and paste one of the sample sections to start.
4. You will most likely need to change the paths specified for the InputFolder, Staging Folder, Working Folder, FailedFolder, Completed Folder and OutputFolder settings. Occasionally you may also need to configure the SearchFilter setting as well. See the [folder settings](#) section for more details.
 - a. You can enable or disable a conversion folder by setting the **Enabled** setting to *true* or *false*. When false, the folder is not monitored. The default is *true* if this setting is not provided.
 - b. The type of output file created is also controlled by the settings in this file. See the section on [output file settings](#) and the sample watch folder settings provided in the configuration file.
 - c. The file extension mapping is controlled by the `<Settings>` section. See [Changing the File Extension to Converter Mapping](#) for details.



```

app.config  X Start Page
<WatchFoldersSection>
  <WatchFolders>
    <!-- This watch folder creates 300 DPI Optimized TIFF Images -->
    <WatchFolder Name="ConvertToTIFF Watch Folder">
      <Settings>
        <!-- Folder options -->
        <add Name="Enabled" Value="False" />
        <add Name="InputFolder" Value="C:\PEERNET\WatchFolders\ConvertToTIFF\Input"/>
        <add Name="SearchFilter" Value="*.*/>
        <add Name="IncludeSubFolders" Value="True"/>
        <add Name="DeleteInputSubFolders" Value="True"/>
        <add Name="StagingFolder" Value="C:\PEERNET\WatchFolders\ConvertToTIFF\Staging"/>
        <add Name="WorkingFolder" Value="C:\PEERNET\WatchFolders\ConvertToTIFF\Working"/>
        <add Name="FailedFolder" Value="C:\PEERNET\WatchFolders\ConvertToTIFF\Failed"/>
        <add Name="CompletedFolder" Value="C:\PEERNET\WatchFolders\ConvertToTIFF\Completed"/>
        <add Name="OutputFolder" Value="C:\PEERNET\WatchFolders\ConvertToTIFF\Output"/>
        <add Name="PollingInterval" Value="15000"/>
        <add Name="DCOMComputerName" Value="localhost"/>
        <add Name="TestMode" Value="false"/>
        <add Name="NormalizeFileNames" Value="false"/>
        <add Name="CopyInstructionsFromResources" Value="ReadMe_ConvertToTIFF"/>
      </Settings>
    </WatchFolder>
  </WatchFolders>
</WatchFoldersSection>

```

5. Save the edited file. The [DCS Editor](#) will validate the file when saving. Restart the service to apply your changes.

The Folder Settings

The folder settings describe the following:

- if the folder is enabled or disabled
- the input folder that is being watched, what files to pick up out of that folder, how often to look for new files in the folder, and whether or not to include any folders under the input folder in the search
- the staging and working folders to use when converting files
- the output folder to store the converted files
- the failed folder to store files that fail to convert
- the completed folder, an optional folder to store the original files that were converted
- other options that define how many files are picked up at once and if batches are run synchronously
- how files are stored in the completed and failed folders



Code Sample - Folder Settings

```
<WatchFolders>

<!-- This watch folder creates 300 DPI Optimized TIFF Images -->
<WatchFolder Name="ConvertToTIFF Watch Folder">
  <Settings>

    <!-- Folder Options -->
    <add Name="Enabled" Value="True"/>
    <add Name="InputFolder" Value="C:\PEERNET\WatchFolders\ConvertToTIFF\Input"/>
    <add Name="SearchFilter" Value="*.*/>
    <add Name="IncludeSubFolders" Value="True"/>
    <add Name="DeleteInputSubFolders" Value="True"/>
    <add Name="StagingFolder" Value="C:\PEERNET\WatchFolders\ConvertToTIFF\Staging"/>
    <add Name="WorkingFolder" Value="C:\PEERNET\WatchFolders\ConvertToTIFF\Working"/>
    <add Name="FailedFolder" Value="C:\PEERNET\WatchFolders\ConvertToTIFF\Failed"/>
    <add Name="CompletedFolder" Value="C:\PEERNET\WatchFolders\ConvertToTIFF\Completed"/>
    <add Name="OutputFolder" Value="C:\PEERNET\WatchFolders\ConvertToTIFF\Output"/>
    <add Name="PollingInterval" Value="15000"/>
    <add Name="DCOMComputerName" Value="localhost"/>
    <add Name="TestMode" Value="false"/>
    <add Name="NormalizeFileNames" Value="false"/>
    <add Name="CopyInstructionsFromResources" Value="ReadMe_ConvertToTIFF"/>

    <!-- 0 means no limit -->
    <add Name="Polling.MaxFilesToProcessAtATime" Value="0"/>
    <add Name="Polling.SynchronousFilePickup" Value="false"/>

    <add Name="UseTimeDateSubFoldersInCompletedFolder" Value="true"/>
    <add Name="UseTimeDateSubFoldersInFailedFolder" Value="true"/>
    <add Name="UseCompressedDateTimeFormat" Value="false" />

    <!-- Preprocess Archive Settings -->
    <!-- Comment out this line or set this as empty string to disable MSG archive processi
    <!-- <add Name="PreprocessArchiveFormatsFilter" Value="*.msg" /> -->
```



Code Sample - Folder Settings

```
<add Name="PreprocessArchive.IncludeExtensionInFolderName" Value="true" />

<!-- Preprocess MSG Archive Settings -->
<add Name="PreprocessArchive.MSG.IncludeInlineAttachments" Value="true" />
<!-- Pipe (|) separated list of file extensions (e.g *.doc|*.docx) to match. -->
<!-- Pass empty string for match all.-->
<add Name="PreprocessArchive.MSG.AttachmentsIncludeFilter" Value="" />
<!-- Pipe (|) separated list of file extensions (e.g *.png|*.jpg) to exclude. -->
<!-- Pass empty string to exclude none.-->
<add Name="PreprocessArchive.MSG.AttachmentsExcludeFilter" Value="" />

<!-- Clustered Processing -->
<!-- This forces batch mode processing with synchronous wait and -->
<!-- no date time stamp used in the Failed\Completed folders -->
<add Name="ClusteredProcessing.Enabled" Value="false" />
<!-- Override this for clustering to customize pickup -->
<!-- <add Name="ClusteredProcessing.MaxFilesToPickup" Value="6"/> -->

<!-- Run Command at End On Success -->
<!-- Any command entered here will run on successful conversion, on each file created.
<!-- Use &quot; to put the command in quotes if there are spaces, and to enclose param
<!-- $(OutputFilePath) is the full path to the created file.-->
<!-- $(SourceFileName) can also be passed as a parameter to allow correlating the sour
<add Name="RunAtEnd.Success.Enabled" Value="false" />
<add Name="RunAtEnd.Success.Command" Value="" />
<add Name="RunAtEnd.Success.Parameters" Value="&quot;$(OutputFilePath)&quot;;" />
<add Name="RunAtEnd.Success.StartDirectory" Value="" />
<!-- One of Normal, Min, Max, Hidden (default) -->
<add Name="RunAtEnd.Success.WindowState" Value="Hidden" />
<!-- Wait mode for the command, one of WaitForCompletion, WaitWithExitCode, DoNotWait
<add Name="RunAtEnd.Success.WaitMode" Value="DoNotWait" />
<!-- Default is 3 minutes -->
<add Name="RunAtEnd.Success.WaitModeMaxTime" Value="180000" />

<!-- Run Command at End On Failure -->
<!-- Any command entered here will run on a failed conversion, on the original source
<!-- Use &quot; to put the command in quotes if there are spaces, and to enclose param
<!-- $(FailedFilePath) is the path to the file in its failed location.-->
<!-- $(SourceFileName) can also be passed as a parameter to allow correlating the sour
<add Name="RunAtEnd.Fail.Enabled" Value="false" />
<add Name="RunAtEnd.Fail.Command" Value="" />
<add Name="RunAtEnd.Fail.Parameters" Value="&quot;$(FailedFilePath)&quot;;" />
<add Name="RunAtEnd.Fail.StartDirectory" Value="" />
<!-- One of Normal, Min, Max, Hidden (default)-->
<add Name="RunAtEnd.Fail.WindowState" Value="Hidden" />
<!-- Wait mode for the command, one of WaitForCompletion, WaitWithExitCode, DoNotWait
<add Name="RunAtEnd.Fail.WaitMode" Value="DoNotWait" />
<add Name="RunAtEnd.V.WaitModeMaxTime" Value="180000" />

...
</Settings>
</WatchFolder>

</WatchFolders>
```

The **Enabled** setting is used to determine if this folder is to be watched. When set to *False*, this folder is not monitored. If this setting is not provided it defaults to *True*.

The **InputFolder** is polled on a customizable time interval looking for files or folders of files to convert. Files are chosen based on the **SearchFilter** setting. The default setting of *"*.*"* means all files will be picked up to be processed. You can specify what files to convert by changing this setting. Different file types are separated by the pipe (|) symbol. For example, to only pick up Word and PDF files from the folder, the SearchFilter can be set to **.doc|*.docx|*.pdf*.

If any files are dropped into the input folder that meet the criteria of what files you want to convert, these files, or the number of files allowed as explained below, are moved into a uniquely named folder (based on date and time) under the **StagingFolder**.

When a folder is dropped into the **InputFolder**, it is searched for files that match the criteria. If any matching files are found in the folder, the folder's structure is mirrored under a new folder in the **StagingFolder** and the files copied for conversion. During all subsequent steps of copying to the **OutputFolder**, **FailedFolder** or **CompletedFolder** the folder's structure is kept intact.

You can set a limit on how many files are picked up at a single time using the **Polling.MaxFilesToProcessAtATime** option. This is useful to when dealing with folders with a very large number of files as it allows you to automatically process the files in smaller groups. If you do need to process a large number of files in smaller batches, **Polling.SynchronousFilePickup** should also be set to *true* to allow the first group of files to finish converting before the next group of files is picked up. See [Large Volume Batch Conversion with Watch Folder Service](#) for a sample configuration.

Once under the **StagingFolder**, the files are passed to Document Conversion Service to be converted using the output format settings provided for that watch folder. Putting files into this temporary folder prevents file name collisions if another file of the same name is dropped into the folder by another user.

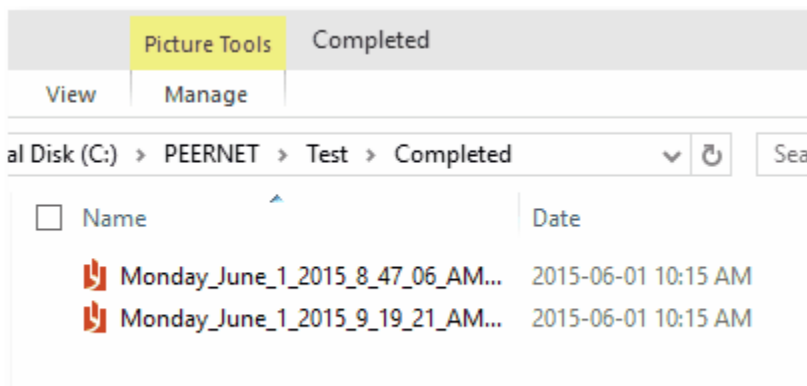
Converted files are first created in the **WorkingFolder** while they are being created. Once complete, they are copied into the **OutputFolder**.

If any file should fail to convert, a folder named using the same date and time stamped name as was created under the staging folder, is also created under the **FailedFolder** and the failed file is copied there.

If the **CompletedFolder** is set, files that were successfully converted are placed into a new subfolder under that folder. This subfolder is named using the date and time the files were picked up from the **InputFolder**. Each time a new set of files is found to convert, a new subfolder will be created.

If you do not want your completed and failed files copied into subfolders under the **CompletedFolder** and **FailedFolder**, this behavior can be changed to copy the files directly into the folders provided without creating the date and time stamped subfolder.

You disable this by setting the **UseTimeDateSubFoldersInCompletedFolder** option to *false*. When disabled, the files are copied directly into the **CompletedFolder**. If a file of the same name already exist in the folder it will be overwritten.



If you do not want to keep a copy of the original source files, you can set the **CompletedFolder** to an empty string, but take note that this will delete any files that have been dropped into the **InputFolder**.

Any file that fails to convert is moved into a new subfolder under the **FailedFolder**. Like the **CompletedFolder**, the subfolder is named using the date and time the files were picked up from the **InputFolder**. This can be disabled by setting the **UseTimeDateSubFoldersInFailedFolder** option to *false*. When disabled, the files are copied directly into the **FailedFolder**, and any file of the same name that already exists in the folder it will be overwritten.

Preprocessing Outlook Message Attachments (DCS 3.0.009)

Attachments to Outlook Message files (*.msg) can be automatically extracted and converted along with the original message file. This setting is off by default. To enable it, remove the comment markers on the **PreprocessArchiveFormatsFilter** setting. See [Processing Outlook Message Attachments](#) for full details.

Clustered Processing (DCS 3.0.010)

Clustered processing is available using the **ClusteredProcessing.Enabled** setting. Clustered processing allows you to point more than one computer running Document Conversion Service and Watch Folder Service at the same folder of files and have both computers convert files from that location. See [High Performance Clustering and Fail Over Conversion](#) for more information.

Post Conversion Processing (DCS 3.0.010)

Each watch folder can optionally run a command at the end of the conversion process. Commands are run on each created file for successful conversions, and on the original source file in the case of a failed conversion. See [Post-Conversion Processing](#) for more information.

Unique Output Filenames and Flat Folder Structures(DCS 3.0.019)

Each watch folder can provide a unique file name for each output file created, as well as the ability to flattening the input folder structure when creating the files in the output folder. See [Unique File Naming and Flat Folder Structures](#) for more information.

All Watch Folder Service Settings

Key	Value
ClusteredProcessing.Enabled	Set this to <i>true</i> to allow clustered processing on this Watch Folder. When this is <i>true</i> , other computers with Document Conversion Service and Watch Folder Service installed can be directed to convert from the same folder of files.
ClusteredProcessing.MaxFilesToPickup	When clustered processing is enabled, only a subset of the files in the InputFolder are picked up each time the folder is checked. This allows the other computers in the cluster to also pick up files to process. The number of files picked up defaults to the NumberOfDocumentsInParallel settings in

Key	Value
	the General settings section of the application configuration file, and can be overridden here.
CompletedFolder	This is optional. If included in the settings the source files and folders that are dropped into the InputFolder location are copied into this folder when the conversion is complete. If this setting is set to an empty string ("") or is not included in the settings the source files are deleted.
CopyInstructionsFromResources	PEERNET internal setting used to copy embedded text file containing instructions to the sample folders.
DCOMComputerName	When converting using a remote computer and DCOM, this setting is the name of the DCOM server where Document Conversion Service is running. See Setting up Client-Server Conversion for more information.
DeleteInputSubFolders	<p>When this is <i>true</i>, any folders dropped into the InputFolder for processing will be deleted when all of the files in the folder and its subfolders are converted. When set to <i>false</i>, all of the files in the folder will be converted but the folder structure will remain in the InputFolder.</p> <p>This setting is often used in conjunction with IncludeSubfolders.</p> <p>When Polling.MaxFilesToProcessAtATime is configured to limit the number of files picked up, this option is automatically set to <i>false</i>.</p>
Enabled	When this is <i>true</i> , the folder will be monitored. When set to <i>false</i> , this folder is not monitored. If this setting is not found, it defaults to <i>true</i> .
FailedFolder	If any file fails to convert, they are copied into a folder under this location. The folder name matches the name of the sub-folder created under the StagingFolder during processing.
IncludeSubfolders	If this value is <i>true</i> then any folders dropped into the InputFolder location will also be searched for files.
InputFolder	This is the folder that is watched for files (and folders if IncludeSubfolders is <i>true</i>)

Key	Value
	to convert.
NormalizeFileNames	<p>When <i>true</i>, file names passed in will be checked for normalization and normalized when necessary. This means that the new output file name, if not specified, will be the normalized file name, which while it may look identical to the original name, is actually not the same.</p> <p>This should be left as <i>false</i> unless you have problems converting files with foreign file name where some international characters are represented using diacritics. A diacritic is a glyph added to a letter; they are used to change the sound of the letter to which they are added. Some examples of a diacritic are the accent grave (') and acute (') in the French language.</p>
OutputFolder	The converted files are copied into this folder from the WorkingFolder when the conversion is done. This is done to prevent accidental pickup of partially created files.
PollingInterval	<p>Specifies the maximum wait period that can elapse between checking the input folder for files. This interval is in milliseconds, 15000 would poll the folder every 15 seconds.</p> <p>Each time a collection of files has completed conversion, the folder is immediately checked for files to convert to maintain throughput. If the folder is empty, the full polling interval will elapse before checking again.</p>
Polling.MaxFilesToProcessAtATime	<p>Allows the setting of a limit on the number of files that will be picked up from the InputFolder during any polling interval. When set to 0, no limit is imposed.</p> <p>This option is useful when the InputFolder is targeting an existing folder with a very large number of files. It allows the files to be processed in batches or groups instead of copying the entire folder structure to the WorkingFolder. This reduces the required amount of disk space used when processing files.</p> <p>When the number of files picked up is limited, the option DeleteInputSubFolders is automatically set to <i>false</i>.</p>

Key	Value
Polling.SynchronousFilePickup	<p>When set to <i>true</i>, the Watch folder will not pick up any files from the InputFolder until the current batch, or group, of files has completed processing.</p> <p>Used in conjunction with Polling.MaxFilesToProcessAtATime to control the flow of files so that a very large group of files can be processed as many smaller batches without overloading the physical disk space of the computer.</p>
Polling.NetworkShareRefreshInterval	<p>Each time a collection of files has completed conversion, the folder is immediately checked for files to convert to maintain throughput. This default behavior may not give slower network shares enough time to refresh and propagate file and directory changes.</p> <p>Use this value to set a wait period interval in milliseconds. This interval applies only when the InputFolder is a network share, not a local drive. It defaults to 0ms (no waiting) unless set.</p>
OutputFolder.MaintainInputFolderStructure	<p>When set to <i>false</i>, this flattens any input folder structures, storing all outputs in a single output folder. This may cause file overwrites due to file name collisions; use OutputFolder.PrependUniqueGUIDToFilename or to OutputFolder.AppendUniqueGUIDToFile name prevent this.</p> <p>This settings also overrides the creation of the MSG extraction subdirectory processing MSG files using the PreprocessArchiveFormatsFilter setting.</p> <p>Default is <i>true</i>, to maintain the directory structure.</p>
OutputFolder.PrependUniqueGUIDToFilename	<p>When set to <i>true</i>, a <i>Globally Unique ID</i>, or GUID, is added at the beginning of the file name and an optional separator string, OutputFolder.UniqueGUIDSeparatorCharacter, can be placed between the GUID and the filename. This setting can be used instead of or in addition to OutputFolder.AppendUniqueGUIDToFile name.</p>

Key	Value
	<p>4c4636f7-e9c5-4c4c-97af-081a328ba7c5_Filename.tif</p> <p>There is also an option, OutputFolder.RemoveHyphensFromGUID, to remove the dashes from the GUID.</p> <p>This settings defaults to <i>false</i>.</p>
OutputFolder.AppendUniqueGUIDToFilename	<p>When set to <i>true</i>, a <i>Globally Unique ID</i>, or GUID, is added at the end of the file name before the extension and an optional separator string, OutputFolder.UniqueGUIDSeparatorCharacter, can be placed between the filename and the GUID. This settings can be used instead or or in addition to OutputFolder.PrependUniqueGUIDToFilename.</p> <p>Filename_4c4636f7-e9c5-4c4c-97af-081a328ba7c5.tif</p> <p>There is also an option, OutputFolder.RemoveHyphensFromGUID, to remove the dashes from the GUID.</p> <p>This settings defaults to <i>false</i>.</p>
OutputFolder.RemoveHyphensFromGUID	<p>This setting defaults to <i>false</i>, meaning the hyphens, or dashes are included in the GUID in the file name. Set this to <i>true</i> to remove them.</p>
OutputFolder.UniqueGUIDSeparatorCharacter	<p>This sets the string that defines the separator character, or characters that will be placed between the GUID and the file name. It defaults to an underscore (-) character. Any invalid filename characters in this string will cause it to default back to a single underscore.</p> <p>To not have a separator between the GUID and the filename, leave this string empty, or comment it out.</p>
PreprocessArchiveFormatsFilter	<p>This setting filters, by file extension, what archive formats will be preprocessed before converting. Currently the only valid extension is "*.msg" for Outlook Message archive files. This setting can be disabled by commenting it out, or passing an empty string.</p>

Key	Value
PreprocessArchive.IncludeExtensionInFolderName	Control whether or not the .msg file extension is included in the name used to create the subfolder that will hold the message and attachments for processing. This is set to <i>True</i> by default and it is recommended to leave it set to prevent output file naming collisions.
PreprocessArchive.MSG.IncludeInlineAttachments	Message attachments can be inline (pasted into the email body) or attached as separate files. Images used in signatures are often inline attachments. Set this to <i>false</i> to not include inline attachments; note that this will also cause inline attached documents to not be processed. Default is <i>true</i> . This setting is applied before the attachment filtering below.
PreprocessArchive.MSG.AttachmentsIncludeFilter	Allows you filter what attachments will be processed. When set to an empty string, all attachments are processed. To filter for specific file types, enter in the extensions for each type separated by the pipe () character, such as "*.doc *.docx *.pdf".
PreprocessArchive.MSG.AttachmentsExcludeFilter	Allows you filter what attachments will not be processed. This option is applied after checking the include filter above. When set to an empty string, all attachments are processed. To filter for specific file types, enter in the extensions for each type separated by the pipe () character, such as "*.doc *.docx *.pdf".
RunAtEnd.Fail.Enabled	Set this to <i>true</i> to run the specified command on the original file if the conversion fails. Default is <i>false</i> .
RunAtEnd.Fail.Command	The full path to the command to be executed without arguments. Default is an empty string, no command to run.
RunAtEnd.Fail.Parameters	<p>The parameters for the command. Use the HTML code to put the command in quotes if there are spaces, and to enclose parameters. The following variables are available to pass arguments to the command.</p> <p>\$(FailedFilePath) - this is the path to the original file in its failed location.</p> <p>\$(SourceFileName) - this is the file name of the original file.</p>
RunAtEnd.Fail.StartDirectory	The directory in which to run the command. Default is an empty string.

Key	Value
RunAtEnd.Fail.WindowState	<p>The state of the command window when it is run.</p> <p>Normal - display the window in its normal state. Min - display the window minimized to the taskbar Max - display the window maximized. Hidden - do not show the window. (Default)</p>
RunAtEnd.Fail.WaitMode	<p>Optionally wait for the command to complete before continuing. The default is to not wait.</p> <p>If WaitForCompletion or WaitWithExitCode is chosen, the RunAtEnd.Fail.WaitModeMaxTime value is always used to stop the command if it has not returned after the set amount of time.</p> <p>WaitForCompletion - wait for the command to complete before continuing. WaitWithExitCode - waits for the command to complete and emits the exit code in the log. DoNotWait - does not wait for the command to complete. (Default)</p>
RunAtEnd.Fail.WaitModeMaxTime	<p>The maximum amount of time to wait for the command being run to complete. Default is 3 minutes.</p>
RunAtEnd.Success.Enabled	<p>Set this to <i>true</i> to run the specified command on each of the created files if the conversion succeeds. Default is <i>false</i>.</p>
RunAtEnd.Success.Command	<p>The full path to the command to be executed without arguments. Default is an empty string, no command to run.</p>
RunAtEnd.Success.Parameters	<p>The parameters for the command. Use the HTML code <code>&quot;</code> to put the command in quotes if there are spaces, and to enclose parameters. The following variables are available to pass arguments to the command.</p> <p>\$(OutputFilePath) - this is the path to the converted file. \$(SourceFileName) - this is the file name of the original file.</p>
RunAtEnd.Success.StartDirectory	<p>The directory in which to run the command. Default is an empty string.</p>

Key	Value
RunAtEnd.Success.WindowState	<p>The state of the command window when it is run.</p> <p>Normal - display the window in its normal state. Min - display the window minimized to the taskbar Max - display the window maximized. Hidden - do not show the window. (Default)</p>
RunAtEnd.Success.WaitMode	<p>Optionally wait for the command to complete before continuing. The default is to not wait.</p> <p>If WaitForCompletion or WaitWithExitCode is chosen, the RunAtEnd.Success.WaitModeMaxTime value is always used to stop the command if it has not returned after the set amount of time.</p> <p>WaitForCompletion - wait for the command to complete before continuing. WaitWithExitCode - waits for the command to complete and emits the exit code in the log. DoNotWait - does not wait for the command to complete. (Default)</p>
RunAtEnd.Success.WaitModeMaxTime	<p>The maximum amount of time to wait for the command being run to complete. Default is 3 minutes.</p>
SearchFilter	<p>A file extension based filter for file matching. By default it is set to *.* to match all files. A filter of *.pdf would only search for PDF documents.</p>
StagingFolder	<p>This folder is a holding location for the files during conversion. When the input folder is polled, each group of files is copied into a uniquely named sub-folder (based on date and time) under this folder. If IncludeSubfolders is <i>true</i> folders are also copied.</p>
TestMode	<p>This flag should be <i>false</i> or removed completely on a production system. Used for development purposes, this flag can be used to simulate load testing by copying the converted files back into the input folder. This value is ignored when clustered processing is enabled.</p>

Key	Value
UseTimeDateSubFoldersInCompletedFolder	<p>When set to <i>true</i>, each set of completed files are stored in a subfolder under the CompletedFolder. This subfolder is named using the date and time the files were picked up from the InputFolder.</p> <p>When set to <i>false</i> the files are copied directly into the CompletedFolder. If a file of the same name already exist in the folder it will be overwritten.</p> <p>This option is not used when the CompletedFolder is set to an empty string ("") or is not included in the settings.</p>
UseTimeDateSubFoldersInFailedFolder	<p>When set to <i>true</i>, any files that fail to convert are stored in a subfolder under the FailedFolder. This subfolder is named using the date and time the files were picked up from the InputFolder.</p> <p>When set to <i>false</i> the files are copied directly into the FailedFolder. If a file of the same name already exist in the folder it will be overwritten.</p>
WorkingFolder	<p>The output files are first created in this folder before being copied to the OutputFolder. If the files were created directly in the OutputFolder and another program was monitoring that folder the files could be picked up before the file was created. This two-stage process eliminates that problem.</p>

The Output File Settings

The `<WatchFolder>` section is also responsible for the type of output that is created. Common settings that would appear here would be:

- what type of file to create (multipaged or serialized TIFF, PDF files, JPEG images)
- the resolution (DPI) of any images created
- create color or black and white files
- create fax mode TIFF images.

The settings are provided as a set of name-value pairs based on the settings outlined in [Conversion Settings](#). In this sample application the conversion setting strings are stored in the configuration file for the application. These settings are read from the configuration file and then passed to the `PNDocConvQueueServiceLib` object through its COM interface. Having the conversion settings external to the program allows the settings to be changed without having to recompile.

The `<WatchFolder>` sample below creates multipaged, color-optimized TIFF files at 300 DPI with Group4 compression. See the sample `WatchFolder` sections provided in the configuration file for more examples of configurations of common output formats.



Code Sample - Output File Settings

```
<WatchFolders>

<!-- This watch folder creates 300 DPI Optimized TIFF Images -->
<WatchFolder Name="ConvertToTIFF Watch Folder">

  <Settings>
    ....

    <!-- Output file options -->
    <add Name="Devmode settings; Resolution" Value="300"/>

    <add Name="Save; Output File Format" Value="TIFF Multipaged"/>
    <!-- Replace the above with this to create serialized images. -->
    <!-- <add Name="Save; Output File Format" Value="TIFF Serialized"/> -->

    <add Name="Save; Append" Value="0"/>
    <add Name="Save; Color reduction" Value="Optimal"/>
    <add Name="Save; Dithering method" Value="Halftone"/>

    <!-- This creates file.ext.tif, change to 1 to create file.tif-->
    <add Name="Save; Remove filename extension" Value="0" />

    <add Name="TIFF File Format; BW compression" Value="Group4"/>
    <add Name="TIFF File Format; Color compression" Value="LZW RGB"/>
    <add Name="TIFF File Format; Indexed compression" Value="LZW"/>
    <add Name="TIFF File Format; Greyscale compression" Value="LZW"/>
    <add Name="JPEG File Format; Color compression" Value="Medium Quality"/>
    <add Name="JPEG File Format; Greyscale compression" Value="High
Qual i ty"/>

  </Settings>
</WatchFolder>

</WatchFolders>
```

Changing the File Extension to Converter Mapping

The file extension of each file is used to determine what converter is used to convert that file. File extensions can be added, removed and changed as needed. When the converter requires a native application to be installed to do the conversion, that application must also be installed.

The mapping consists of the extension (the suffix of the file name past the last *dot* or *period* in file's name) and a semi-colon separated list of converter names. See [What Files Can I Convert?](#) for a list of converter names.

In some cases the file extension may only have one converter that can process that type of file, and in others, such as PDF, have more than one option. To convert a PDF file you can choose between our built-in Adobe PDF converter, Adobe Reader, GhostScript or Outside-In AX. The code sample below shows a small snippet of the file mapping in the configuration file.

If you want to by-pass certain file types, say for instance you are creating TIFF images and you want to skip converting any TIFF images that are dropped into the input folder, you can change the file extension mapping to have files with the *.tif* extension sent to the PEERNET Passthrough converter. See [Skipping Files with the Passthrough Converter](#) for more details.

The default configuration file for Watch Folder Service lists all of the file extensions to converter mappings in the Settings section at the bottom of the files. Any of these mappings can also be placed inside a WatchFolder section to customize the file extension mappings for that folder.

An example why you would need this would be two WatchFolder sections for PDF to TIFF conversion: one that uses the default Adobe PDF - Builtin converter, and another one that uses Adobe Reader to convert PDF to TIFF instead.



Code Sample - File Extension to Converter Mapping

```
<WatchFoldersSection>

<WatchFolders>
  ...
  <WatchFolder Name="PDF to TIFF with PEERNET Adobe Builtin">
    <Settings>
      <!-- Folder options -->
      <add Name="InputFolder" Value="C:\PEERNET\PDF_Builtin\Input"/>
      ...
      <add Name=".pdf" Value="Adobe PDF - Builtin;" />
    </Settings>
  </WatchFolder>

  <WatchFolder Name="PDF to TIFF with Adobe Reader">
    <Settings>
      <!-- Folder options -->
      <add Name="InputFolder" Value="C:\PEERNET\PDF_AdobeReader\Input"/>
      ...
      <add Name=".pdf" Value="Adobe Acrobat Reader" />
    </Settings>
  </WatchFolder>
</WatchFolders>

<Settings>
  <!-- File Extension to Converter Mapping -->
  <!-- These can be added to the Settings section for each WatchFolder -->
  <!-- to tailor each WatchFolder to use different converters for its -->
  <!-- documents. The individual settings take precedence over the -->
  <!-- global WatchFolderSection settings section -->
  <add Name=".doc" Value="Microsoft Word;Outside-In AX" />
  <add Name=".docx" Value="Microsoft Word;Outside-In AX" />

  <add Name=".xlsx" Value="Microsoft Excel;Outside-In AX" />
  <add Name=".xls" Value="Microsoft Excel;Outside-In AX" />

  <add Name=".pptx" Value="Microsoft PowerPoint;Outside-In AX" />
  <add Name=".ppt" Value="Microsoft PowerPoint;Outside-In AX" />

  <add Name=".pdf" Value="Adobe PDF - Builtin;Adobe Acrobat Reader;Ghostscript;Outside-In" />

  ...
  <add Name=".tif" Value="Image - Builtin;PEERNET Image Converter;Outside-In AX" />
  <add Name=".tiff" Value="Image - Builtin;PEERNET Image Converter;Outside-In AX" />
  <add Name=".bmp" Value="Image - Builtin;PEERNET Image Converter;Outside-In AX" />
  <add Name=".jpg" Value="Image - Builtin;PEERNET Image Converter;Outside-In AX" />
  <add Name=".jpeg" Value="Image - Builtin;PEERNET Image Converter;Outside-In AX" />
</Settings>

</WatchFoldersSection>
```

Long Path Name Support

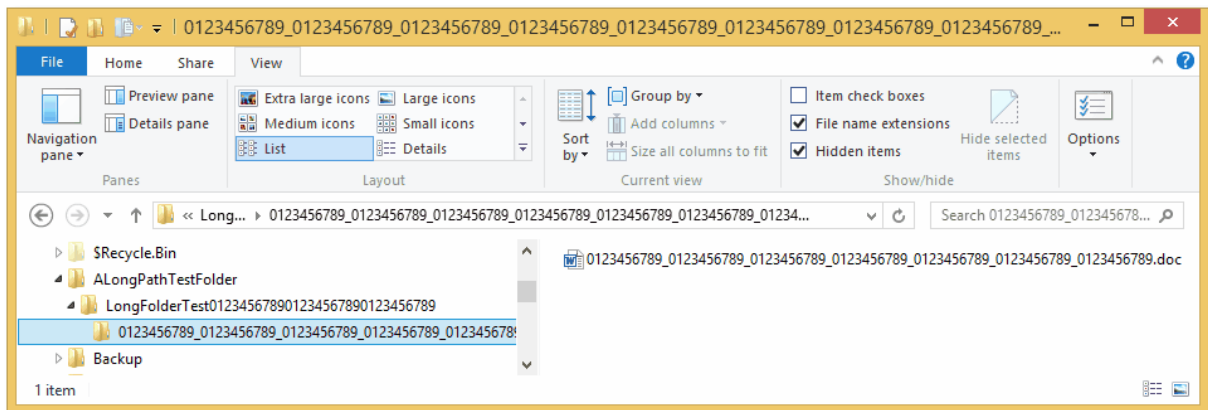
Historically, Windows (and before that, DOS) had a maximum path length (MAXPATH) of 260 characters. While this has changed over the years to allow file paths of up to 32,000 characters, many of the underlying components of Windows are still bound by the MAXPATH limitation.

Most of the time you never have to think about long path support but it does occasionally occur. A common situation would be having to convert all the files in a directory structure on network attached storage (NAS) created in UNIX or another file system where long paths are supported.

To handle this, Document Conversion Service and the Watch Folder Service support long path names for the input, output, failed and completed folders of a watch folder, as well as saving the results XML files and logging files.

The one caveat is that the files and directory structures copied to the staging and working folders to be processed need to be less than 255 characters. We can do this by keeping these paths as short as possible. This staging and working folder limitation is a requirement of the underlying programs, such as Adobe Reader and Microsoft Office, that Document Conversion Service uses to perform conversions. If the file path sent to Document Conversion Service to be converted is longer than MAXPATH that file will gracefully fail to convert.

Keep in mind that even if the input folder path itself is not greater than MAXPATH, the underlying subfolders and file names can create a path that is. You can see by this sample directory shown below that using `C:\LongPathTestFolder` as the input folder path will generate file paths longer than MAXPATH.



In this scenario, you can also set **UseCompressedDateTimeFormat** option to true to use a shorter version of the date-time stamp named subfolder in the working, staging, completed and failed folders. The default creates an easier to read folder name similar to *Thursday_March_31_2016_10_16_32_AM*, while the condensed date-time stamp is strictly numerical and similar to *20160331131645*.

A sample WatchFolder configuration is shown below.



Code Sample

```

<!-- This watch folder creates 300 DPI Optimized TIFF Images -->
<WatchFolder Name="ConvertToTIFF Watch Folder">
  <Settings>
    <!-- Folder options -->
    <add Name="Enabled" Value="true" />
    <add Name="InputFolder" Value="C:\ALongPathToTestFolder\"/>
    <add Name="SearchFilter" Value="*.*)" />
    <add Name="IncludeSubFolders" Value="True"/>
    <add Name="DeleteInputSubFolders" Value="True"/>
    <add Name="StagingFolder" Value="C:\PN\S"/>
    <add Name="WorkingFolder" Value="C:\PN\W"/>
    <add Name="FailedFolder" Value="C:\ALongPathToTestFolderResults\Failed"/>
    <add Name="CompletedFolder" Value="C:\ALongPathToTestFolderResults\Completed"/>
    <add Name="OutputFolder" Value="C:\ALongPathToTestFolderResults\Output"/>
    <add Name="PollingInterval" Value="15000"/>
    <add Name="DCOMComputerName" Value="localhost"/>
    <add Name="TestMode" Value="false"/>
    <add Name="NormalizeFileNames" Value="false"/>
    <add Name="CopyInstructionsFromResources" Value="ReadMe_ConvertToTIFF"/>

    <!-- 0 means no limit -->
    <add Name="Polling.MaxFilesToProcessAtATime" Value="0" />
    <add Name="Polling.SynchronousFilePickup" Value="false" />

    <add Name="UseTimeDateSubFoldersInCompletedFolder" Value="true" />
    <add Name="UseTimeDateSubFoldersInFailedFolder" Value="true" />
    <add Name="UseCompressedDateTimeFormat" Value="true" />

    <!-- Output file options -->
    <add Name="Devmode settings;Resolution" Value="300"/>

    <add Name="Save;Output File Format" Value="TIFF Multipaged"/>
    <!-- Replace the above with this to create serialized images. -->
    <!-- <add Name="Save;Output File Format" Value="TIFF Serialized"/> -->

    <add Name="Save;Append" Value="0"/>
    <add Name="Save;Color reduction" Value="Optimal"/>
    <add Name="Save;Dithering method" Value="Halftone"/>

    <!-- This creates file.ext.tif, change to 1 to create file.tif-->
    <add Name="Save;Remove filename extension" Value="0" />

    <add Name="TIFF File Format;BW compression" Value="Group4"/>
    <add Name="TIFF File Format;Color compression" Value="LZW RGB"/>
    <add Name="TIFF File Format;Indexed compression" Value="LZW"/>
    <add Name="TIFF File Format;Greyscale compression" Value="LZW"/>
    <add Name="JPEG File Format;Color compression" Value="Medium Quality"/>
    <add Name="JPEG File Format;Greyscale compression" Value="High Quality"/>
  </Settings>
</WatchFolder>

```

High Performance Clustering and Fail Over Conversion

New for Document Conversion Service 3.0.010 is high-performance clustering and failover management within the Watch Folder Service.

What is Clustering?

Clustering allows you to install Document Conversion Service on more than one computer, point each computer at the same group of files, and have all the computers working together to convert the files in that folder. This can greatly increase your conversion performance. If you are dealing with large sets of files to be converted, clustering is an easy way to increase your conversion speed and keep your data centralized.

You will need a separate license of Document Conversion Service for each computer you plan to use in the cluster.

High Availability and Failover Support

A side benefit of clustering is failover, or high availability support. As more than one computer is actively converting documents, if one computer has to be restarted or brought off line while other maintenance is performed, the other computers watching the clustered folder are still running and converting until the first one is back up and running again.

Clustering with Watch Folder Service

The Watch Folder Service includes a sample Watch Folder section, *Clustered ConvertToTIFF Watch Folder*, that is pre-configured for clustered processing. This watch folder section uses a network share folder, C:\PEERNET\WatchFolders\CLUSTERED, that is created as part of the Document Conversion Service install. See [Clustering - Use the PEERNET CLUSTERED Share Folder](#) for steps on setting up this type of clustered processing.

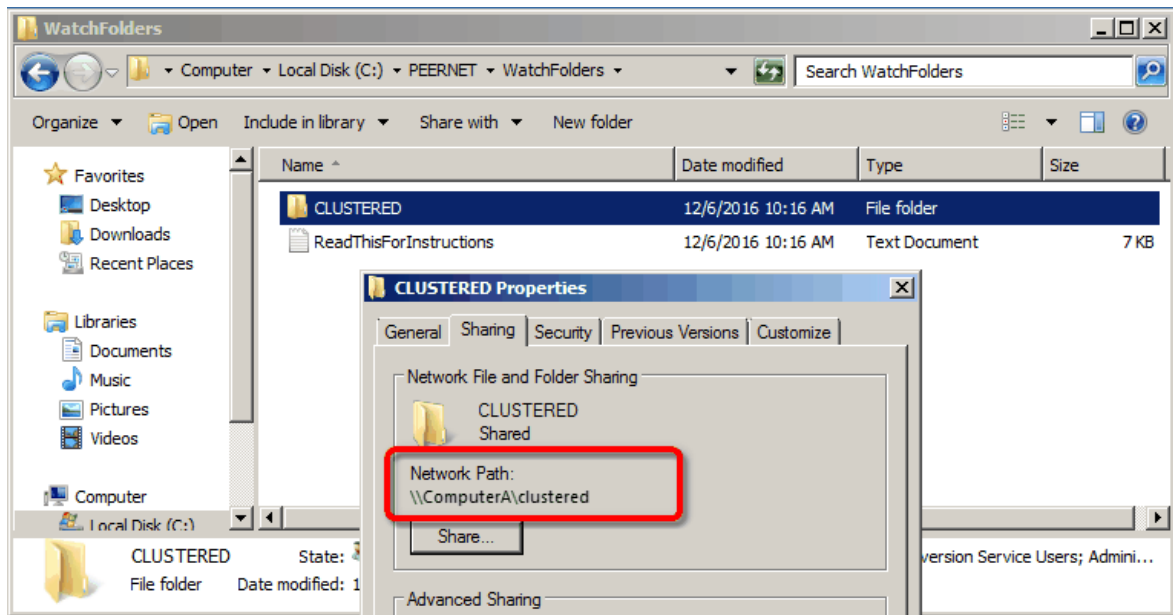
A more common approach is to have Document Conversion Service and Watch Folder Service installed on several computers and watching a network share that is separate from any of the computers in the cluster. See [Clustering - Using an External Network Share](#) for instructions on setting up clustering in this environment.

Clustering - Use the PEERNET CLUSTERED Share Folder


In this scenario, the shared folder that contains the files and/or folders to be processed is on the first computer in the cluster. All of the other computers simply point to the shared folder and process the files from there. The key here is to install Document Conversion Service and create the DCSAdmin account with the same user name and password on all computers in the cluster.

Setting up the First Node in the Cluster

1. Install Document Conversion Service and when prompted, allow the install to create the local DCSAdmin administrative account. Keep note of the password used when creating the DCSAdmin as you will need to use the same password on all the other computers.
2. The install will create a network shared folder, C:\PEERNET\WatchFolders\CLUSTERED.



3. The Watch Folder Service contains a sample Watch Folder configuration using this folder for clustered processing. Leave this configuration as set. The input location, `C:\PEERNET\WatchFolders\CLUSTERED\ConvertToTIFF\Input`, is where you will copy the files to be processed.



Code Sample - Clustered Conversion for Base Node

```
<WatchFolders>
  <!-- This watch folder is configured for clustered processing -->
  <!-- it creates 300 DPI Optimized TIFF Images -->
  <WatchFolder Name="Clustered ConvertToTIFF Watch Folder">
    <Settings>

      <!-- The InputFolder, FailedFolder, CompletedFolder and OutputFolder can point
      <add Name="InputFolder"
            Value="C:\PEERNET\WatchFolders\CLUSTERED\ConvertToTIFF\Input"/>
      <add Name="SearchFilter" Value="*.*/>
      <add Name="IncludeSubFolders" Value="True"/>
      <add Name="DeleteInputSubFolders" Value="True"/>
      <add Name="FailedFolder"
            Value="C:\PEERNET\WatchFolders\CLUSTERED\ConvertToTIFF\Failed"/>
      <add Name="CompletedFolder"
            Value="C:\PEERNET\WatchFolders\CLUSTERED\ConvertToTIFF\Completed"/>
      <add Name="OutputFolder"
            Value="C:\PEERNET\WatchFolders\CLUSTERED\ConvertToTIFF\Output"/>

      <!-- Keep these folders on separate computers for clustering. -->
      <add Name="StagingFolder"
            Value="C:\PEERNET\WatchFolders\CLUSTERED\ConvertToTIFF\Staging"/>
      <add Name="WorkingFolder"
            Value="C:\PEERNET\WatchFolders\CLUSTERED\ConvertToTIFF\Failed"/>

      ...

      <!-- Clustered Processing -->
      <!-- This forces batch mode processing with synchronous wait and -->
      <!-- no date time stamp used in the Failed\Completed folders -->
      <add Name="ClusteredProcessing.Enabled" Value="true"/>
      <!-- Override this for clustering to customize pickup -->
      <!-- <add Name="ClusteredProcessing.MaxFilesToPickup" Value="4"/> -->
      ...

    </Settings>
  </WatchFolder>
</WatchFolders>
```

4. Start Document Conversion Service and Watch Folder Service on this computer.

Setting up the Other Nodes

For all the other computers you want in the cluster, do the following.

1. Install Document Conversion Service and when prompted, allow the install to create the local DCSAdmin administrative account. Use the same password used when the first node in the cluster above. It is this matching account, as well as the shared network drive that allows the clustered processing to take place. If the passwords do not match, clustering will not work.
2. The install will also create a network shared folder, C:\PEERNET\WatchFolders\CLUSTERED on this computer but on this node, the shared folder is only used to keep the staging and working folders for each node separate.
3. Open the watch folder configuration file in [DCS Editor](#) by going to **DCS Dashboard - Watch Folder Settings - Edit Configuration**, or by going to Start - All Programs - PEERNET Document Conversion Service 3.0 - Watch Folder - Configure Watch Folder Settings.

4. Scroll to the *Clustered ConvertToTIFF Watch Folder* section, or use the [Find and Replace tool](#) to find the section in the file. Set the **InputFolder** setting to use the shared computer path to the first node in the cluster instead of the hard drive on this computer.

 Code Sample - Clustered Conversion

```
<WatchFolders>
  <!-- This watch folder is configured for clustered processing -->
  <!-- it creates 300 DPI Optimized TIFF Images -->
  <WatchFolder Name="Clustered ConvertToTIFF Watch Folder">
    <Settings>

      <!-- The InputFolder, FailedFolder, CompletedFolder and OutputFolder can point
      <add Name="InputFolder" Value="\\ComputerA\CLUSTERED\ConvertToTIFF\Input"/>
      <add Name="SearchFilter" Value="*.*/>
      <add Name="IncludeSubFolders" Value="True"/>
      <add Name="DeleteInputSubFolders" Value="True"/>
      <add Name="FailedFolder" Value="\\ComputerA\CLUSTERED\ConvertToTIFF\Failed"/>
      <add Name="CompletedFolder" Value="\\ComputerA\CLUSTERED\ConvertToTIFF\Comple
      <add Name="OutputFolder" Value="\\ComputerA\CLUSTERED\ConvertToTIFF\Output"/>

      <!-- Keep these folders on separate computers for clustering. -->
      <add Name="StagingFolder"
        Value="C:\PEERNET\WatchFolders\CLUSTERED\ConvertToTIFF\Staging"/>
      <add Name="WorkingFolder"
        Value="C:\PEERNET\WatchFolders\CLUSTERED\ConvertToTIFF\Failed"/>

      ...

      <!-- Clustered Processing -->
      <!-- This forces batch mode processing with synchronous wait and -->
      <!-- no date time stamp used in the Failed\Completed folders -->
      <add Name="ClusteredProcessing.Enabled" Value="true"/>
      <!-- Override this for clustering to customize pickup -->
      <!-- <add Name="ClusteredProcessing.MaxFilesToPickup" Value="4"/> -->
      ...

    </Settings>
  </WatchFolder>
</WatchFolders>
```

5. If desired, you can use the setting `ClusteredProcessing.MaxFilesToPickup` to customize how many files at a time are picked up by each computer. This allows you to offload processing to the faster computers, but still provide you with fail over protection if one of the computers in the cluster goes down.
6. Save the file; the [DCS Editor](#) will validate the file when saving and prompt to resolve any syntax errors.
7. Start Document Conversion Service and Watch Folder Service on this computer.
8. Repeat these steps to add more computers to the cluster.

Starting Conversion

Once all the nodes in the cluster have configured, and Document Conversion Service and Watch Folder Service are started on each computer, you can then start dropping files into the `C:\PEERNET\WatchFolder\CLUSTERED\ConvertToTIFF\Input` on the first computer, ComputerA, for conversion.

Each computer in the cluster will check the `InputFolder` for files to process and will pick up a subset of files to process. The number of files picked up defaults to the *NumberOfDocumentsInParallel* settings in the General settings section of the application configuration file but can be overridden in each individual watch folder section using the *ClusteredProcessing.MaxFilesToPickup* setting.

Clustering - Using an External Network Share

A more common approach would be an existing network share and several computers (or virtual machines) all looking at the same location on the share drive for files to process. With this approach, you will need an account that has access to the network share that can be used to run the Watch Folder Service on each computer, and Document Conversion Service installed on all of the computers in the cluster using the local `DCSAdmin` account created during installation.

Setting up the Network Share

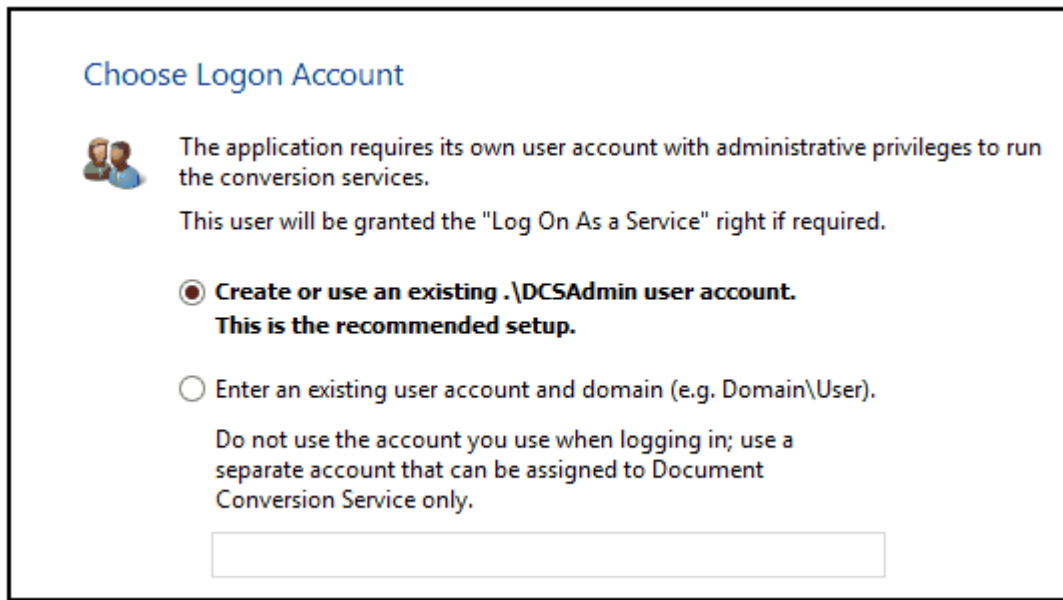
On the network share you will need four folders as shown below. The network share names here are just sample names; replace these with your actual network share name and paths.

Folder on Network	Watch Folder Setting	Description
\\NetworkShareA\\Clustered\\Input	InputFolder	This is the folder that is watched for files (and folders if IncludeSubfolders is <i>true</i>) to convert.
\\NetworkShareA\\Clustered\\Output	OutputFolder	The converted files are copied into this folder when the conversion is done.
\\NetworkShareA\\Clustered\\Failed	FailedFolder	If any file fails to convert, they are copied into a folder under this location.
\\NetworkShareA\\Clustered\\Completed	CompletedFolder	This is optional. If set, the source files and folders that are dropped into the InputFolder location are copied into this folder when the conversion is complete. If this setting is set to an empty string ("") or is not included is the settings the source files are deleted.

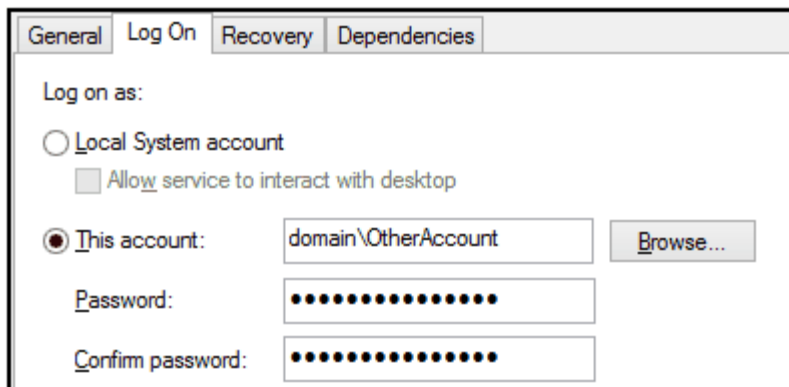
Setting up the Computers

The following steps need to be done for each computer you want as part of the cluster.

1. Install Document Conversion Service and, when prompted, allow the install to create the local `DCSAdmin` administrator account.



2. Go to Start - Control Panel - System and Security - Administrative Tools - Services (or type "Services" into the search field on the Start menu). The Watch Folder Service Log On credentials need to be changed to use the domain or other account that has access to the network share location. This is critical as the Watch Folder Service will run under this account and needs to have full access to the network share to be able to read/write and lock the files as part of the clustered conversion. The setup initially sets the service to use the DCSAdmin as part of the install.
3. In the Services control panel applet, locate the service PEERNET Watch Folder Service and double-click it to open its Properties dialog.
4. On the *Log On* tab, set the service account to the domain or other account that has access to the network share. This account will also need the *Logon As A Service* right. This right is automatically granted through the services panel when possible, otherwise talk to your IT Admin to add this privilege to the account.



5. Click Apply and close the Services panel. Do not start the service at this point!

6. Open the Watch Folder Service configuration file in the [DCS Editor](#) by going to DCS Dashboard - Watch Folder Settings - Edit Configuration, or by going to Start - All Programs - PEERNET Document Conversion Service 3.0 - Watch Folder - Configure Watch Folder Settings.
7. Find and edit the *Clustered ConvertToTIFF Watch Folder* section to use the network share path for its InputFolder, OutputFolder and FailedFolder. If you are using the CompletedFolder, set the path for that as well. Keep the StagingFolder and WorkingFolder local to each computer in the cluster.



Code Sample - Clustered Conversion

```
<WatchFolders>
  <!-- This watch folder is configured for clustered processing -->
  <!-- it creates 300 DPI Optimized TIFF Images -->
  <WatchFolder Name="Clustered ConvertToTIFF Watch Folder">
    <Settings>

      <!-- The InputFolder, FailedFolder, CompletedFolder and OutputFolder can point
      <add Name="InputFolder" Value="\\NetworkShareA\Clustered\Input"/>
      <add Name="SearchFilter" Value="*.*/>
      <add Name="IncludeSubFolders" Value="True"/>
      <add Name="DeleteInputSubFolders" Value="True"/>
      <add Name="FailedFolder" Value="\\NetworkShareA\Clustered\Failed"/>
      <add Name="CompletedFolder" Value="\\NetworkShareA\Clustered\Completed"/>
      <add Name="OutputFolder" Value="\\NetworkShareA\Clustered\Output"/>

      <!-- Keep these folders on separate computers for clustering. -->
      <add Name="StagingFolder"
        Value="C:\PEERNET\WatchFolders\CLUSTERED\ConvertToTIFF\Staging"/>
      <add Name="WorkingFolder"
        Value="C:\PEERNET\WatchFolders\CLUSTERED\ConvertToTIFF\Failed"/>

      ...

      <!-- Clustered Processing -->
      <!-- This forces batch mode processing with synchronous wait and -->
      <!-- no date time stamp used in the Failed\Completed folders -->
      <add Name="ClusteredProcessing.Enabled" Value="true"/>
      <!-- Override this for clustering to customize pickup -->
      <!-- <add Name="ClusteredProcessing.MaxFilesToPickup" Value="4"/> -->
      ...

    </Settings>
  </WatchFolder>
</WatchFolders>
```

8. If desired, you can use the setting ClusteredProcessing.MaxFilesToPickup to customize how many files at a time are picked up by each computer. This allows you to offload processing to the faster computers, but still provide you with fail over protection if one of the computers in the cluster goes down.
9. Save the file; the [DCS Editor](#) will validate the file when saving and prompt to resolve any syntax errors.
10. Start Document Conversion Service and Watch Folder Service on this computer.
11. Repeat these steps to add more computers to the cluster.

OCR Images and Scanned PDF Files to Searchable PDF

Starting with Document Conversion Service 3.0.031, the Watch Folder Service now includes a sample folder, **OCR to AdobePDF Watch Folder**, that will use Optical Character Recognition (OCR) on scanned PDF files and images to create searchable PDF files.

Optical Character Recognition searches for and recognizes text (characters) on scanned pages or images and extracts it as digital text. Outside factors such as image quality, the font used, and any image background on the pages will all affect the validity of the OCR results.

The PDF files created using OCR consist of each page embedded as an image, with the page text as an invisible layer over the top of the image. The invisible text layer can be searched and text content can be copied from the PDF if permissions permit.

Optical Character Recognition can only be used when creating PDF files. OCR will increase the processing time for file conversion and is supported by the following converters:

- Built-in PDF Converter
- Built-in Image Converter



Caution

This feature is not supported on Microsoft® Windows Server 2008 R2 and Microsoft® Windows 7.

OCR Languages And Adding Additional Languages

When recognizing text, the OCR engine has to know which languages to look for on the page. OCR works by analyzing the patterns, shapes, and curves of the text characters on the page and matching them to predefined information for different characters in each language. It assigns a confidence score for each language, with the highest score determining the language chosen.

Document Conversion Service comes with files to support recognizing *Arabic, English, French, German, Hebrew, Hindi, Italian, and Spanish*.

To download individual language files, go to [Tesseract Languages Code and Traineddata Files](#). This link also includes a table listing the language code for each traineddata file for each language. You can download complete sets of language files by going to [Traineddata Files for Tesseract](#).

To add them to Document Conversion Service, copy the desired *.**traineddata** files into the following folder:

```
%PROGRAMDATA%\PEERNET\Document Conversion Service\tessdata
```

Enabling OCR and Page Selection

OCR is disabled to start until the options are added to your watch folder definition. The sample watch folder already has these options set.

ConverterPlugIn.PNBuiltinsOCRPDF.Enabled

Set this to 1 to enable OCR, 0 to turn it off. Default value is 0.

ConverterPlugIn.PNBuiltinsOCRPDF.FirstPageOnly

Set this to 1 to only OCR the first page of any document. Set it to 0 or do not set it to OCR each page in the document. Default is **0**.

Setting Languages

The OCR engine needs to know which languages you want to try to recognize on the page. The more languages listed the longer the OCR process will take as it tries to match each character against each language listed.

ConverterPlugin.PNBuiltinsOCRPDF.Languages

To run OCR on your text and look for multiple languages, list the language code for each language you want, separated by a plus sign. For example, the sample watch folder looks only for English, "**eng**". To look for English, French, and Spanish, you would use the string "**eng+fra+spa**". The default when this is not supplied is English only, "**eng**".

The language codes for the provided languages are as follows.

Language	Language Code
Arabic	ara
English	eng
French	fra
German	deu
Hebrew	heb
Hindi	hin
Italian	ita
Spanish	spa

Processing Outlook and EML Mail Messages and Attachments

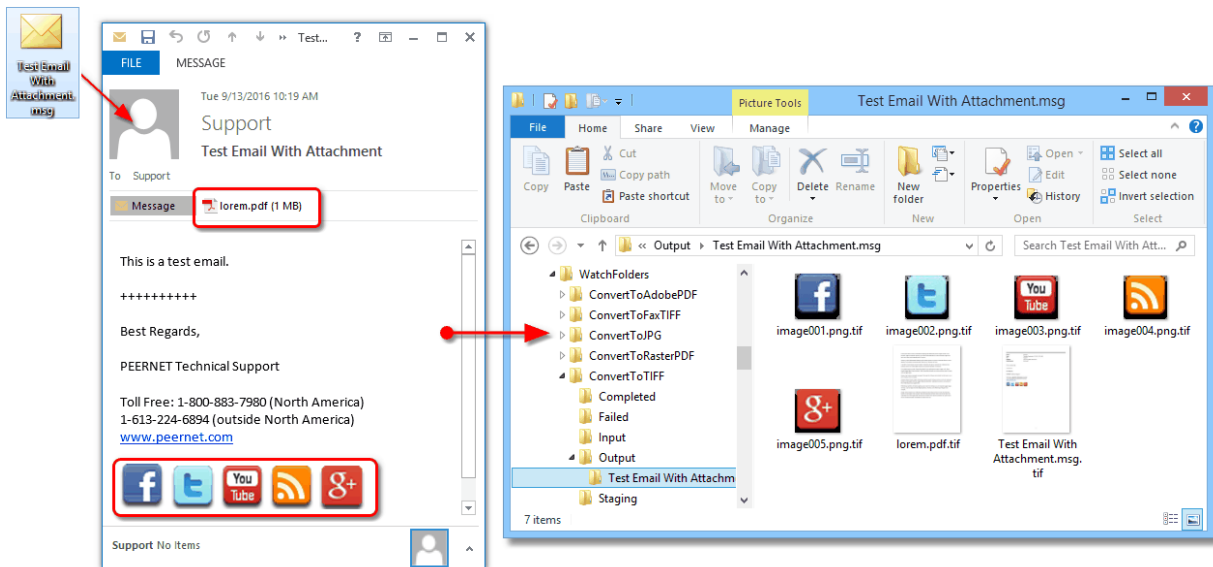
Starting with Document Conversion Service 3.0.009, the Watch Folder Service includes the ability to extract and convert any attachments in Outlook Message files (*.msg) as well converting the Outlook message file itself.

In Document Conversion Service 3.0.029, support for Electronic Mail messages (*.eml) files with attachments was added. EML is the standard file extension for email files stored using the *Internet Message Format protocol*. This format complies with the RFC 5322 industry standard. They can be single messages, or include attachments.

When this option is enabled, the file is checked for attachments and if any are found, the original message file and all of its attachments are converted. The initial settings have the resulting files placed into the **OutputFolder** under a sub folder of the same name as the original file. If any attachments are not of a file type supported by Document Conversion Service, the attachment will not be converted and is placed in the **Failed** folder.

The original message and all attached files and embedded images in the e-mail and signature are processed. This includes recursively processing attachments that are Outlook Messages or EML files that themselves have attachments. All message content and file attachments are extracted into a sub folder of the same name as the original file. If any name collisions are detected, the file names are made to be unique by adding a number in brackets at the end. As an example, an email message with an attached PDF document named lorem.pdf and an attached message that also has an attached PDF document of the same name will create two files - lorem.pdf.tif and lorem(2).pdf.tif.

The sample message below, *Test Email With Attachments.msg*, contains a single attached PDF file, as well as 5 small images from the signature. When the MSG is processed, the original message file and all attachments are processed. The attached PDF file will retain its name, and the inline images that are part of the signature will be named image001 through to image005.



When processing the above message, the option to keep the original filename's extension as part of the new filename was enabled. This can be disabled using the setting `<add Name ="Save;Remove filename extension" Value ="1"/>`. When this option is disabled, the output file from a file named lorem.pdf would become lorem.tif instead of lorem.pdf.tif.

Several settings have been added to control email message file attachment processing. Each of the included pre-configured WatchFolders already have these new settings added with the attachment processing disabled. To enable attachment processing, simply uncomment the setting **PreprocessArchiveFormatsFilter**. To disable it, you can comment it out again, or set it as an empty string.

MSG and EML Extraction Subfolder Options

PreprocessArchive.IncludeExtensionInFolderName

Allows you to control whether or not the .msg or .eml file extension is included in the name used to create the subfolder that will hold the message and attachments for processing. In the screenshot above, the .msg file extension was kept as part of the subfolder name. To minimize possible name collision, we recommend leaving this option enabled.

PreprocessArchive.CreateAllOutputInSubfolder

Added in version 3.0.025, this option controls if the MSG or EML file and extracted attachments will be stored in a subfolder, or at the root of the output folder. To minimize possible name collision, we recommend leaving this option enabled unless you are certain of unique filenames, or are using [Unique File Naming and Flat Folder Structures](#) and the [MSG and EML Unique File Naming Options](#) below.

MSG and EML Extraction Filtering

The next three settings are specific to handling, or *filtering* what message file attachments actually get converted. They apply to both MSG and EML files.

The first setting determines if inline attachments are converted, and the second two settings allow for further filtering of what e-mail attachments will be processed. These *filtering* options are applied in the order of inline attachments, include filter and then finally exclude filter.

Most often only one of the include or exclude filter will be used at a time, depending on how you need to filter. It is easier to say exclude only "*.jpg" attachments, or include only "*.pdf" attachments than to write long, specific lists of all of the file types.

PreprocessArchive.MSG.IncludeInlineAttachments

Message attachments can be inline (pasted into the email body) or attached as separate files. Images used in signatures are often inline attachments, while a PDF file attached to the email would not be. You can disable the processing of all inline attachments by setting this value to false. As some inline attachments can actually be documents, setting this to *False* is not recommended. This setting is always checked first before the message attachment filtering settings below.

PreprocessArchive.MSG.AttachmentsIncludeFilter

Allows for filtering of what attachments will be processed. When set to an empty string, all attachments are processed. To filter for specific file types, enter in the extensions for each type separated by the pipe (|) character. For example, to only convert any attached Word and PDF documents, you could set this as `<add Name="PreprocessArchive.MSG.AttachmentsIncludeFilter" Value="*.doc|*.docx|*.pdf" />`. This setting is always applied after the inline attachment check above and before the exclude filter check below.

PreprocessArchive.MSG.AttachmentsExcludeFilter

The last filtering setting, and also the setting applied last, is the exclude filter, which determines what files (by extension) to not extract from the MSG. As with the include filter above, enter in the extensions for

each file type you do not want to be converted, separated by the pipe (|) character. When left as an empty string, no files are excluded.

MSG and EML Image Attachment Options

Image attachments are converted to the new format using the source image's resolution and not the requested output format resolution. This applies when converting to image as well as PDF files. Up-scaling images to a higher resolution can result in images that are many times larger than the actual source image. Other factors such as going from JPG to a lossless format like TIFF can also cause an increase in file size.


PreprocessArchive.MSG.ImageAttachmentsKeepSourceResolution

This defaults to true. Conversion options such as fax mode and other image option actions can override this. This setting overrides the ConverterPlugin.PNImageConverter.KeepSourceImageResolution setting and applies only to images extracted from an MSG or EML. Setting this to false may cause images to be very large.

Controlling Image Size With Compression

Another way to control the size of extracted and converted images is by setting the compression option for the output format. As an example, converting a JPG image such as a photograph from a camera to a TIFF image using the default settings of LZW compression will create a very large file.

To create a comparable TIFF image, we need to change the compression to one of the JPEG compression options for TIFF.



Code Sample - Controlling Image Size with Compression

```

<WatchFolders>

  <!-- This watch folder creates 300 DPI Optimized TIFF Images -->
  <WatchFolder Name="ConvertToTIFF Watch Folder">
    <Settings>
      <!-- Folder options -->
      ...

      <!-- Use JPEG compression in TIFF images for smaller files from photographs -->
      <add Name="TIFF File Format;BW compression" Value="Group4"/>
      <add Name="TIFF File Format;Color compression" Value="High quality JPEG"/>
      <add Name="TIFF File Format;Indexed compression" Value="High quality JPEG"/>
      <add Name="TIFF File Format;Greyscale compression" Value="High quality JPEG"/>

    </Settings>
  </WatchFolder>
</WatchFolders>

```

MSG and EML Unique File Naming Options

Added in version 3.0.025, these options add the ability to apply unique names for all MSG and EML and attachments as they are processed when using the existing [Unique File Naming and Flat Folder Structures](#) settings. If either or both of *OutputFolder.PrependUniqueGUIDToFilename* or *OutputFolder.AppendUniqueGUIDToFilename* are true, a **Globally Unique ID**, or **GUID** is added to the output filename for the MSG and any extracted attachments.

The default behavior is to use the same GUID in both the MSG subfolder (if using) and in all extracted and converted attachments.

PreprocessArchive.MSG.UseUniqueGUIDInMSGFolderName

Controls if a GUID is used in the MSG or EML folder name created to store the converted MSG or EML and attachments. Applies when *PreprocessArchive.CreateAllOutputInSubfolder* is *true*.

PreprocessArchive.MSG.UseSameGUIDForAllFiles

The default behavior is to use the same GUID in the folder and for all files extracted and converted into that folder. To use a random, unique GUID for each file, set this to *false*.

PreprocessArchive.MSG.UseMessageIDForPrependGUID

PreprocessArchive.MSG.UseMessageIDForAppendGUID


When enabled, any Message ID included in the source email header information is used in place of a randomly generated GUID when building the output folder and email and attachment filenames. If enabled and there is no Message ID in the email, the GUID is used instead.

PreprocessArchive.MSG.UseMessageIDWithoutFQDN

An email Message Id consists of a string of characters which is the unique identifier of this message from the mail server, and ends with ampersand (@) and the Fully Qualified Domain Name (FQDN) of the mail server that sent the message. By default, we set this option to true to only use the first part of the string.

Sample Watch Folder Setting for MSG and EML Extraction

The last set of highlighted settings shown below are not included in the pre-configured WatchFolder settings. These are some recommended settings to help control the size of the final output files when dealing with Outlook Messages with attached images and logos in the signatures.

 Code Sample - Default Outlook Message Processing

```
<WatchFolders>

  <!-- This watch folder creates 300 DPI Optimized TIFF Images -->
  <WatchFolder Name="ConvertToTIFF Watch Folder">
    <Settings>
      <!-- Folder options -->
      ...

      <!-- Preprocess Archive Settings -->
      <!-- Comment out or set this as empty string to disable MSG/EML archive processing -->
      <add Name="PreprocessArchiveFormatsFilter" Value=".msg|.eml" />
      <!-- Setting this to false is not recommend as it increases -->
      <!-- the chance of archive and folder name collision. -->
      <add Name="PreprocessArchive.IncludeExtensionInFolderName" Value="true" />
      <!-- Setting this to false is not recommend as it increases the chance of archive -->
      <!-- and folder name collision. Can be used in with OutputFolder.PrependUniqueGUID -->
      <!-- OutputFolder.AppendUniqueGUIDToFilename to flatten the structure and create u -->
      <add Name="PreprocessArchive.CreateAllOutputInSubfolder" Value="true" />

      <!-- Preprocess MSG Archive Settings -->
      <add Name="PreprocessArchive.MSG.IncludeInlineAttachments" Value="true" />
      <!-- Pipe (|) separated list of file extensions (e.g *.doc/*.docx) to match on whe -->
      <!-- processing message attachments. Pass empty string for match all. Runs after - -->
      <!-- inline attachment check above, precedes exclusion check below.-->
      <add Name="PreprocessArchive.MSG.AttachmentsIncludeFilter" Value="" />
      <!-- Pipe (|) separated list of file extensions (e.g *.png/*.jpg) to exclude when -->
      <!-- processing message attachments. Pass empty string to exclude none. -->
      <add Name="PreprocessArchive.MSG.AttachmentsExcludeFilter" Value="" />
      <!-- When converting image attachments to images, keep the new image's resolution -->
      <!-- same as source image. Fax mode and other image option actions can override th -->
```

```

<!-- This setting overrides ConverterPlugIn.PNImageConverter.KeepSourceImageResolu
<add Name="PreprocessArchive.MSG.ImageAttachmentsKeepSourceResolution" Value="True" />

<!-- The following settings are only used if OutputFolder.PrependUniqueGUIDToFile
<!-- or OutputFolder.AppendUniqueGUIDToFilename are set to true. -->
<!-- When below is set to true, all files extracted from an MSG will have the same
<add Name="PreprocessArchive.MSG.UseSameGUIDForAllFiles" Value="true" />
<!-- When set to true, the folder used to store the msg and its attachments will -
<!-- be formatted with the pre-post GUID strings as set. -->
<!-- If PreprocessArchive.MSG.UseSameGUIDForAllFiles is also true, the GUID -->
<!-- in the folder name will match the files underneath. -->
<add Name="PreprocessArchive.MSG.UseUniqueGUIDInMSGFolderName" Value="true" />
<!-- When set to true, all files extracted from an MSG will use the ID, including
<!-- the Fully Qualified Domain Name (FQDN) -->
<add Name="PreprocessArchive.MSG.UseMessageIDForPrependGUID" Value="true" />
<add Name="PreprocessArchive.MSGUseMessageIDForAppendGUID" Value="true" />
<!-- Set this to true to only use the first part of the Message ID, -->
<!-- dropping the @FQDN part. -->
<add Name="PreprocessArchive.MSG.UseMessageIDWithoutFQDN" Value="true" />

<!-- Keep image resolution the same as source. Applies to all images-->
<add Name="ConverterPlugIn.PNImageConverter.KeepSourceImageResolution" Value="True" />

<!-- Output file options -->
<add Name="Devmode settings;Resolution" Value="300"/>

<add Name="Save;Output File Format" Value="TIFF Multipaged"/>
<!-- Replace the above with this to create serialized images. -->
<!-- <add Name="Save;Output File Format" Value="TIFF Serialized"/> -->

<add Name="Save;Append" Value="0"/>
<add Name="Save;Color reduction" Value="Optimal"/>
<add Name="Save;Dithering method" Value="Halftone"/>

<!-- This creates file.ext.tif, change to 1 to create file.tif-->
<add Name="Save;Remove filename extension" Value="0" />

<add Name="TIFF File Format;BW compression" Value="Group4"/>
<add Name="TIFF File Format;Color compression" Value="High quality JPEG"/>
<add Name="TIFF File Format;Indexed compression" Value="High quality JPEG"/>
<add Name="TIFF File Format;Greyscale compression" Value="High quality JPEG"/>

</Settings>
</WatchFolder>
</WatchFolders>

```

Creating Done Files to Signal Completion

Starting with Document Conversion Service 3.0.025, the Watch Folder Service now includes the ability to create *done* files. These files can be used by other processes to know when the Watch Folder has completed conversion of any file dropped into the input folder.

There are two types of files created, a *.done* file when conversion is successful, and a *.failed* when a file could not be converted. These files are created in the root of the output folder by default.

A *.done* file contains the path to the original file that was dropped into the Watch Folder's input folder, and then lists all files created. This can be one file, or many if creating serialized output, or when [processing Outlook MSG files and extracting attachments](#).

A *.failed* file contains the path to the original file as dropped into the Watch Folder's input folder, and then the path to the file's location in the Watch Folder's failed folder.

Enabling Done and Failed File Creation

Done and failed file creation is disabled to start. Each one can be enabled or disabled independently.

DoneFile.Success.Create

Set this to *true* to enable Done file creation.

DoneFile.Failed.Create

Set this to *true* to enable Failed file creation.

Customizing the File Name and Location

The initial behavior is to create these files in the root of the Output folder using the same name as the input file, including the extension, and to add the *.done* or *.failed* to the end. This can be customized to use a different folder, use GUIDs instead of the name of the input file, and to use different extensions instead of the default of *.done* and *.failed*.

DoneFile.Success.CustomFolder, DoneFile.Failed.CustomFolder

When left blank, the files are created in the root of the output folder. To create these files in their own folder, set the path here.

DoneFile.Success.CustomExt, DoneFile.Failed.CustomExt

If you need a different extension instead of the defaults of *.done* and *.failed*, set them here. A dot (.) is not needed. Do not use the same extension for both files if you are saving them to the same location.

DoneFile.Success.UseGUIDName, DoneFile.Failed.UseGUIDName

To use unique GUIDs as the base name of the files instead of the input file, set these to true. This would create a file similar to 9BE7305721C14242A2BFF4EA06F3FE92.done.

File Contents

The input file is the the first line of information in both the *.done* file and the *.failed* file. This can be disabled to just list the files created.

DoneFile.Success.IncludeSourceFileAsFirstLine, DoneFile.Failed.IncludeSourceFileAsFirstLine

Set this to *false* if you do not want to have the input file listed as the first item in the done or failed file.

Control Sort Order on File Pickup

Starting with Document Conversion Service 3.0.027, the Watch Folder Service now includes the ability to order the files by name, date created or date modified when picking up files from the Input folder.

There are four sorting options - *None*, *Name*, *DateCreated*, and *DateModified*. Files can be returned in *Ascending* or *Descending* order.

This only controls the order in which the files are picked up from the directory. It does not guarantee the order the files are processed in, only that files sorted to the top of the list are submitted for conversion first. A smaller file further down the list might finish before a larger file that was first in the list.

Configuring the Sort Mode and Order

Sort order defaults to name and ascending when picking up files. Files in the root of the input folder are picked up and sorted first. If sub folders are enabled, they are searched in alphabetical order. Any files in each sub folder are then sorted and returned.

Preprocess.SortFiles

There are four sorting modes that can be used:

- **None** - No ordering is used. Files are returned in the order they were given to us from the underlying file system.
- **Name** - This is the default if the setting is not found or the value is incorrect. Files are sorted based on the full path name of the source file in the input folder.
- **DateCreated** - Files are sorted based on their creation date. For watch folders where files are dropped, a file can be moved or copied into the folder. If the files are moved into the Input folder they will retain their original created date. Copying a file into the Input folder will set the created date to the time of the copy.
- **DateModified** - Files are sorted based on when they were last modified on the computer.

Preprocess.SortFilesOrder

This sets the order of the files. The default is Ascending.

- **Ascending** - sorted the files from low to high: 0-9, A-Z.
- **Descending** - sorts the files from high to low: Z-A, 9-0.

Post-Conversion Processing

Starting with Document Conversion Service 3.0.010, the Watch Folder Service now includes the ability run a separate command in the success and failure cases at the end of conversion.

For version 3.0.010, if the conversion is successful, the command is run *for each file* created by the conversion. This means that if you are creating serialized files, the command will be run for every file created. If the conversion fails, the command for the failed scenario is run on the original source file.

This was updated in version 3.0.015 so that the commands are not run until the successfully created files are copied into the output folder, or for the failure case, when the original source file is copied into the failed folder. Also, the default behavior for the success command has been changed to only run the command a single time, instead of once for each file. A new parameter has been added that allows you to change this back so that the command is run for each file.


Also new with version 3.0.015 is a parameter to enable the creation of a text file containing a list of all of the files created. This file is created in the watched folders's OutputFolder location. It is named using the source file name, a random suffix and ends with the extension *.wfsoutputlist*. You are responsible for deleting this file when you are done with it.

Macros for the source file name, the output list, the number of files created, and other information is available to use in your command line parameters to pass this information into your command. There are different macros for the Success and Failed options; a full list is provided below.

The defaults for running a command are not to wait for the command to complete before continuing. If you do need to wait for the command to be completed, this can be changed, and there is a maximum timeout value that must be specified to allow the process to move ahead if the command does something unexpected.

Shown below are sample post-conversion options to process the created files as a group, or to process each file individually. They use the fictional command line tools *BatchUploadToServer.exe*, *UploadToServer.exe* and *TriggerEmailFail.exe*. These executable names are placeholders that you would replace with your own command line tools that fit your workflow. These command line tools are not part of Document Conversion Service.

See the table below for a full description of each setting.



Code Sample - Sample Post Conversion Commands - Run Once on Success/Fail

```

<WatchFolders>

<!-- This watch folder creates 300 DPI Optimized TIFF Images -->
<WatchFolder Name="ConvertToTIFF Watch Folder">
  <Settings>
    <!-- Folder options -->
    ...

    <!-- Run Command at End On Success -->
    <!-- Run the command once, passing in the list of created files -->
    <!-- and the original source file. -->
    <add Name="RunAtEnd.Success.Enabled" Value="true" />
    <add Name="RunAtEnd.Success.Command" Value="C:\MyTools\BatchUploadToServer.exe" />
    <add Name="RunAtEnd.Success.RunForEachFile" Value="false" />
    <add Name="RunAtEnd.Success.CreateOutputFileList" Value="true" />
    <add Name="RunAtEnd.Success.Parameters"
      Value="&quot;$(OutputFileList)&quot; &quot;$(SourceFileName)&quot;" />
    <add Name="RunAtEnd.Success.StartDirectory" Value="" />
    <!-- ** Internal Use Only ** - One of Normal, Min, Max, Hidden (default) -->
    <add Name="RunAtEnd.Success.WindowState" Value="Hidden" />
    <!-- Wait mode for the command, one of WaitForCompletion, WaitWithExitCode, DoNotWait -->
    <add Name="RunAtEnd.Success.WaitMode" Value="DoNotWait" />

```




Code Sample - Sample Post Conversion Commands - Run Once on Success/Fail

```

<!-- Default is 3 minutes -->
<add Name="RunAtEnd.Success.WaitModeMaxTime" Value="180000" />

<!-- Run Command at End On Failure -->
<!-- If a file fails to convert, send an email with the path to the failed file.-->
<add Name="RunAtEnd.Fail.Enabled" Value="true" />
<add Name="RunAtEnd.Fail.Command" Value="C:\MyTools\TriggerFailEmail.exe" />
<add Name="RunAtEnd.Fail.Parameters"
    Value="&quot;$(FailedFilePath)&quot; &quot;$(SourceFileName)&quot;" />
<add Name="RunAtEnd.Fail.StartDirectory" Value="" />
<!-- ** Internal Use Only ** - One of Normal, Min, Max, Hidden (default) -->
<add Name="RunAtEnd.Fail.WindowState" Value="Hidden" />
<!-- Wait mode for the command, one of WaitForCompletion, WaitWithExitCode, DoNotWait
<add Name="RunAtEnd.Fail.WaitMode" Value="DoNotWait" />
<!-- Default is 3 minutes -->
<add Name="RunAtEnd.Fail.WaitModeMaxTime" Value="180000" />
...

</Settings>
</WatchFolder>
</WatchFolders>

```



Code Sample - Sample Post Conversion Commands - Run for Each File on Success/Once on Fail

```

<WatchFolders>

<!-- This watch folder creates 300 DPI Optimized TIFF Images -->
<WatchFolder Name="ConvertToTIFF Watch Folder">
  <Settings>
    <!-- Folder options -->
    ...

    <!-- Run Command at End On Success -->

    <!-- Run the command for each file created, passing in the created file name -->
    <!-- and the original source file. For serialized output this command is run multiple
    <add Name="RunAtEnd.Success.Enabled" Value="true" />
    <add Name="RunAtEnd.Success.Command" Value="C:\MyTools\UploadToServer.exe" />
    <add Name="RunAtEnd.Success.RunForEachFile" Value="true" />
    <add Name="RunAtEnd.Success.CreateOutputFileList" Value="false" />
    <add Name="RunAtEnd.Success.Parameters"
        Value="&quot;$(OutputFilePath)&quot; &quot;$(OutputFileNumber) $(OutputFileNumberCount)" />
    <add Name="RunAtEnd.Success.StartDirectory" Value="" />
    <!-- ** Internal Use Only ** - One of Normal, Min, Max, Hidden (default) -->
    <add Name="RunAtEnd.Success.WindowState" Value="Hidden" />
    <!-- Wait mode for the command, one of WaitForCompletion, WaitWithExitCode, DoNotWait
    <add Name="RunAtEnd.Success.WaitMode" Value="DoNotWait" />
    <!-- Default is 3 minutes -->
    <add Name="RunAtEnd.Success.WaitModeMaxTime" Value="180000" />

    <!-- Run Command at End On Failure -->
    <!-- If a file fails to convert, send an email with the path to the failed file.-->
    <add Name="RunAtEnd.Fail.Enabled" Value="true" />
    <add Name="RunAtEnd.Fail.Command" Value="C:\MyTools\TriggerFailEmail.exe" />
    <add Name="RunAtEnd.Fail.Parameters"
        Value="&quot;$(FailedFilePath)&quot; &quot;$(SourceFileName)&quot;" />
    <add Name="RunAtEnd.Fail.StartDirectory" Value="" />
    <!-- ** Internal Use Only ** - One of Normal, Min, Max, Hidden (default)-->
    <add Name="RunAtEnd.Fail.WindowState" Value="Hidden" />
    <!-- Wait mode for the command, one of WaitForCompletion, WaitWithExitCode, DoNotWait
    <add Name="RunAtEnd.Fail.WaitMode" Value="DoNotWait" />
    <!-- Default is 3 minutes -->

```



Code Sample - Sample Post Conversion Commands - Run for Each File on Success/Once on Fail

```
<add Name="RunAtEnd.Fail.WaitModeMaxTime" Value="180000" />
...
</Settings>
</WatchFolder>
</WatchFolders>
```

RunAtEnd Success Commands:

Key	Value
RunAtEnd.Success.Enabled	<p>Set this to <i>true</i> to run the specified command when conversion succeeds.</p> <p>Starting with version 3.0.015 the default is to run the command ONCE at the end of conversion, when all the files have been created. This means that even when you are creating serialized output, the command is only called once.</p> <p>To run the command for EACH FILE created, set RunAtEnd.Success.RunForEachFile to true. This matches the original behavior from version 3.0.010 when the run commands were first introduced.</p>
RunAtEnd.Success.RunForEachFile	<p>Introduced in version 3.0.015.</p> <p>The default for this is <i>false</i>, meaning the command will only be run a single time when conversion succeeds and all output files are successfully copied into the Output folder.</p> <p>Change this value to <i>true</i> to return to the original behavior of running the command for each file created from version 3.0.010. This means the command can be run multiple times when creating serialized output.</p>
RunAtEnd.Success.CreateOutputFileList	<p>Introduced in version 3.0.015.</p> <p>Setting this value to true will create a Unicode text file that lists all files created. This file is created in the OutputFolder location and is named based on the source file name, a random suffix and the extension <i>.wfsoutputlist</i>. Use the variable <i>\$(OutputFileList)</i> to pass the file name to your command.</p> <p>You are responsible for deleting this file when you are done with it.</p>
RunAtEnd.Success.Command	<p>The full path to the command to be executed without arguments. Default is an empty string, no command to run.</p>
RunAtEnd.Success.Parameters	<p>The parameters for the command. Use the HTML code to put the command in quotes if there are spaces, and to enclose parameters. The following variables are available to pass arguments to the command.</p> <p>\$(OutputFilePath) - this is the full path to the converted file. Beginning with version 3.0.015, when creating serialized output this is the last file created in the set unless <i>RunAtEnd.Success.RunForEachFile</i> is true, in which case it is the full path to the converted file.</p>

Key	Value
	<p>\$(SourceFileName) - this is the file name of the original file.</p> <p>Introduced in version 3.0.015:</p> <p>\$(OutputFileList) - valid when <code>RunAtEnd.Success.CreateOutputFileList</code> is set to true, this variable is the full path to a Unicode text file containing a list of all the files created, one per line. This list can be a single file, for multipaged conversion, or many files in the case of serialized conversion. See the parameter for more information.</p> <p>\$(OutputFileNumber) - this is the numerical index of the file being processed in the run command. Index starts at 1.</p> <p>\$(OutputFileNumberCount) - this is the number of files created, not including any text extraction files.</p>
RunAtEnd.Success.StartDirectory	The directory in which to run the command. Default is an empty string.
RunAtEnd.Success.WindowState	<p>This option is for internal use only; we recommend leaving this set to the default of <i>Hidden</i>.</p> <p>The view state of the command window when it is run.</p> <p>Normal - display the window in its normal state. Min - display the window minimized to the taskbar Max - display the window maximized. Hidden - do not show the window. (Default)</p>
RunAtEnd.Success.WaitMode	<p>Optionally wait for the command to complete before continuing. The default is to not wait.</p> <p>If WaitForCompletion or WaitWithExitCode is chosen, the RunAtEnd.Success.WaitModeMaxTime value is always used to stop the command if it has not returned after the set amount of time.</p> <p>WaitForCompletion - wait for the command to complete before continuing. WaitWithExitCode - waits for the command to complete and emits the exit code in the log. DoNotWait - does not wait for the command to complete. (Default)</p>
RunAtEnd.Success.WaitModeMaxTime	The maximum amount of time to wait for the command being run to complete. Default is 3 minutes.

RunAtEnd Fail Commands:

Key	Value
RunAtEnd.Fail.Enabled	Set this to <i>true</i> to run the specified command on the original file if the conversion fails. Default is <i>false</i> .
RunAtEnd.Fail.Command	The full path to the command to be executed without arguments. Default is an empty string, no command to run.
RunAtEnd.Fail.Parameters	<p>The parameters for the command. Use the HTML code to put the command in quotes if there are spaces, and to enclose parameters. The following variables are available to pass arguments to the command.</p> <p>\$(FailedFilePath) - this is the path to the original file in its failed location.</p> <p>\$(SourceFileName) - this is the file name of the original file.</p>
RunAtEnd.Fail.StartDirectory	The directory in which to run the command. Default is an empty string.
RunAtEnd.Fail.WindowState	<p>This option is for internal use only; we recommend leaving this set to the default of <i>Hidden</i>.</p> <p>The view state of the command window when it is run.</p> <p>Normal - display the window in its normal state. Min - display the window minimized to the taskbar Max - display the window maximized. Hidden - do not show the window. (Default)</p>
RunAtEnd.Fail.WaitMode	<p>Optionally wait for the command to complete before continuing. The default is to not wait.</p> <p>If WaitForCompletion or WaitWithExitCode is chosen, the RunAtEnd.Fail.WaitModeMaxTime value is always used to stop the command if it has not returned after the set amount of time.</p> <p>WaitForCompletion - wait for the command to complete before continuing. WaitWithExitCode - waits for the command to complete and emits the exit code in the log. DoNotWait - does not wait for the command to complete. (Default)</p>
RunAtEnd.Fail.WaitModeMaxTime	The maximum amount of time to wait for the command being run to complete. Default is 3 minutes.

Unique File Naming and Flat Folder Structures

Starting with Document Conversion Service 3.0.019, the Watch Folder Service now includes the ability to create each output file with a unique name, as well as being able to flatten any folder structure used or dropped into the **InputFolder**.

What is Flattening Folder Structures?

When you *flatten* a folder structure, the idea is to remove all subfolders (and subfolders of folders, etc.) of a particular folder, and place all of the files into a single folder.

The default Watch Folder behavior is to maintain any folder structure found in the **InputFolder** when looking for files to convert, which recreates the folder structure in the **OutputFolder**. This can be disabled to allow the creation of all output files in the **OutputFolder** location, essentially flattening the folder.

The setting **OutputFolder.MaintainInputFolderStructure** controls this behavior. It defaults to *true*, meaning it keeps the folder structure. Set this to *false* to create all output files in a single folder.



Caution

If you are flattening folders you may encounter files with the same names coming from different subfolders. Use the **Unique File Names** settings below to create each filename with a unique name in the OutputFolder.

Unique File Names

Each Watch Folder can be configured to create a unique filename for every file that it finds. This is especially useful if you are flattening folders as explained above.

The unique file names are created using **Globally Unique IDs**, or **GUIDs**. A GUID is a sequence of alphanumeric characters, such as 4c4636f7-e9c5-4c4c-97af-081a328ba7c5, that is for all practical purposes, unique. These unique strings can be placed before, after or both before and after the base filename to create unique output file names.

Use **OutputFolder.PrependUniqueGUIDToFilename** and

OutputFolder.AppendUniqueGUIDToFilename set to true to control if a GUID is added and where it is placed. The setting **OutputFolder.RemoveHyphensFromGUID** allows the hyphens to be removed from the GUID, leaving the GUID as a string of alphanumeric characters (4c4636f7e9c54c4c97af081a328ba7c5).

An default separator string of an underscore character (`_`) is used between the GUID and the original filename. It is defined using the **OutputFolder.UniqueGUIDSeparatorCharacter** setting. This can be changed to a different character, a sequence of characters or left as an empty string to not use the separator character between the GUID and the filename. If any invalid file naming characters such as `<`, `>`, or others, are found in this string, it will default back to a single underscore.

When creating serialized output, where each output file is a single page from the input document, each file in the sequence will have the same GUID. Set **OutputFolder.UseSameGUIDForEachSerializedFile** to false to use a unique GUID for each file in the sequence.



Code Sample - Sample Flattening and Unique Naming Settings

```
<WatchFolders>

  <WatchFolder Name="ConvertToTIFF Watch Folder">
    <Settings>
      ...

      <!-- Flatten folder structure and add a GUID to the beginning of the filename. -->
      <!-- 4c4636f7-e9c5-4c4c-97af-081a328ba7c5_ Filename.tif. -->

      <add Name="OutputFolder.MaintainInputFolderStructure" Value="false" />

      <add Name="OutputFolder.PrependUniqueGUIDToFilename" Value="true" />
      <add Name="OutputFolder.AppendUniqueGUIDToFilename" Value="false" />
      <add Name="OutputFolder.UseSameGUIDForEachSerializedFile" Value="true" />

      <add Name="OutputFolder.RemoveHyphensFromGUID" Value="false" />
      <add Name="OutputFolder.UniqueGUIDSeparatorCharacter" Value="_" />
      ...
    </Settings>
  </WatchFolder>
</WatchFolders>
```

Skipping Files with the Passthrough Converter

The PEERNET Passthrough converter is a by-pass mechanism that allows files to be sent through the Document Conversion Service without actually being converted. This type of behavior is useful when dealing with a group of files where some of the input files sent may already be in the desired output format, but you still need them moved to your final destination for further processing.


The PEERNET Passthrough converter will work with any file type as it uses the file's extension to recognize which file types to skip.

For example, if you have the Watch Folder Service configured to convert any files dropped into a specific folder into TIFF files, you can configure the Watch Folder Service to send any files with the ".tif" or ".tiff" extension to the PEERNET Passthrough converter where they are moved directly to the final destination without being converted.

Using the Passthrough Converter with the Watch Folder Service

The steps below show how to take an existing Watch Folder Service folder definition that creates TIFF images and modify it so that the Passthrough converter is used to skip converting any TIFF images. Any TIFF images are moved to the output folder without being converted. This same technique can be used on any file extension.

1. Open the configuration file in the [DCS Editor](#) by going to **DCS Dashboard - Watch Folder Settings - Edit Configuration**, or by going to Start - All Programs - PEERNET Document Conversion Service 3.0 - Watch Folder - Configure Watch Folder Settings.
2. In the configuration file, look for the desired the `<WatchFolder>` section; there can be more than one. To have only this `<WatchFolder>` section use the Passthrough converter for TIFF images, add the *PEERNET Passthrough* converter to the beginning of the list of converters to use for TIFF images.

 **Code Sample - Skip TIFF images on a single watch folder**

```
<WatchFolders>
  <WatchFolder Name="Folder Watch Create TIFF Images">
    <Settings>
      <!-- Folder options -->
      <add Name="InputFolder" Value="C:\PEERNET\Test\Input" />
      <add Name="SearchFilter" Value="*.*/>
      ...
      <add Name="Devmode settings;Resolution" Value="200"/>
      <add Name="Save;Output File Format" Value="TIFF Multipaged" />
      <add Name="Save;Append" Value="0"/>
      ...
      <!-- Skip tiff images in this folder, move them to output. -->
      <add Name=".tif"
        Value="PEERNET Passthrough;PEERNET Image Converter;Outside-In AX" />
      <add Name=".tiff"
        Value="PEERNET Passthrough;PEERNET Image Converter;Outside-In AX" />
      ...
    </Settings>
  </WatchFolder>
</WatchFolders>
```

3. To have all `<WatchFolders>` use the Passthrough converter for TIFF images, the change needs to be done in the `<Settings>` section at the bottom of the configuration file.



Code Sample - Skip TIFF images on all WatchFolders

```
<WatchFoldersSection>
  <WatchFolders>
    ...
    <!-- This Watch folder watches a folder on your local machine -->
    <WatchFolder Name="Folder Watch Local Drive">
      <Settings>
        <!-- Folder options -->
        <add Name="InputFolder" Value="C:\PEERNET\Test\Input"/>
        ...
      </Settings>
    </WatchFolder>
  </WatchFolders>

  <Settings>
    <!-- File Extension to Converter Mapping -->
    <!-- These can be added to the Settings section for each WatchFolder -->
    <!-- to tailor each WatchFolder to use different converters for its -->
    <!-- documents. The individual settings take precedence over the -->
    <!-- global WatchFolderSection settings section -->
    <add Name=".doc" Value="Microsoft Word;Outside-In AX" />
    <add Name=".docx" Value="Microsoft Word;Outside-In AX" />
    ...
    <!-- Skip tiff images in this folder, move them to output. -->
    <add Name=".tif"
      Value="PEERNET Passthrough;PEERNET Image Converter;Outside-In AX" />
    <add Name=".tiff"
      Value="PEERNET Passthrough;PEERNET Image Converter;Outside-In AX" />

    <add Name=".bmp" Value="PEERNET Image Converter;Outside-In AX" />
    <add Name=".jpg" Value="PEERNET Image Converter;Outside-In AX" />
    <add Name=".jpeg" Value="PEERNET Image Converter;Outside-In AX" />
  </Settings>
</WatchFoldersSection>
```

4. Save the configuration file; the [DCS Editor](#) will validate the file when saving and prompt to resolve any syntax errors.
5. Restart the Watch Folder Service to have your new changes applied.

Large Volume Batch Conversion Using Clustering

Introduced in Document Conversion Service 3.0.010, clustering can be used on a single computer to efficiently handle converting folders containing a very large number of files.

The Watch Folder Service was initially intended for use with *hot folders* or *drop folders* where small numbers of files to be converted are dropped periodically into a folder. When files are detected in the input folder, the Watch Folder Service moves the entire contents of the folder to its staging location for processing. With a folder containing a large volume of files this causes long time delays as the files are copied as well as potential issues such as not having enough disk space to copy the files.

When clustering mode is enabled, a small number of files in the input folder are locked and then copied to the staging folder for processing. The original file stays in the input folder until it is processed, then unlocked and copied to the Completed folder. This reduces disk space usage and delays in copying large numbers of files. It also allows for faster turnaround to move on to the next file.

Enabling Clustering

Clustering can be enabled on any Watch Folder definition using the **ClusteredProcessing.Enabled** setting. This setting is *false* by default.

Keeping the Conversion Service Busy

The goal here is to take advantage of the conversion threads available in your licensed copy of Document Conversion Service and to keep those threads busy converting documents. The number of conversion threads you have available depends on your license model.

To maximize throughput, we need to match the number of files that Watch Folder will process in parallel to the number of conversion threads available in Document Conversion Service, and increase it by a small amount. This will allow us to keep the conversion threads continuously converting as there will always be a document or two queued up waiting as long as there are files in our folder left to be processed.

The number of files picked up is initially set to **NumberOfDocumentsInParallel** from the Watch Folder Service configuration file. You can override this setting using the **ClusteredProcessing.MaxFilesToPickup** setting.



Code Sample

```
<WatchFolders>

<!-- This watch folder creates 300 DPI Optimized TIFF Images -->
<WatchFolder Name="ConvertToTIFF" >
  <Settings>
    <!-- Folder options -->
    <add Name="Enabled" Value="True"/>
    <add Name="InputFolder"
      Value="C:\PEERNET\WatchFolders\ConvertToTIFF\Input"/>
    <add Name="SearchFilter" Value="*.*"/>
    <add Name="IncludeSubFolders" Value="True"/>
    <add Name="DeleteInputSubFolders" Value="True"/>
    <add Name="StagingFolder"
      Value="C:\PEERNET\WatchFolders\ConvertToTIFF\Staging"/>
    <add Name="WorkingFolder"
      Value="C:\PEERNET\WatchFolders\ConvertToTIFF\Working"/>
    <add Name="FailedFolder"
      Value="C:\PEERNET\WatchFolders\ConvertToTIFF\Failed"/>
    <add Name="CompletedFolder"
      Value="C:\PEERNET\WatchFolders\ConvertToTIFF\Completed"/>
    <add Name="OutputFolder"
      Value=":\PEERNET\WatchFolders\ConvertToTIFF\Output"/>
    <add Name="PollingInterval" Value="15000"/>
    <add Name="DCOMComputerName" Value=""/>
    <add Name="TestMode" Value="false" />

    <!-- 0 means no limit. -->
    <add Name="Polling.MaxFilesToProcessAtATime" Value="0" />
    <add Name="Polling.SynchronousFilePickup" Value="false" />
    ....

    <!-- Clustered Processing -->
    <!-- This forces batch mode pickup with optional synchronous wait and -->
    <!-- no date time stamp used in the Failed\Completed folders. Files are -->
    <!-- locked so other servers can process files from the same folder. -->
    <add Name="ClusteredProcessing.Enabled" Value="true" />
    <!-- Override this for clustering to customize the number of files picked up. -->
    <add Name="ClusteredProcessing.MaxFilesToPickup" Value="6" />

    <add Name="Devmode settings;Resolution" Value="300"/>
    <add Name="Save;Output File Format" Value="TIFF Multipaged" />
    <!-- <add Name="Save;Output File Format" Value="TIFF Serialized" /> -->

    <add Name="Save;Append" Value="0"/>
    <add Name="Save;Color reduction" Value="Optimal"/>
    <add Name="Save;Dithering method" Value="Halftone"/>

    <!-- This creates file.ext.tif, change to 1 to create file.tif -->
    <add Name="Save;Remove filename extension" Value="1"/>

    <add Name="TIFF File Format;BW compression" Value="Group4"/>
    <add Name="TIFF File Format;Color compression" Value="LZW RGB0"/>
    <add Name="TIFF File Format;Indexed compression" Value="LZW"/>
    <add Name="TIFF File Format;Greyscale compression" Value="LZW"/>
    <add Name="JPEG File Format;Color compression" Value="Medium Quality"/>
    <add Name="JPEG File Format;Greyscale compression" Value="High Quality"/>

  </Settings>
</WatchFolder>
</WatchFolders>
```

Large Volume Batch Conversion Using Synchronous File Pickup



Deprecated - Use Clustering Instead

This method of converting very large folder of files is deprecated in favor of using clustered processing to automatically pick up the files from the source folder without having to copy large numbers of files. Clustering is more efficient and allows for higher throughput. See [Large Volume Batch Conversion Using Clustering](#).

The watch folder, **LargeBatchTIFF**, included with the Watch Folder Service, is configured to handle folder containing a large number of files. It picks up a maximum number of files each time, converts them and returns to continue processing through the collection files until all files are complete.

The Watch Folder Service basic design was for use with *hot folders* or *drop folders* where files to be converted are dropped periodically into a folder. It was meant to handle small groups of files being dropped occasionally into the input folder. When files are detected in the input folder, the Watch Folder Service will try and copy the entire contents of the folder to its staging location for processing. When dealing with a folder containing a large volume of files this can cause large time delays as the files are copied, and other issues such as not having enough disk space to copy the files.

To allow for processing folders containing a very large number of files, the settings **Polling.MaxFilesToProcessAtATime** and **Polling.SynchronousFilePickup** were added. These settings are used to control how many files are picked up at every polling interval, and if the first batch of files needs to complete before the next group is picked up.

In this scenario, you would also typically set **UseTimeDateSubFoldersInCompletedFolder** and **UseTimeDateSubFoldersInFailedFolder** to false so that the date-timestamp folders for each mini-batch of files are not created under the output and failed folders.

You may also want to add the setting `<add Name = "Save;Remove filename extension" Value = "1"/>` to make sure that the file extension from the original source file is not used to name the output file. This means that the output file from a file named Manual.docx would become Manual.tif. If this settings is not included, or is set to "0", the output file name would be Manual.docx.pdf.

As an extra precaution, if possible, we recommend making a copy of the original source files and processing off of the copied. This ensures you still have your original collection of files if anything unexpected should happen during the conversion process.



Code Sample

```
<WatchFolders>

<!-- This watch folder is set to allow for dropping a large number of files -->
<!-- at once. The files are picked up in small batches of up to 10 files until -->
<!-- all files have been completed. -->
<WatchFolder Name="LargeBatchTIFF Watch Folder" >
  <Settings>
    <!-- Folder options -->
    <add Name="Enabled" Value="True"/>
    <add Name="InputFolder"
      Value="C:\PEERNET\WatchFolders\LargeBatchTIFF\Input\"/>
    <add Name="SearchFilter" Value="*.*/>
    <add Name="IncludeSubFolders" Value="True"/>
    <add Name="DeleteInputSubFolders" Value="True"/>
    <add Name="StagingFolder"
      Value="C:\PEERNET\WatchFolders\LargeBatchTIFF\Staging"/>
    <add Name="WorkingFolder"
      Value="C:\PEERNET\WatchFolders\LargeBatchTIFF\Working"/>
    <add Name="FailedFolder"
      Value="C:\PEERNET\WatchFolders\LargeBatchTIFF\Failed"/>
    <add Name="CompletedFolder"
      Value="C:\PEERNET\WatchFolders\LargeBatchTIFF\Completed"/>
    <add Name="OutputFolder"
      Value=":\PEERNET\WatchFolders\LargeBatchTIFF\Output"/>
    <add Name="PollingInterval" Value="15000"/>
    <add Name="DCOMComputerName" Value=""/>
    <add Name="TestMode" Value="false" />

    <!-- These settings control the how many files in the batch -->
    <!-- are picked up each time, 0 means no limit. -->
    <add Name="Polling.MaxFilesToProcessAtATime" Value="10" />
    <add Name="Polling.SynchronousFilePickup" Value="true" />

    <add Name="UseTimeDateSubFoldersInCompletedFolder" Value="false" />
    <add Name="UseTimeDateSubFoldersInFailedFolder" Value="false" />
    ....
    <add Name="Devmode settings;Resolution" Value="300"/>
    <add Name="Save;Output File Format" Value="TIFF Multipaged" />
    <!-- Replace the above with this to create serialized images. -->
    <!-- <add Name="Save;Output File Format" Value="TIFF Serialized" /> -->

    <add Name="Save;Append" Value="0"/>
    <add Name="Save;Color reduction" Value="Optimal"/>
    <add Name="Save;Dithering method" Value="Halftone"/>

    <!-- This creates file.ext.tif, change to 1 to create file.tif -->
    <add Name="Save;Remove filename extension" Value="1"/>

    <add Name="TIFF File Format;BW compression" Value="Group4"/>
    <add Name="TIFF File Format;Color compression" Value="LZW RGB0"/>
    <add Name="TIFF File Format;Indexed compression" Value="LZW"/>
    <add Name="TIFF File Format;Greyscale compression" Value="LZW"/>
    <add Name="JPEG File Format;Color compression" Value="Medium Quality"/>
    <add Name="JPEG File Format;Greyscale compression" Value="High Quality"/>

  </Settings>
</WatchFolder>

</WatchFolders>
```

Converting With PEER.NET.ConvertUtility

Welcome to the PEER.NET.ConvertUtility help. The table below outlines the different sections of this help manual.

Topic	Description
Getting Started	Step-by step tutorials in this section explain how to call the PEER.NET.ConvertUtility from your own code, and how to use the returned information to find the converted files, information messages or error messages.
Working With PEER.NET.ConvertUtility	This section covers the more advanced topics such as passing custom settings and creating your own custom conversion profiles.
Deploying Applications	This sections lists the required PEER.NET.ConvertUtility files needed when deploying applications.
PEER.NET.ConvertUtility Namespace	This reference section contains detailed descriptions of all classes in the PEER.NET.ConvertUtility library.

Requirements

The supported development environments and platforms are listed below. Take note that these are different from the platforms supported by Document Conversion Service.

Supported Platforms

Both 32-bit and 64-bit Microsoft® Windows operating systems are supported.

- Windows Server 2022
- Windows 11
- Windows Server 2019
- Windows Server 2016
- Windows 10
- Windows Server 2012 R2
- Windows Server 2012

Limited Support for End-of-Life Platforms

These Microsoft® Windows operating systems have reached the end of their support lifecycle. Document Conversion Service will still install and run on these platforms, but new features added to Document Conversion Service may not be supported.

- Windows Server 2008 R2
- Windows Server 2008
- Windows 8.1
- Windows 7

Supported Development Environments

PEERNET.ConvertUtility requires Microsoft® .NET Framework 4.5 or higher to be installed. The following development environments can be used:

- Visual Basic .NET 2010, 2012, 2013, 2015, 2017, 2022
- Visual C# .NET 2010, 2012, 2013, 2015, 2017, 2022
- PowerShell

Getting Started

The tutorials in this section are designed to provide a quick introduction to the PEERNET.ConvertUtility .NET library. If you are new to the Document Conversion Service, the quickest way to learn how to add file conversion into your .NET application is to follow the tutorial in your language of choice.

The section [Working With PEERNET.ConvertUtility](#) provides information on the more advanced features of the PEERNET.ConvertUtility library.

Starting with one of the following tutorials is recommended:

- [C# Tutorial](#)
- [Visual Basic .NET Tutorial](#)
- [Using the Results Object](#)

C# Tutorial

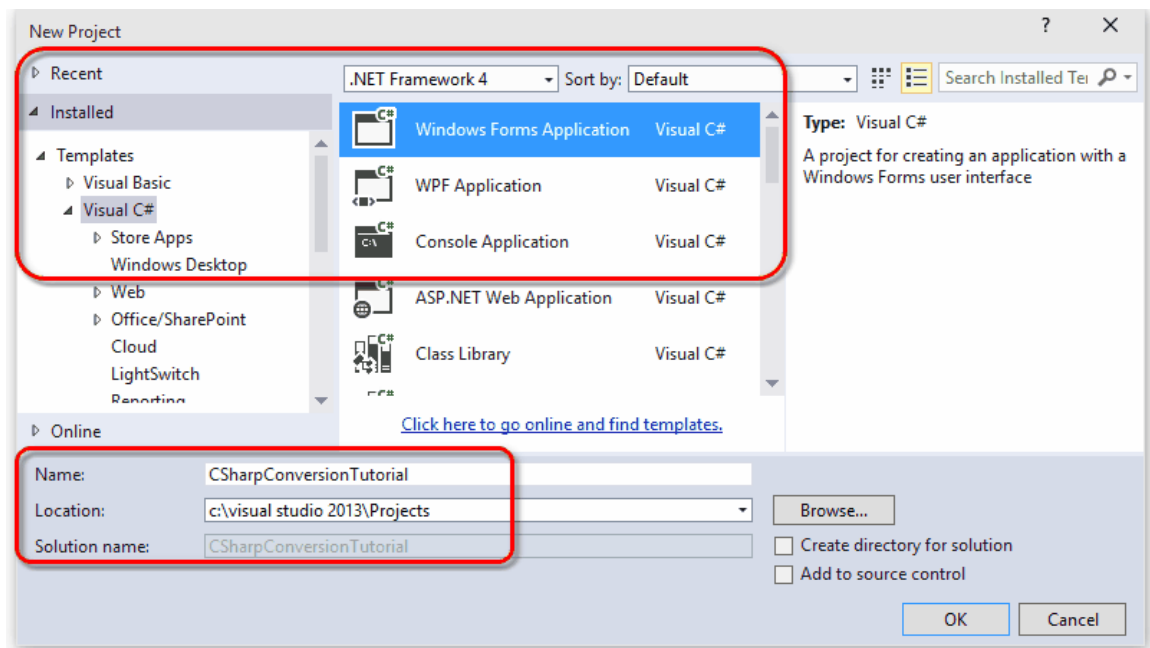
This tutorial will show you how to add file conversion to your C#.NET application using PEER.NET.ConvertUtility. The tutorial creates a simple C# Windows forms application with a single button that converts a file when pressed and displays the results in a list box when finished. It also assumes that Document Conversion Service is installed on your local computer.

- Step 1: [Creating a Simple Application](#)
- Step 2: [Adding the PEER.NET.ConvertUtility Library](#)
- Step 3: [Converting a File](#)
- Step 4: [Displaying the Conversion Results](#)
- Step 5: [Testing the Application](#)

1. Creating a Simple Application

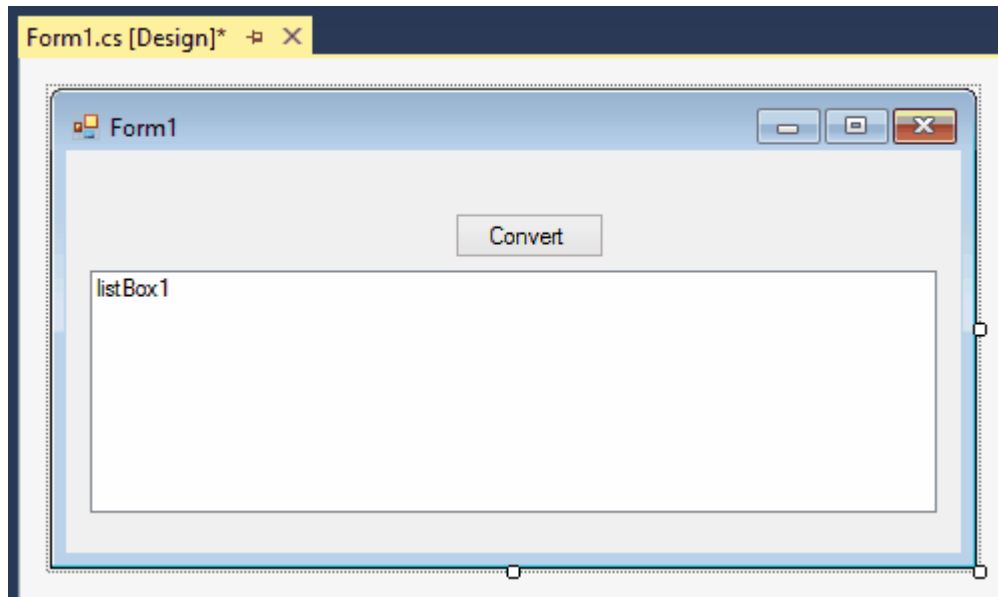
In this first step we will create a simple C# forms application with a single button and a list box.

1. Start Visual Studio .NET and select *New Project* from the start page or File - New - Project... from the menu.
2. Select the Visual C# Windows Forms Application template and target the .NET Framework 4.
3. Enter a name and location for this sample and press OK.



4. Next, add two controls onto the new form.
 - a. From the toolbox, drag a button onto Form1 and change the text of the button to "Convert".
 - b. Go back to the toolbox and drag a listbox onto the form.

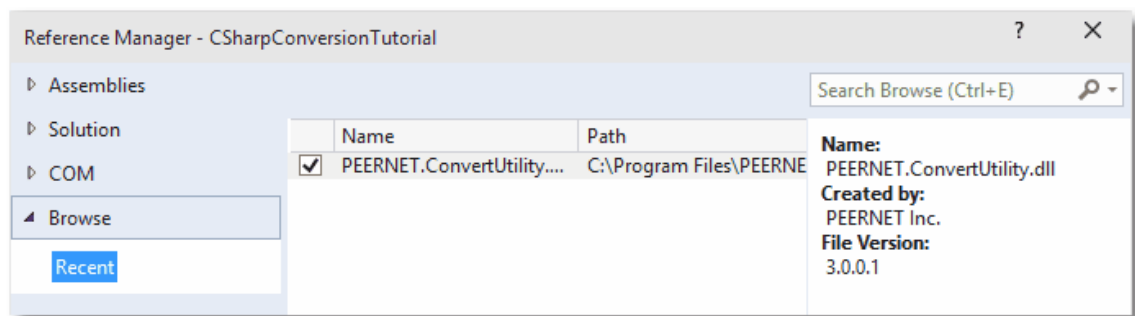
- c. Change the width of both the form and the listbox to be able to display more information in the listbox.



2. Adding the PEERNET.ConvertUtility Library

In this section we will add PEERNET.ConvertUtility support to the project.

1. Right click References in the solution explorer and select *Add Reference*.
2. Click the Browse tab and add a reference to the PEERNET.ConvertUtility.dll into the project. It is located in the \Samples\Redist folder under the Document Conversion Services installation folder.



3. Right click on *Form1* and open the source code view by selecting View Code.
4. Add the following statement to the top of the *Form1.cs* file.

```
using PEERNET.ConvertUtility;
```

3. Converting a File

Now we have all the pieces we need to convert a file and display the results into the listbox.

1. On the design view of *Form1*, double click the button added above to create the Click event and switch to code view.
2. In the *button1_Click* method, add the following code to call [ConvertFile](#) to convert a file to a 200dpi TIFF image.
 - a. Replace the underlined arguments with your own *input filename*, *output folder*, and *converted filename*.
 - b. The *output folder* must exist before calling *ConvertFile*.
 - c. The call to *ConvertFile* is a blocking call and will not return until the conversion is complete. When it returns we then want to display the results of the conversion in the listbox.
 - d. A *try-catch-finally* block is in place so that, success or failure, the call to *DisplayResultsItems* is always executed and the result of the conversion will always be displayed in the listbox.

```
private void button1_Click(object sender, EventArgs e)
{
    PNConversionItem resultItem = null;
    String strOutputFolder = @"C:\Test\Output";

    try {
        button1.Enabled = false;
        this.listBox1.Items.Clear();
        this.listBox1.Items.Add("Converting...");

        // Directory must exist
        if ( !Directory.Exists(strOutputFolder) )
        {
            Directory.CreateDirectory(strOutputFolder);
        }
        // This is the single call needed to convert a file
        resultItem = PNConverter.ConvertFile(
            @"C:\Test\File.pdf",
            strOutputFolder, // output folder
            @"ConvertedFromPDF", // converted file name
            true, // overwrite existing
            false, // do not remove file ext
            false, // do not create log
            "TIFF 200dpi OptimizedColor", // profile
            String.Empty,
            String.Empty,
            null, // no custom user settings
            String.Empty, // not using DCOM
            String.Empty, // use default working folder
            String.Empty // no custom log folder
        );
    }
    catch (Exception ex) {
        this.listBox1.Items.Add(String.Format("An error occurred during conversion. {0}", ex.ToString()));
    }
    finally {
        button1.Enabled = true;
        DisplayResultsItems(resultItem);
    }
}
```

4. Displaying the Conversion Results

All of the conversion methods in *PEERNET.ConvertUtility* return a results item, or in the case of converting a list or a folder of files, a list of results items.

This item contains information about the original conversion request and the results of the conversion. The results of the conversion can be a list of created files or a collection of error messages detailing why the file was not converted.

1. Add the following method into *Form1.cs*. This method will display the name of the file we tried to convert, and then will list the new file that was created. If the conversion failed, the error messages are displayed instead.

```
private void DisplayResultsItems(PNConversionItem result)
{
    if (result != null) {
        // With single file conversion this will be a single item
        // The PNConversionResult object in each item contains the error and file list.
        // Failed items will have an error list > 0 and no output files.

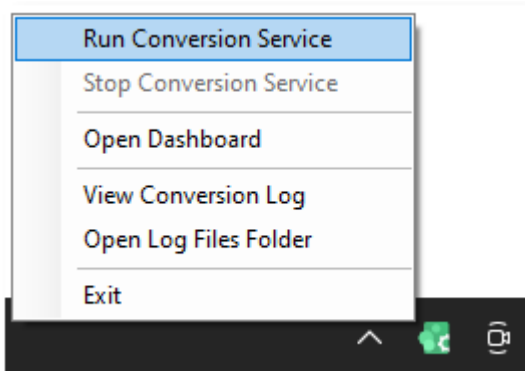
        listBox1.Items.Add("Conversion Item: " + result.SourceFilePath);
        listBox1.Items.Add("=====");

        if (result.HasErrors()) {
            if (result.ConversionResult.Errors.Count > 0) {
                listBox1.Items.Add("Errors occurred during conversion: ");
                foreach (PNConversionResultError itemError in
                    result.ConversionResult.Errors) {
                    listBox1.Items.Add(itemError.Value);
                }
            }
        }
        else {
            if (result.ConversionResult.OutputFiles != null) {
                if (result.ConversionResult.OutputFiles.Count > 0) {
                    listBox1.Items.Add("The following files were created: ");
                    foreach (PNConversionResultOutputFile itemOutputFile in
                        result.ConversionResult.OutputFiles) {
                        listBox1.Items.Add(itemOutputFile.OutputFilePath);
                    }
                }
                else {
                    listBox1.Items.Add("No files were created.");
                }
            }
        }
    }
    // results not null
    else {
        listBox1.Items.Add("Conversion module did not run.");
    }
}
```

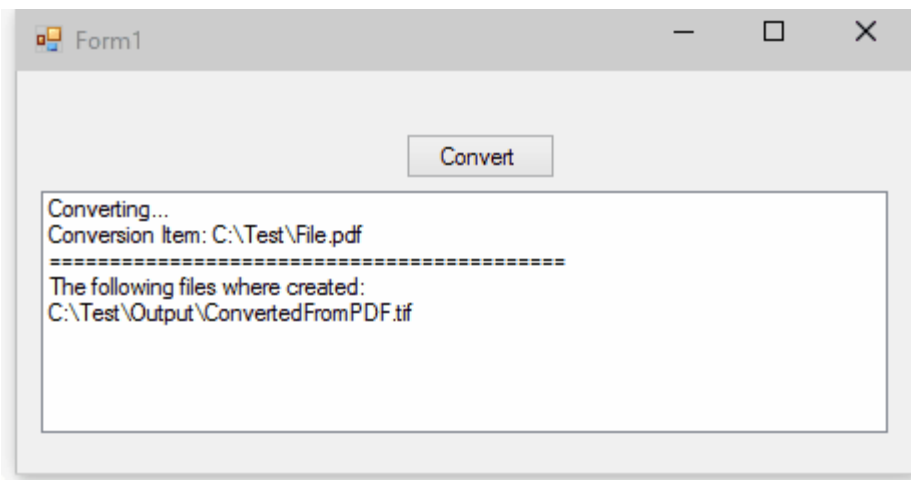
5. Testing the Application

To test the application, Document Conversion Service has to be running as PEERNET.ConvertUtility communicates with Document Conversion Service to perform the file conversion.

1. From the system tray icon menu select Run Conversion Service to start the service. If this menu item is disabled the service is already running.



2. When the service has finished initializing, build and run your C# project.
3. Click on the button to convert your file. The listbox will display the message "Converting..." and then the results of the conversion.



Visual Basic .NET Tutorial

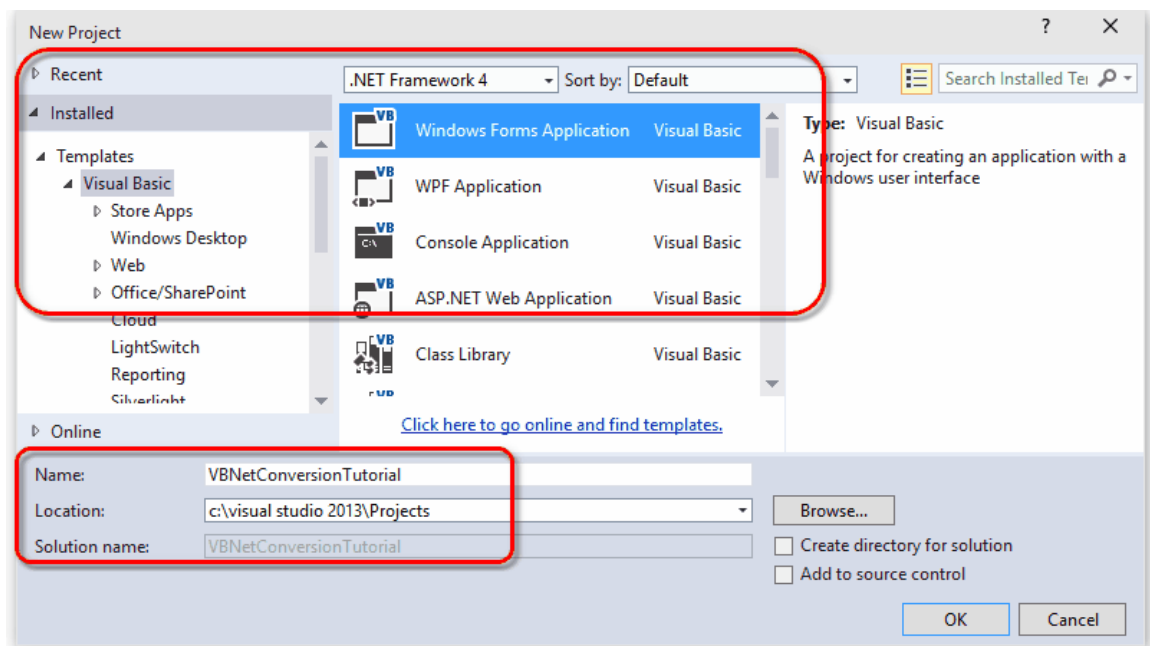
This tutorial will show you how to add file conversion to your Visual Basic .NET application using PEERNET.ConvertUtility. The tutorial creates a simple Visual Basic Windows forms application with a single button that converts a file when pressed and displays the results in a list box when finished. It also assumes that Document Conversion Service is installed on your local computer.

- Step 1: [Creating a Simple Application](#)
- Step 2: [Adding the PEERNET.ConvertUtility Library](#)
- Step 3: [Converting a File](#)
- Step 4: [Displaying the Conversion Results](#)
- Step 5: [Testing the Application](#)

1. Creating a Simple Application

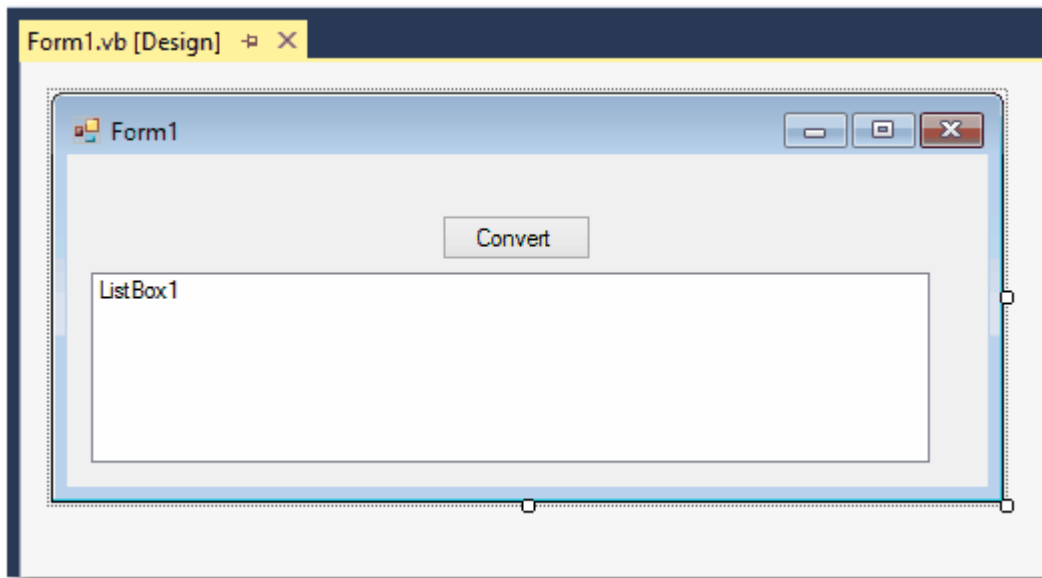
In this first step we will create a simple Visual Basic .NET forms application with a single button and a list box.

1. Start Visual Studio .NET and select *New Project* from the start page or File - New - Project... from the menu.
2. Select the Visual Basic Windows Forms Application template and target the *.NET Framework 4*.
3. Enter a name and location for this sample and press OK.



4. Next, add two controls onto the new form.
 - a. From the toolbox, drag a button onto Form1 and change the text of the button to "Convert".
 - b. Go back to the toolbox and drag a listbox onto the form.

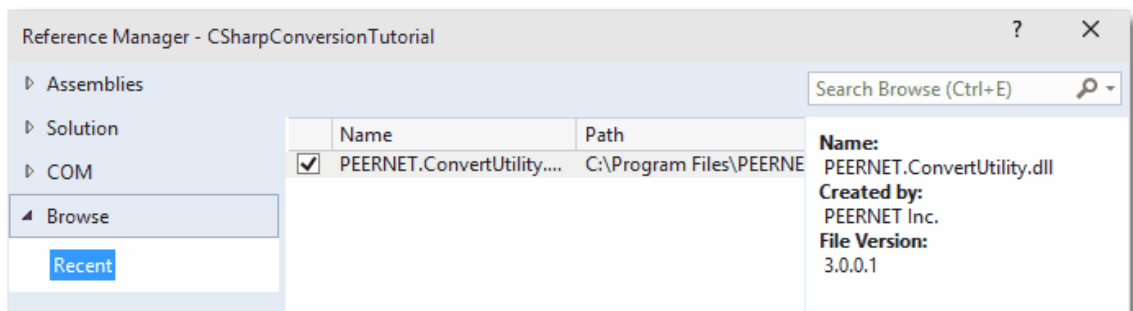
- c. Change the width of both the form and the listbox to be able to display more information in the listbox.



2. Adding the PEERNET.ConvertUtility Library

In this section we will add PEERNET.ConvertUtility support to the project.

1. In the solution explorer, right click the project and select *Add Reference*.
2. Click the Browse tab and add a reference to the PEERNET.ConvertUtility.dll into the project. It is located in the \Samples\Redist folder under the Document Conversion Services installation folder.



3. Right click on *Form1* and open the source code view by selecting *View Code*.
4. Add the following statement to the top of the *Form1.vb* file.

```
Imports PEERNET.ConvertUtility
```

3. Converting a File

Now we have all the pieces we need to convert a file and display the results into the listbox.

1. On the design view of *Form1*, double click the button added above to create the Click event and switch to code view.
2. In the *Button1_Click* method, add the following code to call [ConvertFile](#) to convert a file to a 200dpi TIFF image.
 - a. Replace the underlined arguments with your own *input filename*, *output folder*, and *converted filename*.
 - b. The *output folder* must exist before calling *ConvertFile*.
 - c. The call to *ConvertFile* is a blocking call and will not return until the conversion is complete. When it returns we then want to display the results of the conversion in the listbox.
 - d. A *Try-Catch-Finally* block is in place so that, success or failure, the call to *DisplayResultsItems* is always executed and the result of the conversion will always be displayed in the listbox.

```
Private Sub Button1_Click(sender As System.Object, _
                        e As System.EventArgs) _
    Handles Button1.Click

    Dim resultItem As PNConversionItem
    Dim strOutputFolder As String

    resultItem = Nothing
    strOutputFolder = "C:\Test\Output"

    Try
        Button1.Enabled = False
        ListBox1.Items.Clear()
        ListBox1.Items.Add("Converting...")

        ' Directory must exist
        If Not Directory.Exists(strOutputFolder) Then
            Directory.CreateDirectory(strOutputFolder)
        End If

        ' This is the single call needed to convert a file
        resultItem = PNConverter.ConvertFile(
            "C:\Test\File.pdf", _
            "C:\Test\Output", _
            "ConvertedFromPDF", _
            True, _
            False, _
            False, _
            "TIFF 200dpi OptimizedColor", _
            String.Empty, _
            String.Empty, _
            Nothing, _
            String.Empty, _
            String.Empty, _
            String.Empty)

    Catch ex As Exception
        ListBox1.Items.Add(String.Format("An error occurred during conversion. {0}",
            ex.ToString()))
    Finally
        Button1.Enabled = True
        DisplayResultsItems(resultItem)
    End Try
End Sub
```

4. Displaying the Conversion Results

All of the conversion methods in *PEERNET.ConvertUtility* return a results item, or in the case of converting a list or a folder of files, a list of results items.

This item contains information about the original conversion request and the results of the conversion. The results of the conversion can be a list of created files or a collection of error messages detailing why the file was not converted.

1. Add the following method into *Form1.vb*. This method will display the name of the file we tried to convert, and then will list the new file that was created. If the conversion failed, the error messages are displayed instead.

```
Private Sub DisplayResultsItems(result As PNConversionItem)

    If Not result Is Nothing Then

        ' With single file conversion this will be a single item,
        ' The PNConversionResult object in each item contains the error and file list
        ' Failed items will have an error list > 0 and no output files.

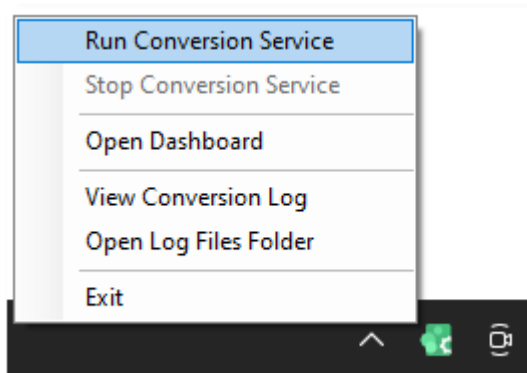
        ListBox1.Items.Add("Conversion Item: " & result.SourceFilePath)
        ListBox1.Items.Add("=====")

        If (result.HasErrors()) Then
            If (Not result.ConversionResult.Errors Is Nothing And _
                result.ConversionResult.Errors.Count > 0) Then
                ListBox1.Items.Add("Errors occurred during conversion: ")
                For Each itemError As PNConversionResultError In _
                    result.ConversionResult.Errors
                    ListBox1.Items.Add(itemError.Value)
                Next
            End If
        Else
            If (Not IsNothing(result.ConversionResult.OutputFiles)) Then
                If (result.ConversionResult.OutputFiles.Count > 0) Then
                    ListBox1.Items.Add("The following files were created: ")
                    For Each itemOutputFile As PNConversionResultOutputFile In _
                        result.ConversionResult.OutputFiles
                        ListBox1.Items.Add(itemOutputFile.OutputFilePath)
                    Next
                Else
                    ListBox1.Items.Add("No files were created.")
                End If
            End If
        End If
        Else
            ListBox1.Items.Add("Conversion module did not run.")
        End If
    End Sub
```

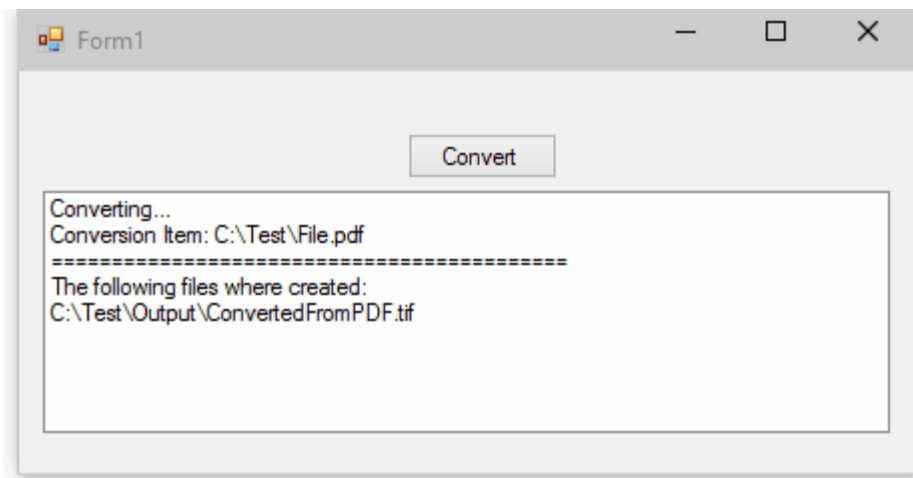
5. Testing the Application

To test the application, Document Conversion Service has to be running as PEERNET.ConvertUtility communicates with Document Conversion Service to perform the file conversion.

1. From the system tray icon menu select Run Conversion Service to start the service. If this menu item is disabled the service is already running.



2. When the service has finished initializing, build and run your VB.NET project.
3. Click on the button to convert your file. The listbox will display the message "Converting..." and then the results of the conversion.



Using the Results Object

The convert method [ConvertFile](#) returns a [PNConversionItem](#) object that describes the original conversion request and contains an internal object, [PNConversionResult](#) which contains the results of the conversion request.

The methods [ConvertFileList](#) and [ConvertFolder](#), which convert groups of files, return a list of [PNConversionItem](#) objects, one for every file found to convert.

It is this object that is queried to find out the following:

- The status of the conversion - success or failure?
- If the conversion failed, what errors occurred?
- What files were created?

Getting the Conversion Status and Error Information

To find out the status of a conversion you can call either one of two methods: [HasErrors](#) or [GetConversionStatus](#).

[HasErrors](#) returns **True** if there were any errors during conversion for this item. All error message are available through the [Errors](#) collection in the [ConversionResult](#) property.

The method [GetConversionStatus](#) returns a [PNConvertResultStatus](#) conversion status for this item.

```
private void ReportStatusAndErrors(PNConversionItem result)
{
    if (result != null) {
        String status = result.GetConversionStatus();
        listBox1.Items.Add("Status:" + status);

        if (result.HasErrors()) {
            if (result.ConversionResult.Errors.Count > 0) {
                listBox1.Items.Add("Errors occured during conversion: ");
                foreach (PNConversionResultError itemError in
                    result.ConversionResult.Errors) {
                    listBox1.Items.Add(itemError.Value);
                }
            }
        }
        // results not null
    } else {
        listBox1.Items.Add("Conversion module did not run.");
    }
}
```

What Files Were Created?

The [PNConversionResult](#) object in each [PNConversionItem](#) object contains a collection listing all of the files created by this conversion request. The [ConvertFile](#) method returns a single [PNConversionItem](#) while the methods [ConvertFileList](#) and [ConvertFolder](#) will return a list of [PNConversionItem](#) objects, one for each file in the list or folder that was converted.

```
private void ListCreatedFiles(PNConversionItem result)
{
    if (result != null) {
        listBox1.Items.Add("Conversion Item: " + result.SourceFilePath);
        listBox1.Items.Add("=====");

        if (!result.HasErrors()) {
            if (result.ConversionResult.OutputFiles != null) {
                if (result.ConversionResult.OutputFiles.Count > 0) {
                    listBox1.Items.Add("The following files where created: ");
                    foreach (PNConversionResultOutputFile itemOutputFile in
                        result.ConversionResult.OutputFiles) {
                        listBox1.Items.Add(itemOutputFile.OutputFilePath);
                    }
                }
                else {
                    listBox1.Items.Add("No files were created.");
                }
            }
        }
    }
}
```

Working With PEER.NET.ConvertUtility

This section contains information on more advanced topics such as using custom conversion settings, creating your own profiles and controlling parallel document conversion.

If you are new to the PEER.NET.ConvertUtility you should start with the tutorials in the [Getting Started](#) section.

- [Passing Custom Conversion Settings](#)
- [Converting a Folder of Files](#)
- [Converting a List of Files](#)
- [Combining a List of Files into a Single File](#)
- [Combining a Folder of Files into a Single File](#)
- [Combining Select Pages Of Each File](#)
- [Converting Files with Long Path Names](#)
- [Controlling Parallel Document Conversion](#)
- [Waiting for Document Conversion Service to be Ready to Convert](#)
- [Creating and Customizing Profiles](#)

Passing Custom Conversion Settings

When calling the Convert methods from the PEERNET.ConvertUtility library, you have many ways of configuring the output files created.

The easiest method is to pass the name of one of the *profiles* included with Document Conversion Service into the convert method. You can also pass the full path to a profile on disk that you have created yourself.

A profile is just an XML file that contains the list of *conversion settings* settings as name/value pairs. Using a profile has the advantage of allowing you to change the conversion settings in the profile on disk without having to recompile your code.

The other two methods are as follows:

- [Pass Additional Settings with User Settings](#)
- [Passing a Custom List of Conversion Settings](#)

Pass Additional Settings with User Settings

If you are using a profile to specify your conversion settings you can dynamically modify the profile settings without changing the profile on disk by passing a list of *user settings* to the convert methods. Settings provided in this list will override matching settings in the profile while new settings will be added to the list of conversion settings for this call only.

This C# code sample demonstrates creating a list of two user settings and passing it to the ConvertFile method. The first additional setting will cause serialized TIFF images to be created instead of multipaged. The second setting is used to control how the Word document is printed; in this case we want to see any document and markup that occurred on the document.

```
IDictionary<String, String> UserSettings = new Dictionary<String, String>();
PNConversionItem resultItem = null;

UserSettings.Add("Save;Output File Format", "TIFF Serialized");
UserSettings.Add("Microsoft.Word.Document.PrintOut.Item", "DocumentAndMarkup");

// conversion results returned in result item, use it to find files created or errors
resultItem = PNConverter.ConvertFile(@"C:\Input\Memo.doc",
    @"C:\Output\",
    "ConvertedMemo",
    true, // overwrite existing
    false, // remove file extension
    false, // create log file
    "TIFF 200dpi Monochrome",
    settings,
    String.Empty,
    String.Empty,
    UserSettings, // custom settings
    String.Empty, // remote computer
    String.Empty, // use default working folder
    String.Empty);
```

Passing a Custom List of Conversion Settings

You can also configure the output files by passing in a list of *conversion settings* that you define before you call the convert method. Conversion settings are name/value pairs of settings that define the output files. The same name/value pairs that you would use when creating a profile on disk are used when building these lists of settings.

These settings are commonly used to control what type of output file to create - TIFF, PDF, JPEG, or others, the resolution of the created images, or single-paged or multi-paged output.

Additionally, you can control some aspects of the conversion modules such as having Word documents print with tracking and revisions visible, or having all PowerPoint slides printed with the notes.

The C# code sample below demonstrates calling [ConvertFile](#) with a custom list of conversion settings to create a PDF file. The input file `C:\Input\Memo.doc` will be converted to a PDF file and saved as `C:\Output\ConvertedMemo.pdf`.

```
IDictionary<String, String> settings = new Dictionary<String, String>();
PNConversionItem resultItem = null;

settings.Add("Devmode settings;Resolution", "300");
settings.Add("Save;Output File Format", "Adobe PDF Multipaged");
settings.Add("Save;Append", "0");
settings.Add("Save;Color reduction", "Optimal");
settings.Add("Save;Dithering method", "Halftone");
settings.Add("PDF File Format;PDF Standard", "None");
settings.Add("PDF File Format;Content encoding", "LZW");
settings.Add("PDF File Format;Use ASCII", "0");
settings.Add("PDF File Format;Color compression", "LZW");
settings.Add("PDF File Format;Greyscale compression", "LZW");
settings.Add("PDF File Format;Indexed compression", "LZW");
settings.Add("PDF File Format;BW compression", "Group4");
settings.Add("PDF Security;Use Security", "1");
settings.Add("PDF Security;Encrypt Level", "1");
settings.Add("PDF Security;Can Copy", "1");
settings.Add("PDF Security;Can Print", "1");
settings.Add("PDF Security;Can Change Doc", "0");
settings.Add("PDF Security;Can ChangeOther", "0");
settings.Add("PDF Security;User Pswd On", "0");
settings.Add("PDF Security;Owner Pswd On", "0");

// conversion results returned in result item, use it to find files created or errors
resultItem = PNConverter.ConvertFile(@"C:\Input\Memo.doc",
                                     @"C:\Output\",
                                     "ConvertedMemo",
                                     true, // overwrite existing
                                     false, // remove file extension
                                     false, // create log file
                                     settings,
                                     String.Empty,
                                     String.Empty,
                                     null, // no extra settings
                                     String.Empty, // remote computer
                                     String.Empty, // use default working folder
                                     String.Empty );
```

Converting a Folder of Files

The [ConvertFolder](#) method is used to convert files in the given folder and optionally any subfolders as well. As with the [ConvertFile](#) method, the *conversion settings* are passed as a profile, or through a custom list of settings. When converting a folder of files, all files are converted with the same conversion settings.

If an output location is provided the directory structure, including subfolders, will be maintained in the new location. This directory must exist before the call to `ConvertFolder` is made.

If an output location is not provided a new folder named *.converted* is created in the same location as the source file and all output files are saved there.

Filtering Files in the Folder

You can use the *Filter* and the *ExcludeFilter* arguments to specify what files in the folder you want to convert. The *Filter* is always applied to the directory contents first, then the *ExcludeFilter* is applied to that list of files to remove the unwanted files.

Hidden and system files are ignored, and the search pattern filters files based on a regular expression match of the long name of a file. The *Filter* defaults to all files in the folder (*.**) if *String.Empty* or *null* are passed. *ExcludeFilter* is ignored when *String.Empty* or *null* is passed.

Multiple filters can be combined using the pipe (|) character, such as **.doc|*.pdf* to process only Word and PDF files. The table below lists some examples of filtering directory contents.

Filter	Exclude Filter	Action
*.pdf	String.Empty	Process only PDF documents.
.	*.tif *.jpg	Process all documents except TIFF and JPEG images.
*.doc *.docx *.txt	Draft_*	Process all Word and Text documents except those starting with <i>Draft_</i> .

Sorting the Files for Pickup

Starting with Document Conversion Service 3.0.029, this method now includes the ability to order the files by name, date created or date modified when picking up files from the Input folder.

Configuring the Sort Mode and Order

Sort order defaults to name and ascending when picking up files. Files in the root of the input folder are picked up and sorted first. If sub folders are enabled, they are searched in alphabetical order. Any files in each sub folder are then sorted and returned. Uses the [PNFileSortMode](#) enumeration.

Note: Any sorting options applied only control the order in which the files are picked up from the directory. Sorting does not guarantee the order the files are processed in, only that files sorted to the top of the list are submitted for conversion first. A smaller file further down the list might finish before a larger file that was first in the list.

There are four sorting modes that can be used:

- **None** - No ordering is used. Files are returned in the order they were given to us from the underlying file system.

- **Name** - This is the default if the setting is not found or the value is incorrect. Files are sorted based on the full path name of the source file in the input folder.
- **DateCreated** - Files are sorted based on their creation date. For watch folders where files are dropped, a file can be moved or copied into the folder. If the files are moved into the Input folder they will retain their original created date. Copying a file into the Input folder will set the created date to the time of the copy.
- **DateModified** - Files are sorted based on when they were last modified on the computer.

The order of the files is either Ascending or Descending. Uses the [PNFileSortOrder](#) enumeration.

- **Ascending** - sorted the files from low to high: 0-9, A-Z.
- **Descending** - sorts the files from high to low: Z-A, 9-0.

Converting a Folder of Files

The code sample below will convert all files in the folder `C:\Test\InputFiles\` except TIFF, JPEG and BMP images. Any subfolders will also be searched for files to convert. A sort order of *DateCreated* is set, meaning files created first will be submitted for processing first. This does not control the order in which the files are completed.

```

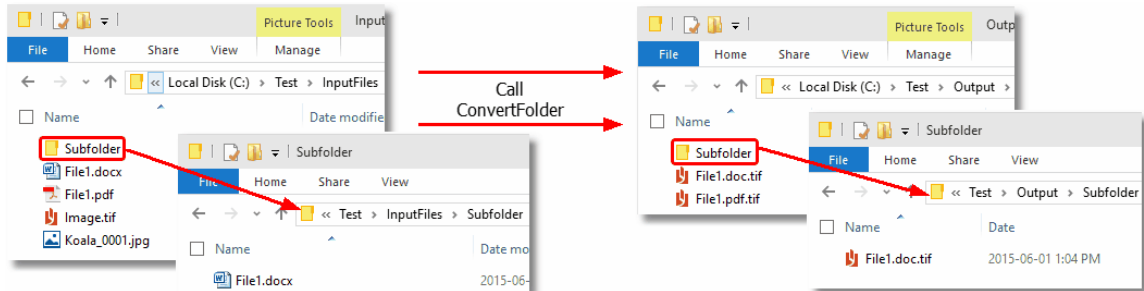
IList<PNConversionItem> results = new List<PNConversionItem>();
String strOutputFolder = @"C:\Test\Output";

// Directory must exist
if ( !Directory.Exists(strOutputFolder) )
{
    Directory.CreateDirectory(strOutputFolder);
}

// Convert the folder
results = PNConverter.ConvertFolder(@"C:\Test\InputFiles",
    true, // include subfolders
    ".*", // filter
    ".*.tif|*.jpg|*.bmp", // exclude filter
    strOutputFolder, // output folder
    true, // overwrite existing
    false, // remove file ext
    false, // create log
    "TIFF 200dpi OptimizedColor", // settings
    String.Empty, // extensions profile
    String.Empty, // MIME profile
    null, // User settings
    String.Empty, // not using remote conversion (DCOM)
    String.Empty, // use default working folder
    String.Empty,
    PNFileSortMode.DateCreated, // sort by created date
    PNFileSortOrder.Ascending); // A-Z, 0-9

```

The created files, in this case TIFF images, will be placed in the output folder `C:\Test\Output` and the directory structure maintained. The name of the original file, including the extension, is used as the base name for the new file. Files matching the *ExcludeFilter* were not converted.



Reading the Results Collection

When converting a folder of files a list of [PNConversionItem](#) objects will be returned, one for every file found to convert. Each [PNConversionItem](#) object contains information about the original conversion request and an internal [PNConversionResult](#) object that lists the results of the conversion. The results of the conversion can be a list of created files or a collection of error messages detailing why the file was not converted.

This code sample traverses the returns results from the above folder conversion and lists the files created.

```
if (results != null)
{
    int idx = 0;
    foreach (PNConversionItem item in results)
    {
        idx++;
        Console.WriteLine("*****");
        Console.WriteLine(String.Format("Item {0}", idx));
        Console.WriteLine("*****");

        if (item != null)
        {
            Console.WriteLine("Item: " + item.SourceFilePath);
            Console.WriteLine("      " + item.OutputDirectory);
            Console.WriteLine("      " + item.OutputBaseName);

            if (item.HasErrors() == false)
            {
                foreach (PNConversionResultOutputFile outputfile in
                    item.ConversionResult.OutputFiles)
                {
                    Console.WriteLine("      Converted to: " + outputfile.OutputFilePath);
                }
            }
            else
            {
                foreach (PNConversionResultError errorItem in
                    item.ConversionResult.Errors)
                {
                    Console.WriteLine("      Error: " + errorItem.Value);
                }
            }
        }
    }
}
```

```
}  
}  
}
```

The console output from the above code is shown below.

```
*****  
* Item 1 *  
*****  
Item: C:\Test\InputFiles\Subfolder\File1.doc  
      C:\Test\Output\Subfolder  
      File1.doc  
      Converted to: C:\Test\Output\Subfolder\File1.doc.tif  
*****  
* Item 2 *  
*****  
Item: C:\Test\InputFiles\File1.doc  
      C:\Test\Output  
      File1.doc  
      Converted to: C:\Test\Output\File1.doc.tif  
*****  
* Item 3 *  
*****  
Item: C:\Test\InputFiles\File1.pdf  
      C:\Test\Output  
      File1.pdf  
      Converted to: C:\Test\Output\File1.pdf.tif  
Press any key to exit...
```

Converting a List of Files

The [ConvertFileList](#) method allows you to convert a list of files from various locations in a single call. Each file in the list can optionally have different conversion settings and different output locations. This is different from [ConvertFolder](#) where all files are converted with the same settings.

Building the List of Files

The list of files is passed to the [ConvertFileList](#) method as a collection of [PNConvertFileInfo](#) objects.

The [PNConvertFileInfo](#) class requires the path to the source file and two optional arguments - the path to the output folder and a list of additional conversion settings to use when the source file is converted.

If an output folder is specified in the, this folder must exist before calling the conversion method. If this path is left empty the output folder specified in the [ConvertFileList](#) call is used. If that folder path is also empty, the file will be created in the same location as the source file.

The settings provided in the [PNConvertFileInfo](#) class are used *in addition* to the conversion settings passed to the [ConvertFileList](#) method either as a profile or through the settings list parameter.

A sample list of files to convert is created below. This list will output each file into its own folder. The second file in the list also includes additional settings to use when converting the file.

```
IList<PNConvertFileInfo> fileList = new List<PNConvertFileInfo>();
IList<PNSetting> filesettings = new List<PNSetting>();

// This file uses only the conversion settings from the profile
fileList.Add(new PNConvertFileInfo(@"C:\Test\InputFiles\File1.pdf",
                                   @"C:\Test\Output\ConvertedPDF\",
                                   null));

// This file also changes the conversion settings to 300 dpi and
// causes the Word converter to print markup.
filesettings.Add( new PNSetting("Devmode settings;Resolution", "300")); // driver setting
filesettings.Add( new PNSetting("Microsoft.Word.Document.PrintOut.Item", // converter settir
                                "DocumentAndMarkup") );
fileList.Add(new PNConvertFileInfo(@"C:\Test\InputFiles\File1.doc",
                                   @"C:\Test\Output\ConvertedDocs\",
                                   filesettings));
```

Converting the List of Files

The code sample below uses the file list created above. It first checks the that all of the output paths exist and creates them if necessary, then calls [ConvertFileList](#) to convert all files in the list and place the created files in the output folder specified.

The first file in the list will use only the conversion settings from the profile *TIFF 200dpi OptimizedColor*.

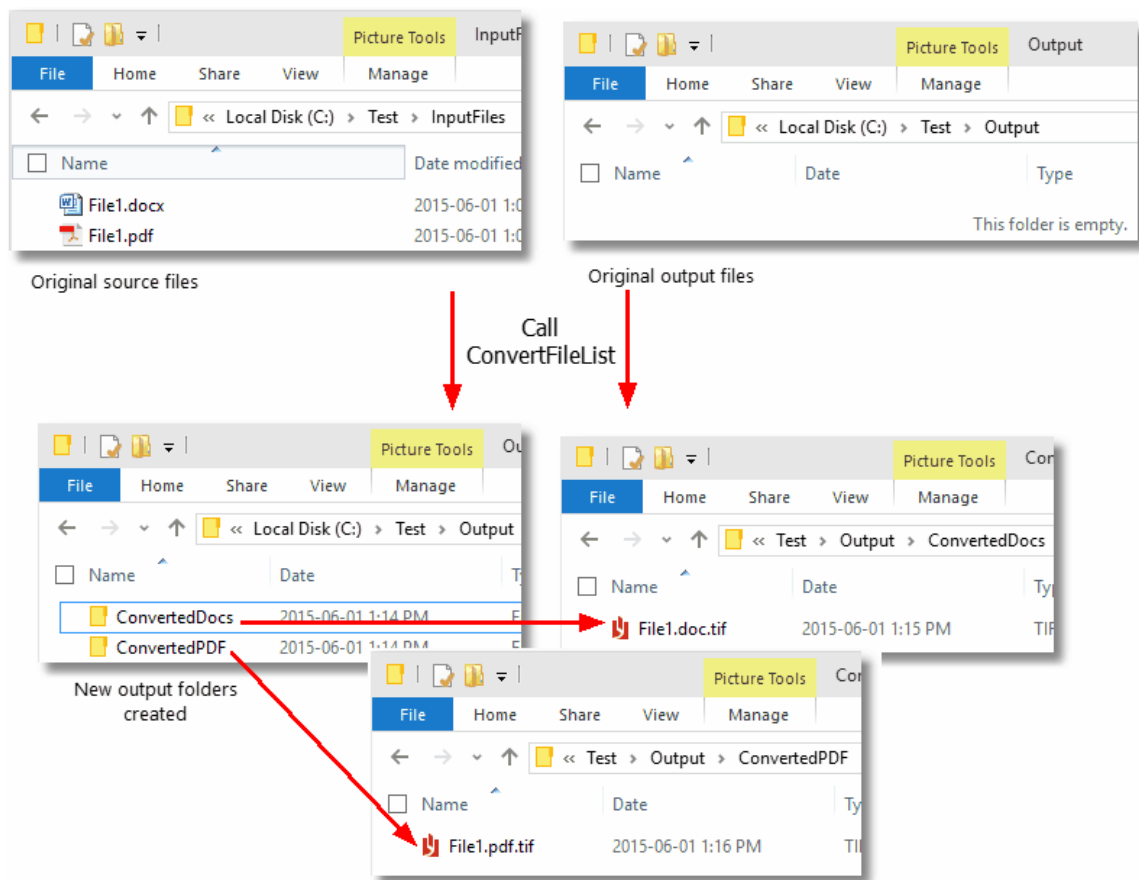
The second file has additional settings: the first to change the image resolution from 200 dpi to 300 dpi, and the second setting to tell the Word converter to have any markup (comments, review) visible in the converted file.

```
IList<PNConversionItem> results = new List<PNConversionItem>();
```

```
// Test output, directories need to be created
foreach (PNConvertFileInfo info in fileList)
{
    if (!String.IsNullOrEmpty(info.OutputPath) &&
        !Directory.Exists(info.OutputPath))
    {
        Directory.CreateDirectory(info.OutputPath);
    }
}

results = PNConverter.ConvertFileList(fileList,
                                     String.Empty, // no output folder
                                     String.Empty, // no converted file name
                                     false, // do not overwrite
                                     false, // remove file ext
                                     false, // create log
                                     "TIFF 200dpi OptimizedColor", // settings
                                     String.Empty, // extension profile
                                     String.Empty, // MIME profile
                                     null, // User settings
                                     String.Empty, // not using remote conversion (DCOM)
                                     String.Empty, // use default working folder
                                     String.Empty);
```

The created files, in this case TIFF images, will be placed in the specified output folder for each file. The name of the original file, including the extension, is used as the base name for the new file.



Reading the Results Collection

When converting a list of files, the results are returned as a collection of [PNConversionItem](#) object, one for every file sent to be converted. Each [PNConversionItem](#) object contains information about the original conversion request and an internal [PNConversionResult](#) object that lists the results of the conversion. The results of the conversion can be a list of created files or a collection of error messages detailing why the file was not converted.

This code sample traverses the returned results from the above conversion and lists the files created.

```
if (results != null)
{
    int idx = 0;
    foreach (PNConversionItem item in results)
    {
        idx++;
        Console.WriteLine("*****");
        Console.WriteLine(String.Format("* Item {0}                                *", idx));
        Console.WriteLine("*****");
    }
}
```

```

if (item != null)
{
    Console.WriteLine("Item: " + item.SourceFilePath);
    Console.WriteLine("      " + item.OutputDirectory);
    Console.WriteLine("      " + item.OutputBaseName);

    if (item.HasErrors() == false)
    {
        foreach (PNConversionResultOutputFile outputfile in
            item.ConversionResult.OutputFiles)
        {
            Console.WriteLine("      Converted to: " + outputfile.OutputFilePath);
        }
    }
    else
    {
        foreach (PNConversionResultError errorItem in
            item.ConversionResult.Errors)
        {
            Console.WriteLine("      Error: " + errorItem.Value);
        }
    }
}
}
}

```

The console output from the above code is shown below.

```

*****
* Item 1 *
*****
Item: C:\Test\InputFiles\File1.doc
      C:\Test\Output\ConvertedDocs\
      File1.doc
      Converted to: C:\Test\Output\ConvertedDocs\File1.doc.tif
*****
* Item 2 *
*****
Item: C:\Test\InputFiles\File1.pdf
      C:\Test\Output\ConvertedPDF\
      File1.pdf
      Converted to: C:\Test\Output\ConvertedPDF\File1.pdf.tif
Press any key to exit...

```

Combining a List of Files

The [CombineFiles](#) method allows you to combine (append) a list of files from various locations into a single output file, or a serialized sequence of single page output files in a single call. To combine files with different setting per file, see [Combining Select Pages Of Each File](#).

Building the List of Files

The list of files is passed to the [CombineFiles](#) method is a simple `ICollection` of file paths. The path to each file must be a fully qualified path name, relative paths are not accepted.

The files are converted in the order in which they are added to the list. A sample list of files to convert is created below.

```
ICollection<String> fileList = new List<String>();  
  
filelist.Add(@"C:\Test\PDF\InputFile1.pdf");  
filelist.Add(@"C:\Test\DOC\InputFile2.doc");  
filelist.Add(@"C:\Test\XLS\InputFile3.xls");
```

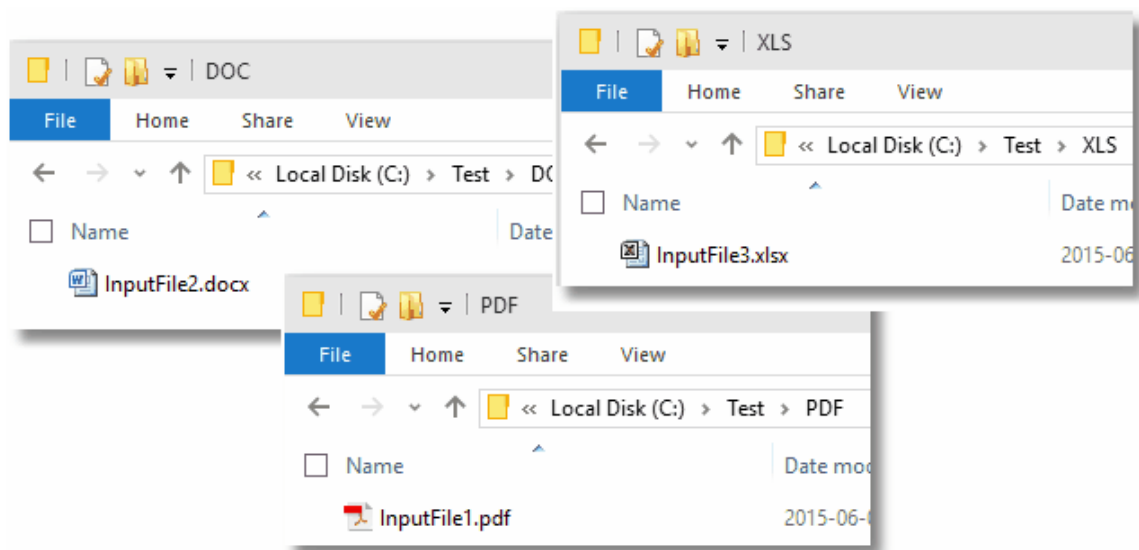
Combining the List of Files

The code sample below uses the file list created above to append all three files into a single multipaged TIFF image. When combining files, the output directory and final output file name must be provided and the directory must exist before the call is made. The code calls [CombineFiles](#) to combine all files in the list and place the final output file in the output folder specified.

The combined file will be created using the conversion settings from the profile *TIFF 200dpi OptimizedColor*. You can change this to use any profile you require.

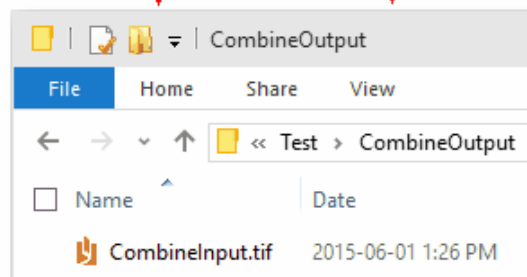
```
PNCombineItem resultsItem = null;  
String outputDir = @"C:\Test\CombineOutput";  
String outputName = "CombinedInput";  
  
resultsItem =  
    PNConverter.CombineFiles(fileList, // files collection  
                             outputDir, // output folder  
                             baseName, // name of combined file  
                             false, // overwrite  
                             false, // create results log  
                             "TIFF 200dpi OptimizedColor", // profile  
                             String.Empty, // File-ext  
                             String.Empty, // MIME  
                             null, // user settings  
                             String.Empty, // not using remote conversion (DCOM)  
                             String.Empty, // use default working folder  
                             String.Empty // Log path  
    );
```

The created file, in this case a multipaged TIFF image, will be placed in the specified output folder C:\Test\CombineOutput and named CombinedInput.tif



Original source files in separate directories

Call
CombineFiles



A single output file containing all three input files has been created.

Reading the Results

When combining a list of files, a [PNCombineItem](#) object is returned. This object contains information about the original combine request, the input files used, a list of the output files created and a collection of [PNConversionResult](#) objects that lists the results of the conversion for each input file. The results of the conversion can be a list of created files or a collection of error messages detailing why the files were not combined.

This code sample traverses the returns results from the above combine and lists the input files used and the files created.


```
if (resultsItem != null)
{
    Console.WriteLine("*****");
    Console.WriteLine("* Combined ITEM *");
    Console.WriteLine("*****");
    Console.WriteLine("BaseName: " + resultsItem.OutputBaseName);
    Console.WriteLine("Directory: " + resultsItem.OutputDirectory);
    Console.WriteLine("Input Files:");
}
```

```
foreach (String inputFile in resultsItem.InputFiles)
{
    Console.WriteLine("    " + inputFile);
}

Console.WriteLine("Combined Output:");
if (resultsItem.CombinedOutputFileList.Count == 0)
{
    Console.WriteLine("    None");
}
foreach (String combinedFile in resultsItem.CombinedOutputFileList)
{
    Console.WriteLine("    " + combinedFile);
}

if (resultsItem.HasErrors() == true)
{
    foreach (PNConversionResultError errorItem in resultsItem.Errors)
    {
        Console.WriteLine("    Error: " + errorItem.Value);
    }
}
}
```

The console output from the above code is shown below.



```
*****
* Combined ITEM *
*****
BaseName: CombinedInput
Directory: C:\Test\CombineOutput\
Input Files:
    C:\Test\PDF\InputFile1.pdf
    C:\Test\DOC\InputFile2.doc
    C:\Test\XLS\InputFile3.xls
Combined Output:
    C:\Test\CombineOutput\CombinedInput.tif
```

Serialized Results

The sample code above used the profile *TIFF 200dpi OptimizedColor* which created a single, multipaged output file. You can also combine multiple files into a serialized sequence of files. For instance, JPEG images are a single page image format and using the profile *JPEG 300dpi Color* will create a serialized sequence of files, one JPEG image for each page of each file.

```
*****
* Combined ITEM                               *
*****
BaseName: CombinedInput
Directory: C:\Test\CombineOutput\
Input Files:
  C:\Test\PDF\InputFile1.pdf
  C:\Test\DOC\InputFile2.doc
  C:\Test\XLS\InputFile3.xls
Combined Output:
  C:\Test\CombineOutput\CombinedInput_001.jpg
  C:\Test\CombineOutput\CombinedInput_002.jpg
  C:\Test\CombineOutput\CombinedInput_003.jpg
  C:\Test\CombineOutput\CombinedInput_004.jpg
  C:\Test\CombineOutput\CombinedInput_005.jpg
  C:\Test\CombineOutput\CombinedInput_006.jpg
  C:\Test\CombineOutput\CombinedInput_007.jpg
  C:\Test\CombineOutput\CombinedInput_008.jpg
  C:\Test\CombineOutput\CombinedInput_009.jpg
  C:\Test\CombineOutput\CombinedInput_010.jpg
```

Combining a Folder of Files

The [CombineFolder](#) method allows you to combine (append) the files in the given folder and optionally any subfolders as well, into a single output file, or a serialized sequence of single page output files.

The conversion settings are passed as a profile, or through a custom list of settings. When converting a folder of files, all files are converted with the same conversion settings. To combine files with different setting per file, see [Combining Select Pages Of Each File](#).

Filtering Files in the Folder

You can use the *Filter* and the *ExcludeFilter* arguments to specify which files in the folder you want to convert. The *Filter* is always applied to the directory contents first, then the *ExcludeFilter* is applied to that list of files to remove the unwanted files.

Hidden and system files are ignored, and the search pattern filters files based on a regular expression match of the long name of a file. The *Filter* defaults to all files in the folder (*.*) if *String.Empty* or *null* are passed. *ExcludeFilter* is ignored when *String.Empty* or *null* is passed.

Multiple filters can be combined using the pipe (|) character, such as *.doc|*.pdf to process only Word and PDF files. The table below lists some examples of filtering directory contents.

Filter	Exclude Filter	Action
*.pdf	String.Empty	Process only PDF documents.
.	*.tif *.jpg	Process all documents except TIFF and JPEG images.
*.doc *.docx *.txt	Draft_*	Process all Word and Text documents except those starting with <i>Draft_</i> .

Sorting the Files for Pickup

Starting with Document Conversion Service 3.0.029, this method now includes the ability to order the files by name, date created or date modified when picking up files from the Input folder.

Configuring the Sort Mode and Order

Sort order defaults to name and ascending when picking up files. Files in the root of the input folder are picked up and sorted first. If sub folders are enabled, they are searched in alphabetical order. Any files in each sub folder are then sorted and returned. Uses the [PNFileSortMode](#) enumeration.

There are four sorting modes that can be used:

- **None** - No ordering is used. Files are returned in the order they were given to us from the underlying file system.
- **Name** - This is the default if the setting is not found or the value is incorrect. Files are sorted based on the full path name of the source file in the input folder.
- **DateCreated** - Files are sorted based on their creation date. For watch folders where files are dropped, a file can be moved or copied into the folder. If the files are moved into the Input folder they will retain their original created date. Copying a file into the Input folder will set the created date to the time of the copy.
- **DateModified** - Files are sorted based on when they were last modified on the computer.

The order of the files is either Ascending or Descending. Uses the [PNFileSortOrder](#) enumeration.

- **Ascending** - sorted the files from low to high: 0-9, A-Z.
- **Descending** - sorts the files from high to low: Z-A, 9-0.

Combining a Folder of Files

The code sample below will convert all files except TIFF, JPEG and BMP images from the folder C:\Test\Input\l. Any subfolders are also be searched for files to convert.

The combined file will be created using the conversion settings from the profile *PDF 200dpi OptimizedColor*. You can change this to use any profile you require.

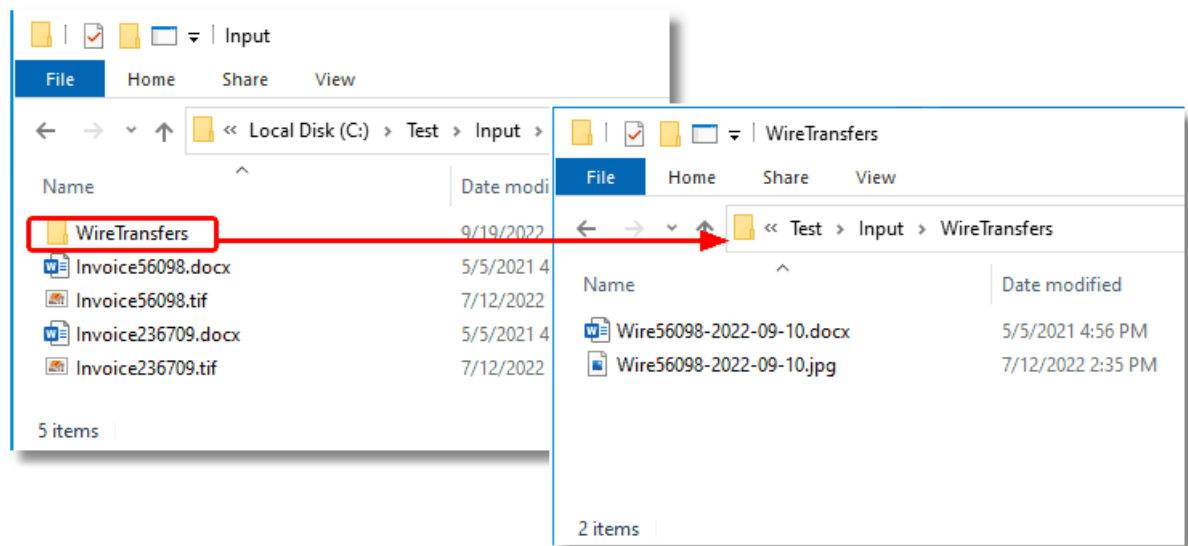
A sort order of *DateCreated* is set, meaning files created first will be submitted for processing first. This will determine the order of the files and pages in the combined file at the end.

```
PNCombineItem resultsItem = null;
String inputDir = @"C:\Test\Input";
String outputDir = @"C:\Test\CombinedOutput";
String outputName = "CombinedInput";

// Directory must exist
if ( !Directory.Exists(outputDir) )
{
    Directory.CreateDirectory(outputDir);
}

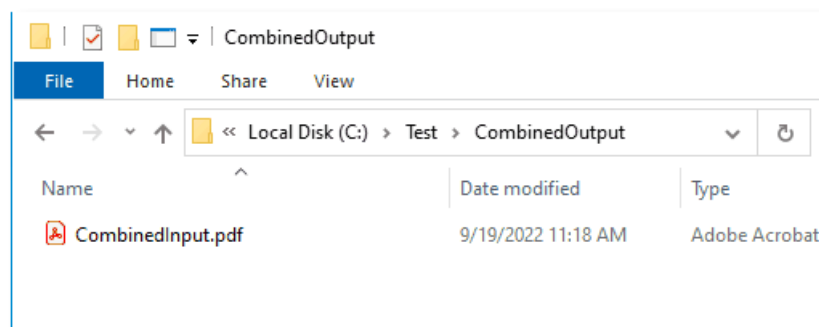
resultsItem =
    PNConverter.CombineFolder(inputDir, // folder of files
                             true, // include subfolders
                             "*.*", // filter
                             "*.tif|*.jpg|*.bmp", // exclude filter
                             outputDir, // output folder
                             baseName, // name of combined file
                             false, // overwrite
                             false, // create results log
                             "PDF 200dpi OptimizedColor", // profile
                             String.Empty, // File-ext
                             String.Empty, // MIME
                             null, // user settings
                             String.Empty, // not using remote conversion (DCOM)
                             String.Empty, // use default working folder
                             String.Empty, // Log path
                             PNFileSortMode.DateCreated, // sort by created date
                             PNFileSortOrder.Ascending); // A-Z, 0-9
    );
```

The created file, in this case a multipaged PDF document, will be placed in the specified output folder C:\Test\CombinedOutput and named CombinedInput.pdf. Files matching the *ExcludeFilter* (*.tif, *.jpg) were not converted.



Original source files in folder and subfolder

Call
CombineFolder



A single output file containing all three input files has been created.

Reading the Results

When combining a folder of files, a [PNCombineItem](#) object is returned. This object contains information about the original combine request, the files found to be converted, list of the output files created and a collection of [PNConversionResult](#) objects that lists the results of the conversion for each input file. The results of the conversion can be a list of created files or a collection of error messages detailing why the files were not combined.

This code sample traverses the returns results from the above combine and lists the input files used and the files created.

```
if (resultsItem != null)
{
    int idx = 0;
```

```

Console.WriteLine("*****");
Console.WriteLine("* Combined ITEM *");
Console.WriteLine("*****");
Console.WriteLine("BaseName: " + resultsItem.OutputBaseName);
Console.WriteLine("Directory: " + resultsItem.OutputDirectory);
Console.WriteLine("Input Files:");

foreach (String inputFile in resultsItem.InputFiles)
{
    Console.WriteLine("    " + inputFile);
}

Console.WriteLine("Combined Output:");
if (resultsItem.CombinedOutputFileList.Count == 0)
{
    Console.WriteLine("    None");
}
foreach (String combinedFile in resultsItem.CombinedOutputFileList)
{
    Console.WriteLine("    " + combinedFile);
}

if (resultsItem.HasErrors() == true)
{
    foreach (PNConversionResultError errorItem in resultsItem.Errors)
    {
        Console.WriteLine("    Error: " + errorItem.Value);
    }
}
else
{
    foreach (PNConversionItem item in results)
    {
        idx++;
        Console.WriteLine("*****");
        Console.WriteLine(String.Format("* Item {0} ", idx));
        Console.WriteLine("*****");

        if (item != null)
        {
            Console.WriteLine("Item: " + item.SourceFilePath);
            Console.WriteLine("OutputDir: " + item.OutputDirectory);
            Console.WriteLine("BaseName: " + item.OutputBaseName);

            if (item.HasErrors() == false)
            {
                foreach (PNConversionResultOutputFile outputfile in
                    item.ConversionResult.OutputFiles)
                {
                    Console.WriteLine("    Converted to: " + outputfile.OutputFilePath);
                }
            }
            else
            {
                foreach (PNConversionResultError errorItem in
                    item.ConversionResult.Errors)
                {
                    Console.WriteLine("    Error: " + errorItem.Value);
                }
            }
        }
    }
}
}

```

The console output from the above code is shown below.

```
*****
* Combined ITEM                                     *
*****
BaseName: CombinedInput
Directory: C:\Test\CombinedOutput\
Input Files:
    C:\Test\Input\Invoice236709.docx
    C:\Test\Input\Invoice56098.docx
    C:\Test\Input\WireTransfers\Wire56098-2022-09-10.docx
Combined Output:
    C:\Test\CombinedOutput\CombinedInput.pdf
*****
* Item 1                                           *
*****
Item:      C:\Test\Input\Invoice236709.docx
OutputDir  C:\Test\CombinedOutput
BaseName   CombinedInput
           Converted to: C:\Test\CombinedOutput\CombinedInput.pdf
*****
* Item 2                                           *
*****
Item:      C:\Test\Input\Invoice56098.docx
OutputDir  C:\Test\CombinedOutput
BaseName   CombinedInput
           Converted to: C:\Test\CombinedOutput\CombinedInput.pdf
*****
* Item 3                                           *
*****
Item:      C:\Test\Input\WireTransfers\Wire56098-2022-09-10.docx
OutputDir  C:\Test\CombinedOutput
BaseName   CombinedInput
           Converted to: C:\Test\CombinedOutput\CombinedInput.pdf
Press 'Q' to exit or any key to run again..
```


Combining Select Pages Of Each File

The [CombineFiles](#) method also allows you to combine (append) a list of files, each with their own settings, into a single output file, or a serialized sequence of single page output files in a single call. A common use of this is to print only select pages, or all pages, from each file to build the resulting file.

Building the List of Files

To allow each file to have their own settings, the list of files passed to the [CombineFiles](#) method needs to be an `ICollection` of [PNConvertFileInfo](#) objects. The `PNConvertFileInfo` object contains a list of settings that can be applied instead or in addition to the profile settings used when combining.

Only the following converter settings are valid as settings when combining files:

- [General Converter Options](#)
- [Endorsement Options](#)
- [Word Converter Options](#)
- [Excel Converter Options](#)
- [PowerPoint Converter Options](#)
- [Adobe Reader Options](#)
- [Internet Explorer Options](#)
- [Ghostscript Converter Options](#)
- [Image Converter Options](#)
- [OutsideIn AX Options](#)

The path to each file must be a fully qualified path name, relative paths are not accepted. When combining files, the `OutputFolder` property on the `PNConvertFileInfo` object is ignored.

The files are converted in the order in which they are added to the list. A sample list of files to combine is created below; the resulting file will contain all of the pages of the first file, and only the first three pages of the second file.

```
ICollection<PNConvertFileInfo> fileInfoList = new List<PNConvertFileInfo>();
ICollection<PNSetting> fileSettings = new List<PNSetting>();

// This file will print all pages and uses only the conversion
// settings from the profile - we aren't passing any additional settings.
fileInfoList.Add(new PNConvertFileInfo(@"C:\Test\InputFiles\File1.pdf",
                                     String.Empty,
                                     null));

// This file only prints the first 3 pages, but also shows all markup
// in the Word document.
fileSettings.Add( new PNSetting("PageRange", "1-3"));
fileSettings.Add( new PNSetting("Microsoft.Word.Document.PrintOut.Item", // converter settir
                               "DocumentAndMarkup") );
fileInfoList.Add(new PNConvertFileInfo(@"C:\Test\InputFiles\File1.doc",
                                     String.Empty,
                                     fileSettings));
```

Combining the List of Files

The code sample below uses the [PNConvertFileInfo](#) list created above to append both files into a single multipaged TIFF image containing all the pages of the PDF and the first 3 pages of the Word document with markup displayed.

When combining files, the output directory and final output file name must be provided and the directory must exist before the call is made. The code calls [CombineFiles](#) to combine all files in the list and place the final output file in the output folder specified.

The combined file will be created using the conversion settings from the profile *TIFF 200dpi OptimizedColor*, plus any optional settings supplied for each file.

```
PNCombineItem resultsItem = null;
String outputDir = @"C:\Test\CombineOutput";
String outputName = "CombinedInput";

resultsItem =
    PNConverter.CombineFiles(fileInfoList, // PNConvertFileInfo collection
        outputDir, // output folder
        baseName, // name of combined file
        false, // overwrite
        false, // create results log
        "TIFF 200dpi OptimizedColor", // profile
        String.Empty, // File-ext
        String.Empty, // MIME
        null, // user settings
        String.Empty, // not using remote conversion (DCOM)
        String.Empty, // use default working folder
        String.Empty // Log path
    );
```

Converting Files with Long Path Names

Historically, Windows (and before that, DOS) had a maximum path length (MAXPATH) of 260 characters. While this has changed over the years to allow file paths of up to 32,000 characters, many of the underlying components of Windows, including parts of Microsoft.NET, are still bound by the MAXPATH limitation.

Most of the time you never have to think about long path support but it does occasionally occur. A common situation would be having to convert all the files in a directory structure on network attached storage (NAS) created in UNIX or another file system where long paths are supported.



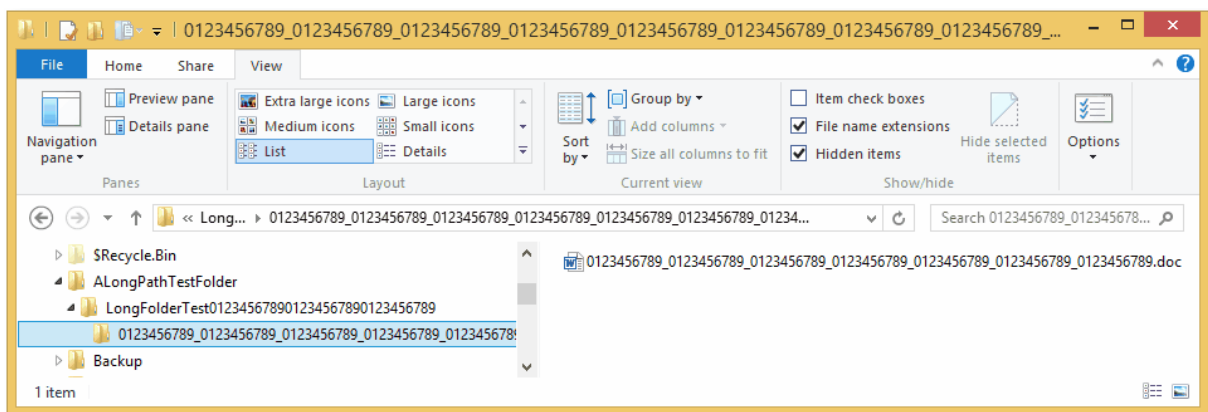
MAXPATH Limitation in Microsoft .NET

Several of the Microsoft.NET System.IO components, namely System.IO.File, System.IO.Directory and System.IO.Path, are all limited by the length of MAXPATH when dealing with files, directories and paths. If you need long path support you will need to P/Invoke the WIN32 File API calls, or use a third-party library that provides long path name support.

All of the conversion methods in PEERNET.ConvertUtility will handle long path names for the input file or folder, output locations, output file name and for the location of the XML results file and logging files.

The one caveat when dealing with long paths is that once the files and directory structures to be converted are copied to the internal staging and working folders in the *ConversionWorkingFolder* to be processed, those paths need to be less than 255 characters. This staging and working folder limitation is a requirement of the underlying programs, such as Adobe Reader and Microsoft Office, that Document Conversion Service uses to perform conversions. If the file path sent to Document Conversion Service to be converted is longer than MAXPATH that file will gracefully fail to convert.

Keep in mind that even if the input folder path itself is not greater than MAXPATH, the underlying subfolders and file names can create a path that is once they have been moved to the staging and working folders. You can see by this sample directory shown below that the path *C:\ALongPathTestFolder* we are using as the input folder path will generate file paths longer than MAXPATH.



Setting the ConversionWorkingFolder to Convert Files with Long Path Names

The code sample below shows the settings and options that can be used to control the location of the *ConversionWorkingFolder*, or the TEMP folder, where the internal staging and working folders are created. Configuring this to a short path off the root of a drive can allow, in most case, for short enough paths internally to convert the files and folders stored in longer paths elsewhere.

Inside the staging and working folders, PEERNET.ConvertUtility uses a date-time stamped subfolder to control the conversion. By default an easy to read folder name similar to *Thursday_March_31_2016_10_16_32_AM* is used. To shorten this further, you can set the **UseCompressedDateTimeFormat** option to true to use the condensed date-time stamp. The condensed version is strictly numerical and similar to 20160331131645, which is much shorter.

```
IList<PNConversionItem> results = new List<PNConversionItem>();
String strOutputFolder = @"C:\LongPathsTest\Output";
String strCustomTempFolder = @"C:\PN";

// This sample path is 263 chars
String strLogFile = @"C:\LongPathsTest\01234567890123456789012345678901234567890" +
    "123456789012345678901234567890123456789012345678901234567890" +
    "123456789012345678901234567890123456789012345678901234567890" +
    "12345678901234567890123456789\Output\SIL\" +
    "ConvertFolderWithAVeryLongName1234567890.sil";

// Directories must exist
if ( !Directory.Exists(strOutputFolder) )
{
    Directory.CreateDirectory(strOutputFolder);
}

if ( !Directory.Exists(strCustomTempFolder) )
{
    Directory.CreateDirectory(strCustomTempFolder);
}

IDictionary<String, String> UserSettings = new Dictionary<String, String>();
UserSettings.Add("UseCompressedDateTimeFormat", "True");

// Convert the folder
results = PNConverter.ConvertFolder(@"C:\ALongPathTestFolder",
    true, // include subfolders
    ".*", // filter
    String.Empty, // exclude filter
    strOutputFolder, // output folder
    true, // overwrite existing
    true, // remove file ext
    true, // create log
    "TIFF 200dpi OptimizedColor", // conversion settings
    String.Empty, // extensions profile
    String.Empty, // MIME profile
    UserSettings, // User settings, compressed datetime
    String.Empty, // not using remote conversion (DCOM)
    strCustomTempFolder, // use custom working folder
    strLogFile); // long path to log file
```

Controlling Parallel Document Conversion

When converting a folder of files or a list of files it is important to remember that these files can be processed **in parallel**, meaning that multiple files can be converted at the same time. This needs to be taken into consideration with folders and list of files to avoid name collisions and accidental overwrites of created files in the output folders.

The number of files that can be converted in parallel is firstly controlled by Document Conversion Service, up to the limits of its license model. Secondly, the PEERNET.ConvertUtility also submits the documents to the Document Conversion Service on parallel threads. The number of documents submitted is automatically determined based on the number of CPU's and cores on your system multiplied by two.

We recommend that you allow this value to be determined automatically for best performance. If you do need to customize how many documents you submit to Document Conversion Service in parallel, the conversion setting **NumberOfDocumentsInParallel** can be passed as additional *user settings* to control how many parallel threads the PEERNET.ConvertUtility uses.

Please note that this only applies to the [ConvertFolder](#) and [ConvertFileList](#) methods where you are processing multiple files.

```
PNConversionItem resultItem = null;

// Add the number of threads
Dictionary<String, String> customSettings = new Dictionary<String, String>();
customSettings[ "NumberOfDocumentsInParallel" ] = "6";

resultItem = PNConverter.ConvertFolder(@"C:\Test\InputFiles",
    true, // include subfolders
    "*.pdf", // filter
    String.Empty, // exclude filter
    @"C:\Test\Output", // output folder
    true, // overwrite existing
    false, // remove file ext
    false, // create log
    "TIFF 200dpi OptimizedColor", // settings
    String.Empty, // extensions profile
    String.Empty, // MIME profile
    customSettings, // User settings
    String.Empty, // not using remote conversion (DCOM)
    String.Empty, // use default working folder
    String.Empty);
```

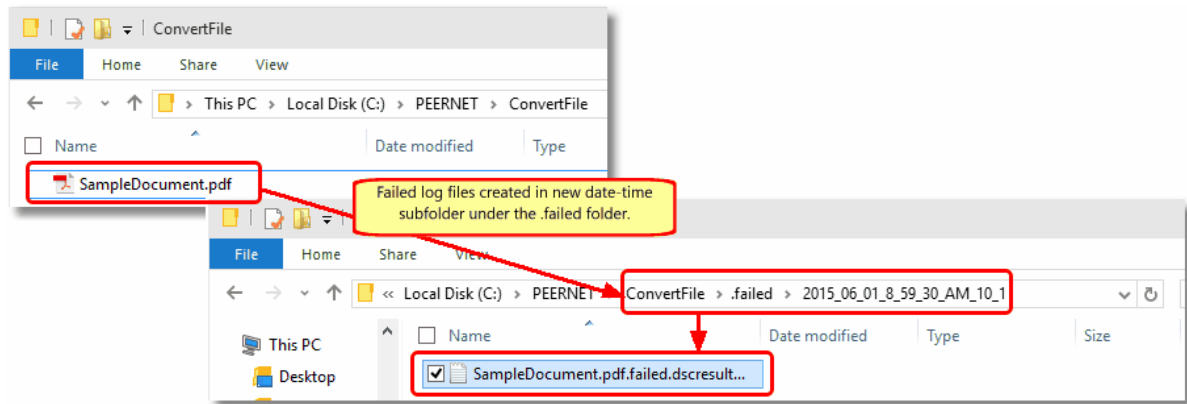
Controlling the Failed Results File Location

The results log file is the XML file representation of the [PNConversionItem](#) returned when calling [ConvertFile](#), [ConvertFileList](#), [ConvertFolder](#) and the XML file representation of the [PNCombineItem](#) when calling [CombineFiles](#).

This file is always created when a file fails to convert. The name of the results log file is based on the name of the original file and also indicates the conversion status. For example, a failed conversion results file for *SampleDocument.pdf* would be named *SampleDocument.pdf.failed.dcsresults*.

If a file has failed to convert, the default behavior when converting files, file lists and folder of files is to create a *.failed* folder in the same location as the source file. When combining files the *.failed* folder is created in the save location.

The conversion results log file is then saved in the *.failed* folder under a new subfolder created using the date and time of the conversion. This subfolder is created to keep subsequent runs separate and can be disabled.



Saving The Results Files in a Different Location

The location of these files can be customized and the use of the date and time named subfolder turned off with the following custom settings:

- FailedFolder
- UseDateTimeInFailedFolder

The code sample below shows how to override the default use of the date time folder under the .failed folder and to provide a specific folder in which to store the failed results log files.

```
IList<PNConversionItem> results = new List<PNConversionItem>();
String strOutputFolder = @"C:\Test\Output";
Dictionary<String, String> customSettings = new Dictionary<String, String>();

// Directory must exist
if ( !Directory.Exists(strOutputFolder) )
{
    Directory.CreateDirectory(strOutputFolder);
}

// Store .failed.dcsresults files in this folder
customSettings["FailedFolder"] = @"C:\Test\FailedFiles";

// DO NOT store results log files in date time folder under C:\Test\FailedFiles
customSettings["UseDateTimeInFailedFolder"] = "False";

// Convert the folder
results = PNConverter.ConvertFolder(@"C:\Test\InputFiles",
    true, // include subfolders
    ".*", // filter
    "*.tif|*.jpg|*.bmp", // exclude filter
    strOutputFolder, // output folder
    true, // overwrite existing
    false, // remove file ext
    false, // create log
    "TIFF 200dpi OptimizedColor", // settings
    String.Empty, // extensions profile
    String.Empty, // MIME profile
    customSettings, // User settings
    String.Empty, // not using remote conversion (DCOM)
    String.Empty, // use default working folder
    String.Empty);
```

Disable Creation of Failed Results Files

You can disable the creation of the conversion results log files with the setting **KeepFailedItemResultsFiles**. When this is set to false, the *.failed.dcsresults* files and the *.failed* folder will not be created, even when conversion does not succeed.

```
IList<PNConversionItem> results = new List<PNConversionItem>();
String strOutputFolder = @"C:\Test\Output";
Dictionary<String, String> customSettings = new Dictionary<String, String>();

// Directory must exist
if ( !Directory.Exists(strOutputFolder) )
{
    Directory.CreateDirectory(strOutputFolder);
}

// Set this to False to discard failed results log files
customSettings["KeepFailedItemResultsFiles"] = "False";

// Convert the folder
results = PNConverter.ConvertFolder(@"C:\Test\InputFiles",
    true, // include subfolders
    ".*", // filter
    "*.tif|*.jpg|*.bmp", // exclude filter
    strOutputFolder, // output folder
    true, // overwrite existing
    false, // remove file ext
    false, // create log
    "TIFF 200dpi OptimizedColor", // settings
    String.Empty, // extensions profile
    String.Empty, // MIME profile
    customSettings, // User settings
    String.Empty, // not using remote conversion (DCOM)
    String.Empty, // use default working folder,
    String.Empty);
```


Controlling the SmartInspect Logging Files

A SmartInspect log file is created each time a [ConvertFile](#), [ConvertFolder](#), [ConvertFileList](#) or [CombineFiles](#) method is called. If the conversion is successful, this log file is automatically deleted. If the conversion fails, the file is kept and stored in the Windows temp (%TEMP%) folder. Each logging file is assigned a unique date, time and thread prefix followed by a name that identifies which method was used.

These log files are a tracing of the entire conversion process and are not the same as the conversion results log files created when a conversion fails. These files can be viewed using the SmartInspect Redistributable Console.

Method	Sample Logging Filename
ConvertFile	2014_09_11_2_38_43_PM_4_PNConvertFile.sil
ConvertFileList	2014_09_11_2_41_56_PM_3_PNConvertFileList.sil
ConvertFolder	2014_09_12_3_35_37_PM_6_PNConvertFolder.sil
CombineFiles	2014_09_13_10_24_32_PM_2_PNConvertFile.sil

Saving The SmartInspect Log Files in a Different Location

You can customize where the SmartInspect log files are saved and how they are named through the parameter *ConvertFileProcessLoggingPath* on the methods [ConvertFile](#), [ConvertFolder](#), [ConvertFileList](#) or [CombineFiles](#).

This parameter can take a folder or a path to a filename. If a path without a trailing backslash is provided, the last part of the path is assumed to be a filename.

Pass ConvertFileProcessLoggingPath as...	Is interpreted as...
"C:\Test\LogFile"	Create the SmartInspect log file as C:\Test\LogFile.sil.
"C:\Test\LogFile\"	Create the SmartInspect log file as C:\Test\LogFile\datetime_PNConvertFile.sil
"C:\Test\LogFile\ConvertFileCustom.sil"	Create the SmartInspect log file as C:\Test\LogFile\ConvertFileCustom.sil

You can remove the unique date, time and thread prefix used in the log file naming by passing the custom setting **RemoveDateTimePrefixOnProcessingLoggingFiles** as *True*.

The code below will store all logging files for any failed conversion in the folder *C:\Test\SI\Logging* and remove the datetime prefix from all logging files. This will create a logging file named *PNConvertFolder.sil* as we are calling the *ConvertFolder* method.

```

IList<PNConversionItem> results = new List<PNConversionItem>();
String strOutputFolder = @"C:\Test\Output";
String strSILoggingFile = @"C:\Test\SI\Logging\";
Dictionary<String, String> customSettings = new Dictionary<String, String>();

// Directory must exist
if ( !Directory.Exists(strOutputFolder) )
{
    Directory.CreateDirectory(strOutputFolder);
}

// Remove datetime prefix from SI logging files
customSettings["RemoveDateTimePrefixOnProcessingLoggingFiles"] = "True";

// Convert the folder
results = PNConverter.ConvertFolder(@"C:\Test\InputFiles",
    true, // include subfolders
    ".*", // filter
    ".*tif|*.jpg|*.bmp", // exclude filter
    strOutputFolder, // output folder
    true, // overwrite existing
    false, // remove file ext
    false, // create log
    "TIFF 200dpi OptimizedColor", // settings
    String.Empty, // extensions profile
    String.Empty, // MIME profile
    customSettings, // User settings
    String.Empty, // not using remote conversion (DCOM)
    String.Empty, // use default working folder
    strSILoggingFile); // SI logging file location

```

Disable Creation of Logging Files

To disable the creation of the SmartInspect log files when a conversion fails, the custom setting **KeepFailedProcessingLoggingFiles** can be pass as *False*.

This setting can be overridden by the setting **AlwaysKeepProcessingLoggingFiles**, which when set to *True*, will create SmartInspect logging files for both successful and failed conversions. The logging files are still stored in the %TEMP% folder or the location specified by the *ConvertFileProcessLoggingPath* parameter.

```
IList<PNConversionItem> results = new List<PNConversionItem>();
String strOutputFolder = @"C:\Test\Output";
Dictionary<String, String> customSettings = new Dictionary<String, String>();

// Directory must exist
if ( !Directory.Exists(strOutputFolder) )
{
    Directory.CreateDirectory(strOutputFolder);
}

// Set this to True to discard all SI logging files
customSettings["KeepFailedProcessingLoggingFiles"] = "False";

// Convert the folder
results = PNConverter.ConvertFolder(@"C:\Test\InputFiles",
    true, // include subfolders
    ".*", // filter
    "*.tif|*.jpg|*.bmp", // exclude filter
    strOutputFolder, // output folder
    true, // overwrite existing
    false, // remove file ext
    false, // create log
    "TIFF 200dpi OptimizedColor", // settings
    String.Empty, // extensions profile
    String.Empty, // MIME profile
    customSettings, // User settings
    String.Empty, // not using remote conversion (DCOM)
    String.Empty, // use default working folder
    String.Empty);
```

Custom Settings for Logging Files

The table below lists all custom settings for controlling the SmartInspect logging files created through the PEER.NET.ConvertUtility methods.

Custom Setting	Description
RemoveDateTimePrefixOnProcessingLoggingFiles	Pass <i>True</i> to disable the adding of the unique date, time and thread prefix when a custom file name has not been specified in the <i>ConvertFileProcessLoggingPath</i> parameter.
KeepFailedProcessingLoggingFiles	Pass as <i>False</i> to disable the automatic creation of SmartInspect logging files when conversion fails. This setting can be overridden by <i>AlwaysKeepProcessingLoggingFiles</i> .
AlwaysKeepProcessingLoggingFiles	When set to <i>True</i> , the SmartInspect logging files are always created in the %TEMP% or other specified folder for both successful and failed conversions. If set to <i>False</i> , no logging files are created. This setting will override the <i>KeepFailedProcessingLoggingFiles</i> setting.

Waiting for Document Conversion Service to be Ready to Convert

The Document Conversion Service must be running, either locally or on a remote computer for files or folders of files to be converted. If it is not running the [ConvertFile](#), [ConvertFolder](#), [ConvertFileList](#) and [CombineFiles](#) methods will all return immediately with an error.

In some scenarios, such as using these methods from another long running service, it may be desirable to wait for Document Conversion Service to be running instead of failing to convert the files.

This can be done in either of two ways:

- use the [IsConversionServiceRunning](#) check to detect the running state in your code and wait in your own loop accordingly
- pass a wait timeout value as a custom setting down the any of the methods to have the PEERNET.ConvertUtility wait

Detect Running State

The following code demonstrates using the [IsConversionServiceRunning](#) method to wait a maximum of fifteen minutes for the conversion service to be running. With this method, you can provide feedback or the ability to cancel on shorter intervals.

```
PNConversionItem resultItem = null;
Boolean bIsRunning = false;
int maxTimeout = 900000, // 15 minutes
    currentTimeout = 0;

do
{
    if (PNConverter.IsConversionServiceRunning(String.Empty))
    {
        bIsRunning = true;

        resultItem = PNConverter.ConvertFolder(@"C:\Test\InputFiles",
            true, // include subfolders
            "*.pdf", // filter
            String.Empty, // exclude filter
            @"C:\Test\Output", // output folder
            true, // overwrite existing
            false, // remove file ext
            false, // create log
            "TIFF 200dpi OptimizedColor", // settings
            String.Empty, // extensions profile
            String.Empty, // MIME profile
            null, // User settings
            String.Empty, // not using remote conversion
            String.Empty, // use default working folder
            String.Empty);
    }
    else
    {
        if (currentTimeout < maxTimeout)
        {
            // Sleep for 30 seconds
            Console.WriteLine("Waiting for service to be available...");
            Thread.Sleep(30000);
            currentTimeout += 30000;
        }
        else
        {
            Console.WriteLine("Timeout on available service. No conversion performed.");
            bIsRunning = true;
        }
    }
}
```

```
}  
} while (!bIsRunning );
```

Passing the Timeout Value to the PEERNET.ConvertUtility

The following code demonstrates passing the timeout value to the PEERNET.ConvertUtility to have the utility wait for the desired maximum of fifteen minutes for the conversion service to be running. In this case, if the conversion service is not running in the timeout value given the `PNConversionItem` object *resultItem* returned will contain the error message that the conversion service is not running.

```
PNConversionItem resultItem = null;  
  
// Pass down how long to wait for the conversion service to be running.  
Dictionary<String, String> customSettings = new Dictionary<String, String>();  
customSettings["SecondsToWaitForRunningConversionService"] = "900"; // 15 minutes max  
  
resultItem = PNConverter.ConvertFolder(@"C:\Test\InputFiles",  
    true, // include subfolders  
    "*.pdf", // filter  
    String.Empty, // exclude filter  
    @"C:\Test\Output", // output folder  
    true, // overwrite existing  
    false, // remove file ext  
    false, // create log  
    "TIFF 200dpi OptimizedColor", // settings  
    String.Empty, // extensions profile  
    String.Empty, // MIME profile  
    customSettings, // User settings  
    String.Empty, // not using remote conversion (DCOM)  
    String.Empty, // use default working folder  
    String.Empty);
```

Deploying Applications

When deploying applications build with PEER.NET.ConvertUtility, the following files must be included with your application.



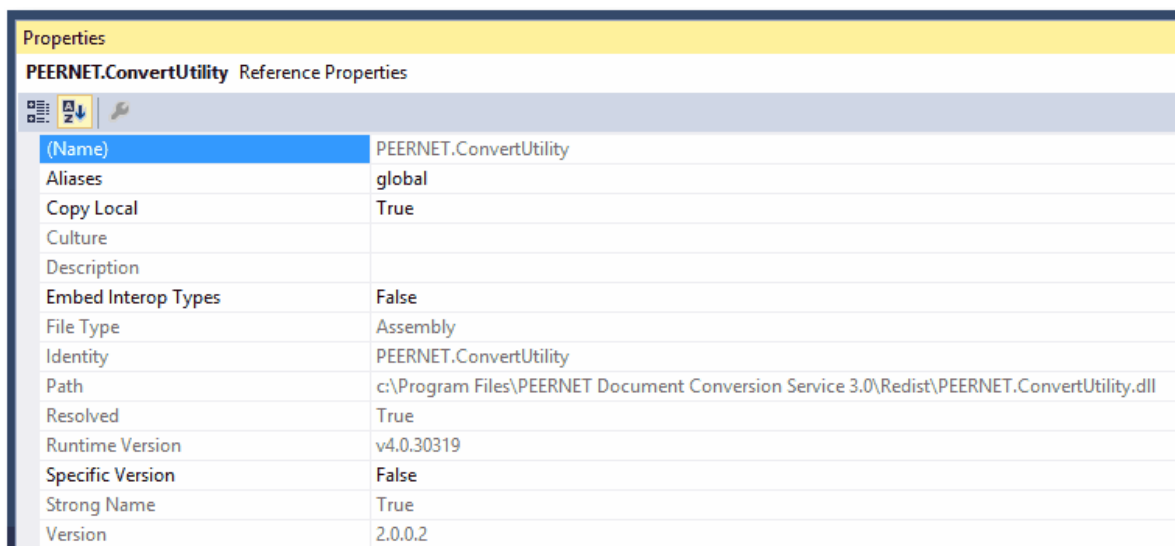
Changes to Files starting with version 3.27

Starting with Document Conversion Service 3.27, the Xfinium library used by PEER.NET.ConvertUtility is **Xfinium.Pdf.Win.dll**. Any programs using PEER.NET.ConvertUtility.dll and scripts used to copy the required files that use the old library name of XFinium.Pdf.Pcl.xml need to be updated.

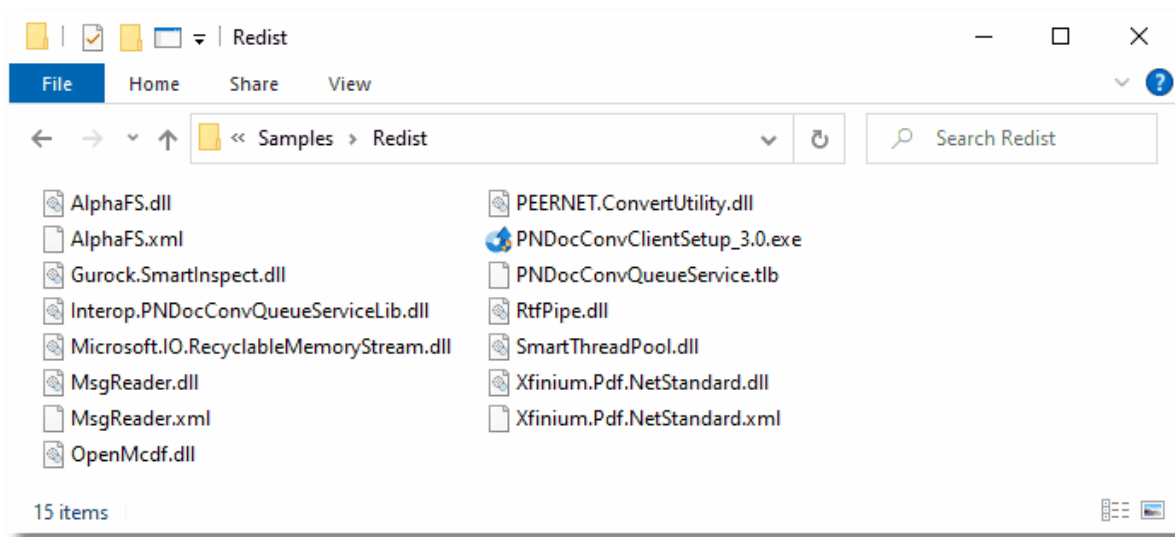
- PEER.NET.ConvertUtility.dll
- Gurock.SmartInspect.dll
- SmartThreadPool.dll
- Xfinium.Pdf.Win.dll, Xfinium.Pdf.Win.xml starting with Document Conversion Service 3.27
 - XFinium.Pdf.Pcl.dll, XFinium.Pdf.Pcl.xml prior to Document Conversion Service 3.27
- AlphaFS.dll
- AlphaFS.xml
- If your application will installed on client machines where Document Conversion Service is not installed, the Document Conversion Service Client Redistributable also needs to be installed with your application.
- Any custom profiles (see [Creating and Customizing Profiles](#)) that you created.

Adding the Reference to Your Application

When you add the PEER.NET.ConvertUtility.dll as a reference into your .NET application, set its *Copy Local* property to *True* to have this library and its dependencies automatically copied to your build output path when you do a build.



If you need to manually copy these files you can find them in the **\Samples\Redist** folder under the Document Conversion Service installation tree.



Installing the Document Conversion Service Client Redistributable With Your Application

When your application will be running on *client* computers where Document Conversion Service is not installed, meaning you are doing remote conversion, also called *client-server* conversion, the Document Conversion Service Client Redistributable will also need to be installed.

This redistributable can be installed as a separate step from your application, called from your installation, or you can bundle it with your own install by using command line arguments to run the install silently.

There are two types of setup that can be controlled from the command line - *BASIC*, and *FULL*. The BASIC setup only installs the required components for remote conversion in a client-server environment. The FULL setup will also install the Watch Folder Service and sample code, the command line conversion tools and all additional sample code.

When the client install is not run silently, the command line arguments are ignored.

```
PNDocConvClientSetup_3.0.exe /S  
PASSWORD="password"  
[SETUPTYPE=BASIC|FULL]  
[DCSUSER="domain\user"]
```

Sample Command Lines

```
PNDocConvClientSetup_3.0.exe /s PASSWORD="password"
```

Runs the basic client setup silently with no UI. The local DCSAdmin account will be created with the supplied password, or if it already exists, will be validated and used with the supplied password.

```
PNDocConvClientSetup_3.0.exe /s SETUPTYPE=BASIC DCSUSER=".MyLocalUser" PASSWORD="password"
```

Runs the basic client setup silently with no UI.

The local account MyLocalUser will be created with the supplied password, or if it already exists, will be validated and used with the supplied password.

```
PNDocConvClientSetup_3.0.exe /s SETUPTYPE=FULL DCSUSER="DOMAIN\MyUser" PASSWORD="password"
```

Runs the full client setup silently with no UI.

The domain account MyUser will be validated and used with the supplied password.

/S - Silent Install

This will run the installation silently with no wizard. If no SETUPTYPE is specified, then a BASIC install is done.

The client install also requires that the *PASSWORD=* variable be provided. When used without the *DCSUSER=* variable, the password is used to create or validate an existing *DCSAdmin* account. If not provided the setup will terminate.

PASSWORD="password"

The client install requires a user account with administrative privileges to initialize the services and configure for client-server conversion. A password must be supplied to create the account, or validate the account if an existing one is used. If the account cannot be validated the setup will terminate.

SETUPTYPE=BASIC|FULL

Choose the setup type - *BASIC* or *FULL*. The BASIC setup only installs the required components for remote conversion in a client-server environment. The FULL setup will also install the Watch Folder Service and sample code, the command line conversion tools and all additional sample code.

When this argument is not specified, a *BASIC* setup is installed.

DCSUSER="domain\user"

The services and configuration for client-server conversion require a user account, local or domain-level, that has administrative privileges. We normally recommend that you let us create and use our local account DCSAdmin.

If you cannot use this account you can specify here a different user. If using a domain account, you need to specify the domain and user name. The install process also needs to be able to validate the account. The setup will fail if the account cannot be validated. If you are using a different local account, specify the local account using the dot syntax for local, ".MyLocalUser".

PEERNET.ConvertUtility Namespace

The PEERNET.ConvertUtility namespace contains the main classes that allow you to communicate with Document Conversion Service and convert files from your own applications.







If you are new to the PEERNET.ConvertUtility namespace, see the [C# Tutorial](#) or the [Visual Basic .NET Tutorial](#) for step-by-step instructions to get you started.









Note


All public constructors, methods and properties are documented here. Constructors, methods, and properties that are visible through the Object Browser or through Intellisense but are not documented here are private to the namespace and should not be used.

Objects

Object	Description
 PNConverter	This class contains all of the static conversion methods for converting files, folders of files and collections of individual files.
 PNConvertFileInfo	Describes a single input file to be converted, the output path for that file and an optional collection of settings to use when converting the file. This object is used with the PNConvert method ConvertFileList .
 PNConversionItem	This object, or a collection of these objects, is returned by all of the conversion methods in PNConverter except for CombineFiles . It contains information about the original conversion request and a PNConversionResult object containing information about the conversion results.
 PNCombineItem	This object is returned by the PNConverter method CombineFiles . It contains information about the original file combine (append) request, a list of the output files created, and an inner list of PNConversionResult items for each file included as part of the combine operation.
 PNConversionResult	This object contain the list of files created, or if no files were created, a list of errors and messages explaining why the conversion failed.
 PNConversionResultError	A collection of these objects, one for each error, is returned as part of the NConversionResult object if a conversion fails.

Object	Description
 PNConversionResultMessage	A collection of these objects containing informational messages is returned as part of the PNConversionResult object. This collection can be empty.
 PNConversionResultOutputFile	A collection of one or more of these objects is returned as part of the PNConversionResult object. This object contains the output path to the converted file.
 PNConversionResultOutputFileRenderedPage	A collection of one or more of these objects is returned as part of the PNConversionResult object. This object contains information about this created page such as the page number in the file and the resolution, orientation and bit level of the created page.
 PNConversionResultPrintJob	A collection of one or more of these objects is returned as part of the PNConversionResult object. This object contains information about the print job, such as the number of pages spooled or pages printed and the status of the print job that was used to convert the input file.
 PNConversionResultPrintJobPrintedPage	A collection of one or more of these objects is returned as part of the PNConversionResult object. This object contains information about each printed page of the input file, such as the resolution, orientation, and page number of the print job.
 PNProfile	Provides an interface for working with the profiles that Document Conversion Service uses to convert documents. Profiles control both the type of file created and optionally the behavior of the converters.
 PNSetting	An object that represents a setting as a name-value pair.

Enumerations

Object	Description
 PNConvertResultStatus	Conversion status result as a short string message.







PNConverter

Description

PNConverter is the main class in the PEERNET.ConvertUtility .NET library. It contains all of the methods that you will use to convert files, folders of files and list of files in your application code.

It also contains the method [IsConversionServiceRunning](#) that allows you to synchronize with the Document Conversion Service being running and ready to convert.

Methods

 ConvertFile	Converts a single file, using the supplied conversion settings, to the specified output folder. Can optionally specify a custom name.
 ConvertFileList	Converts a list of files. Uses the PNConvertFileInfo class to build the list of files.
 ConvertFolder	Converts all files in the folder that match the given file extension filter. Can optionally recurse into subdirectories.
 CombineFiles	Combines the list of files into a single, multipaged output file. Supports TIFF and PDF output.
 CombineFolder	Combines all files in the folder, and optionally all subfolders into a single, multipaged output file. Supports TIFF and PDF output.
 IsConversionServiceRunning	Query if Document Conversion Service is running. The service must be running, either locally or remotely on another computer for conversion to take place.

Methods

ConvertFile

Description

Static method.

Converts a file using the requested conversion settings.

Syntax

```
PNConverter.ConvertFile(InputFile, OutputFolder, OutputName,
                        OverwriteExisting, RemoveFileExtension, CreateResultsLogFiles,
                        SettingsProfile, ExtensionsProfile, MIMEProfile, UserSettings,
                        RemoteComputerName, ConversionWorkingFolder,
                        ConvertFileProcessLoggingPath)
```

```
PNConverter.ConvertFile(InputFile, OutputFolder, OutputName,  
                        OverwriteExisting, RemoveFileExtension, CreateResultsLogFiles,  
                        SettingsList, ExtensionsProfile, MIMEProfile, UserSettings,  
                        RemoteComputerName, ConversionWorkingFolder,  
                        ConvertFileProcessLoggingPath)
```

Returns a [PNConversionItem](#) object that contains information about the original conversion request and an inner [PNConversionResult](#) object containing information about the conversion results.

Parameters

String InputFile

The full path to the file to convert. This can be on the local computer, on a shared location using a mapped drive letter or by passing a UNC formatted file path.

String OutputFolder

Full path to the save file location, or *String.Empty* to create the new file in the same location as the source file. If the path doesn't exist, the conversion will fail. This folder must be created before the call to `ConvertFolder` is made.

If a file of the same name already exists in the save file location, the conversion will fail. Pass **True** for *OverwriteExisting* to allow file overwriting.

String OutputName

The name to use for the output file, without extension. The default file extension for the type of file being created will always be added to the name provided here.

Pass *String.Empty* to use the base name of the source file. When using the source name, the extension of the source file is always used as part of the new file name unless *RemoveFileExtension* is set to **True**.

Boolean OverwriteExisting

Set to **True** to overwrite existing files, or **False** to fail conversion when a file of the same name already exists in the save location.

Boolean RemoveFileExtension

This parameter is ignored if you have provided a file name in *OutputName*.

If *OutputName* is not specified, the name of the each output file is created using the base name and file extension of the original file. This is done to prevent name collision when you have two files in the folder with the same base name. Set this to **True** if you do not want the original file name extension as part of your output file name.

Boolean CreateResultsLogFile

Pass **True** to create a results log file containing a complete snapshot of the conversion information. This file is saved with the output file. The name of the results log file is based on the name of the original file and also indicates the conversion status. For example, when converting *Sample.doc*, a successful conversion will create *Sample.doc.succeeded.dcsresults* and if the conversion did not succeed, the file would be named *Sample.doc.failed.dcsresults*.

These log files can later be read from disk using the [DeserializeFromXML](#) method of the [PNConversionItem](#) class.

String SettingsProfile

The name of the profile to use, with or without the XML extension. Document Conversion Service includes several sample profiles for common types of output files for your use, or you can create your own and pass in a full path to your custom profile. See [Creating and Customizing Profiles](#) for a list of included profiles and how to create your own.

IDictionary<String, String> SettingsList

A dictionary of name\value pairs of settings that describes the conversion options. The name\value pairs that make up this dictionary are the same settings that are used to create the XML-formatted profiles included with Document Conversion Service. See [Conversion Settings](#) for a list of all of the settings that are available.

String ExtensionsProfile

Name of the file mapping profile XML file, with or without the XML extension. Providing this parameter is optional and an internal default mapping is provided. You would only need to provide this file if you wanted to override the default file extension to converter mappings provided.

String MimeProfile

Reserved for future use - pass String.Empty.

IDictionary<String, String> UserSettings

Optional. Pass a dictionary of additional conversion settings. These settings will override any matching settings in either *SettingsProfile* or *SettingsList*. Pass *null* if not using.

String RemoteComputerName

Optional. Pass *String.Empty* if you are converting locally or the name of the remote computer where Document Conversion Service is running. When converting remotely, a *ConversionWorkingFolder* must also be provided.

String ConversionWorkingFolder

Used to provide a shared path to be used when doing remote conversion or an alternate temporary working instead of our default of the Windows TEMP folder.

This setting is required when *RemoteComputerName* is provided for remote conversion (DCOM) as both the local and the remote computer need access to a shared path in which to do the conversion. Pass *String.Empty* if you are not using this setting.

When not doing remote conversion, this setting is not required in most cases but can be useful when dealing with folder and file names longer than 255 characters. When converting a file, the conversion tool copies the file and performs the conversion in temporary staging and working folders created on demand in the default Windows temp folder. These folders need to be less than 255 characters as required by the underlying programs used by Document Conversion Service to perform conversions. When dealing with these long path and file names the default folders created can occasionally cause path names that are too long for Document Conversion Service to process. When this happens this switch can be used to set the temporary folder to a shorter path to allow processing. Again, pass *String.Empty* if you are not using this setting.

String ConvertFileProcessLoggingPath

Optional. Specify a path to a folder in which to store the SmartInspect logs files of any failed conversion process. These files are stored in the temp folder by default and can be viewed using the SmartInspect Redistributable Console. These log files are a tracing of the entire conversion process and are not the same as the conversion results log files created when a conversion fails. See [Controlling the SmartInspect Logging Files](#) to change where these files are stored, how they are named, or to disable creation of these files.

Remarks

If the conversion does not succeed, a folder named *.failed* is created in the same location as the source file. Inside the *.failed* folder is a timestamped folder that contains the conversion results log file that is always created with each failed file. The results log file named based on the source file's name and its conversion status. For example, if converting *Document.doc* failed the results log file would be named *Document.doc.failed.dcsresults*. See [Controlling the Failed Results File Location](#) to store these files in a different location or to disable the creation of these file.

Exceptions

Exception	Condition
ArgumentException	An empty, or badly formatted profile was passed for <i>SettingsProfile</i> . An empty list was passed for <i>SettingsList</i> . An empty, or badly formatted profile was passed for <i>ExtensionsProfile</i> . Null or empty string passed for <i>InputFile</i> . A name for <i>RemoteComputerName</i> was passed but no corresponding <i>ConversionWorkingFolder</i> specified.
FileNotFoundException	<i>InputFile</i> doesn't exist.
DirectoryNotFoundException	When a path to <i>OutputFolder</i> is specified but does not exist or is invalid. When <i>ConversionWorkingFolder</i> is specified but does not exist or is invalid.

See Also:

[ConvertFileList](#) [ConvertFolder](#) [CombineFiles](#) [CombineFolder](#) [IsConversionServiceRunning](#)



Code Sample - C#

```
PNConversionItem resultItem = null;

resultItem = PNConverter.ConvertFile(@"C:\Test\File.pdf",
                                     @"C:\Test\Output",
                                     @"ConvertedFromPDF",
                                     true, // overwrite existing
                                     false, // do not remove file ext
                                     false, // do not create log
                                     "TIFF 200dpi OptimizedColor",
                                     String.Empty,
                                     String.Empty,
                                     null, // no custom user settings
                                     String.Empty, // not using DCOM
                                     String.Empty, // use default working folder
                                     String.Empty); // do not use custom log folder
```


**Code Sample - VB.NET**

```
Dim resultItem As PNConversionItem

resultItem = Nothing

resultItem = PNConverter.ConvertFile("C:\Test\File.pdf", _
    "C:\Test\Output", _
    "ConvertedFromPDF", _
    True, _
    False, _
    False, _
    "TIFF 200dpi OptimizedColor", _
    String.Empty, _
    String.Empty, _
    Nothing, _
    String.Empty, _
    String.Empty, _
    String.Empty)
```

ConvertFileList**Description**

Static method.

Converts a list of files using the requested conversion settings.

Syntax

```
PNConverter.ConvertFileList(FileInfoList, OutputFolder, OutputName,
    OverwriteExisting, RemoveFileExtension, CreateResultsLogFiles,
    SettingsProfile, ExtensionsProfile, MIMEProfile, UserSettings,
    RemoteComputerName, ConversionWorkingFolder,
    ConvertFileProcessLoggingPath)
```

```
PNConverter.ConvertFileList(FileInfoList, OutputFolder, OutputName,
    OverwriteExisting, RemoveFileExtension, CreateResultsLogFiles,
    SettingsList, ExtensionsProfile, MIMEProfile, UserSettings,
    RemoteComputerName, ConversionWorkingFolder,
    ConvertFileProcessLoggingPath)
```

```
PNConverter.ConvertFileList(FileList, OutputFolder, OutputName,
    OverwriteExisting, RemoveFileExtension, CreateResultsLogFiles,
    SettingsProfile, ExtensionsProfile, MIMEProfile, UserSettings,
    RemoteComputerName, ConversionWorkingFolder,
    ConvertFileProcessLoggingPath)
```

```
PNConverter.ConvertFileList(FileList, OutputFolder, OutputName,
    OverwriteExisting, RemoveFileExtension, CreateResultsLogFiles,
    SettingsList, ExtensionsProfile, MIMEProfile, UserSettings,
    RemoteComputerName, ConversionWorkingFolder,
    ConvertFileProcessLoggingPath)
```

Returns an *ICollection* of [PNConversionItem](#) objects, one for each file in the supplied list of files. Each *PNConversionItem* contains information about the original conversion request and an inner [PNConversionResult](#) object containing information about the conversion results.

Parameters

ICollection<PNConvertFileInfo> FileInfoList

A list of [PNConvertFileInfo](#) objects. Each *PNConvertFileInfo* object describes a single input file to be converted, the output path for that file and an optional collection of settings to use when converting the file. If the output path for the file is not set in the *PNConvertFileInfo* object then the *OutputFolder* parameter is used.

ICollection<String> FileList

A list of strings representing the full paths of each file to convert. The files can be on the local computer, on a shared location using a mapped drive letter or by passing a UNC formatted file path.

String OutputFolder

Full path to the save file location, or *String.Empty* to create the new file in the same location as the source file.

This folder must be created before the call to *ConvertFileList* is made. If the path doesn't exist or a file of the same name already exists in the save file location, the conversion will fail. Pass **True** for *OverwriteExisting* to allow file overwriting.

If *FileInfoList* is used and the an output path is specified in the *PNConvertFileInfo* object, this parameter is ignored.

String OutputName

The name to use for the output file, without extension. The default file extension for the type of file being created will always be added to the name provided here.

Pass *String.Empty* to use the base name of the source file. When using the source name, the extension of the source file is used as part of the new file name unless *RemoveFileExtension* is set to **True**.

String OverwriteExisting

Set to **True** to overwrite existing files, or **False** to fail conversion when a file of the same name already exists in the save location.

String RemoveFileExtension

This parameter is ignored if you have provided a file name in *OutputName*.

If *OutputName* is not specified, the name of the each output file is created using the base name and file extension of the original file. This is done to prevent name collision when you have two files in the folder with the same base name. Set this to **True** if you do not want the original file name extension as part of your output file name.

String CreateResultsLogFile

Pass **True** to create a results log file containing a complete snapshot of the conversion information for each file. This file is saved with each output file. The name of the results log file is based on the name of the original file and also indicates the conversion status. For example, when converting *Sample.doc*, a successful conversion will create *Sample.doc.succeeded.dcsresults* and if the conversion did not succeed, the file would be named *Sample.doc.failed.dcsresults*.

These log files can later be read from disk using the [DeserializeFromXML](#) method of the [PNConversionItem](#) class.

String SettingsProfile

The name of the profile to use, with or without the XML extension. Document Conversion Service includes several sample profiles for common types of output files for your use, or you can create your own and pass in a full path to your custom profile. See [Creating and Customizing Profiles](#) for a list of included profiles and how to create your own.

IDictionary<String, String> SettingsList

A dictionary of name/value pairs of settings that describes the conversion options. The name/value pairs that make up this dictionary are the same settings that are used to create the XML-formatted profiles included with Document Conversion Service. See [Conversion Settings](#) for a list of all of the settings that are available.

String ExtensionsProfile

Name of the file mapping profile XML file, with or without the XML extension. Providing this parameter is optional and an internal default mapping is provided. You would only need to provide this file if you wanted to override the default file extension to converter mappings provided.

String MimeProfile

Reserved for future use - pass String.Empty.

IDictionary<String, String> UserSettings

Optional. Pass a dictionary of additional conversion settings. These settings will override any matching settings passed in for *SettingsProfile* or *SettingsList*. Pass *null* if not using.

String RemoteComputerName

Optional. Pass *String.Empty* if you are converting locally or the name of the remote computer where Document Conversion Service is running. When converting remotely, a *ConversionWorkingFolder* must also be provided.

String ConversionWorkingFolder

Used to provide a shared path to be used when doing remote conversion or an alternate temporary working instead of our default of the Windows TEMP folder. This setting is required when *RemoteComputerName* is provided for remote conversion (DCOM) as both the local and the remote computer need access to a shared path in which to do the conversion. Pass *String.Empty* if you are not using this setting.

When not doing remote conversion, this setting is not required in most cases but can be useful when dealing with folder and file names longer than 255 characters. When converting a file, the conversion tool copies the file and performs the conversion in temporary staging and working folders created on demand in the default Windows temp folder. These folders need to be less than 255 characters as required by the underlying programs used by Document Conversion Service to perform conversions. When dealing with these long path and file names the default folders created can occasionally cause path names that are too long for Document Conversion Service to process. When this happens this switch can be used to set the temporary folder to a shorter path to allow processing. Again, pass *String.Empty* if you are not using this setting.

String ConvertFileProcessLoggingPath

Optional. Specify a path to a folder in which to store the SmartInspect logs files of any failed conversions. These files are stored in the temp folder by default and can be viewed using the SmartInspect Redistributable Console. These log files are a tracing of the entire conversion process and are not the same as the conversion results log files created when a conversion fails. See [Controlling the SmartInspect Logging Files](#) to change where these files are stored, how they are named, or to disable creation of these files.

Remarks

If conversion of any of the files in the list does not succeed, a folder named *.failed* is created in the same location as that source file. Inside the *.failed* folder is a timestamped folder that contains the conversion results log file that is always created with each failed file. The results log file named based on the source file's name and its conversion status. For example, if converting *Document.doc* failed the results log file would be named *Document.doc.failed.dcsresults*. See [Controlling the Failed Results File Location](#) to store these files in a different location or to disable the creation of these file.

Exceptions

Exception	Condition
ArgumentException	An empty, or badly formatted profile was passed for <i>SettingsProfile</i> . An empty list was passed for <i>SettingsList</i> . An empty, or badly formatted profile was passed for <i>ExtensionsProfile</i> . An empty list was passed for <i>FileInfoList</i> or <i>FileList</i> . A name for <i>RemoteComputerName</i> was passed but no corresponding <i>ConversionWorkingFolder</i> specified.
FileNotFoundException	One of the input files in the <i>FileInfoList</i> or <i>FileList</i> does not exist or cannot be accessed.
DirectoryNotFoundException	One of the output paths in the <i>FileInfoList</i> does not exist, or when <i>OutputFolder</i> is specified but the path does not exist or is invalid. When <i>ConversionWorkingFolder</i> is specified but does not exist or is invalid.

See Also:

[ConvertFile](#) [ConvertFolder](#) [CombineFiles](#) [CombineFolder](#) [IsConversionServiceRunning](#)



Code Sample - C#

```

IList<PNConversionItem> results = new List<PNConversionItem>();
IList<PNConvertFileInfo> filesToTIFF = new List<PNConvertFileInfo>();

filesToTIFF.Add(new PNConvertFileInfo(@"C:\Test\File1.pdf",
                                     @"C:\Test\Output\1");

filesToTIFF.Add(new PNConvertFileInfo(@"C:\Test\File2.pdf",
                                     @"C:\Test\Output\2");

results = PNConverter.ConvertFileList(filesToTIFF,
                                     String.Empty,
                                     String.Empty,
                                     true, // overwrite existing
                                     false, // do not remove file ext
                                     false, // do not create log
                                     "TIFF 200dpi OptimizedColor",
                                     String.Empty,
                                     String.Empty,
                                     null, // no custom user settings
                                     String.Empty, // not using DCOM
                                     String.Empty, // use default working folder
                                     String.Empty); // do not use custom log folder

```

**Code Sample - VB.NET**

```

Dim filesToTIFF As IList(Of PNConvertFileInfo)
Dim results As IList(Of PNConversionItem)

filesToTIFF.Add(New PNConvertFileInfo("C:\Test\File1.pdf", _
                                     "C:\Test\Output\1"))
filesToTIFF.Add(New PNConvertFileInfo("C:\Test\File2.pdf", _
                                     "C:\Test\Output\2"))

results = PNConverter.ConvertFileList(filesToTIFF, _
                                     String.Empty, _
                                     String.Empty, _
                                     True, _
                                     False, _
                                     False, _
                                     "TIFF 200dpi OptimizedColor", _
                                     String.Empty, _
                                     String.Empty, _
                                     Nothing, _
                                     String.Empty, _
                                     String.Empty, _
                                     String.Empty)

```

ConvertFolder**Description**

Static method.

Converts all files in the folder, and optionally all subfolders, using the requested conversion settings.

A filter pattern can be used to only process files in the folder that match the provided pattern, such as *.doc to process all Word documents, or ABC* to process all files that start with the letters ABC.

An exclude filter is also provided, to allow you to skip files that match the exclude pattern.

Syntax

```

PNConverter.ConvertFolder(InputFolder, IncludeSubFolders, Filter, ExcludeFilter,
                          OutputFolder, OverwriteExisting, RemoveFileExtension, CreateResultsLogFiles,
                          SettingsProfile, ExtensionsProfile, MIMEProfile, UserSettings,
                          RemoteComputerName, ConversionWorkingFolder,
                          ConvertFolderProcessLoggingFilePath)

PNConverter.ConvertFolder(InputFolder, IncludeSubFolders, Filter, ExcludeFilter,
                          OutputFolder, OverwriteExisting, RemoveFileExtension, CreateResultsLogFiles,
                          SettingsProfile, ExtensionsProfile, MIMEProfile, UserSettings,
                          RemoteComputerName, ConversionWorkingFolder,
                          ConvertFolderProcessLoggingFilePath,
                          SortOrder, SortMode)

PNConverter.ConvertFolder(InputFolder, IncludeSubFolders, Filter, ExcludeFilter,
                          OutputFolder, OverwriteExisting, RemoveFileExtension, CreateResultsLogFiles,
                          SettingsList, ExtensionsProfile, MIMEProfile, UserSettings,

```

```
RemoteComputerName, ConversionWorkingFolder,
ConvertFolderProcessLoggingFilePath)

PNConverter.ConvertFolder(InputFolder, IncludeSubFolders, Filter, ExcludeFilter,
OutputFolder, OverwriteExisting, RemoveFileExtension, CreateResultsLogFiles,
SettingsList, ExtensionsProfile, MIMEProfile, UserSettings,
RemoteComputerName, ConversionWorkingFolder,
ConvertFolderProcessLoggingFilePath
SortOrder, SortMode)
```

Returns an IList of [PNConversionItem](#) objects, one for each file in the folder (and subfolders, if selected) that matched the filter pattern. Each PNConversionItem contains information about the original conversion request and an inner [PNConversionResult](#) object containing information about the conversion results.

Parameters

String InputFolder

The full path to the folder containing the files to convert. This can be on the local computer, on a shared location using a mapped drive letter or by passing a UNC formatted file path.

String IncludeSubFolders

Set to **True** to include the subfolders under the folder when building the list of files to be converted.

String Filter

A filter to process only the files matching the filter pattern, such as using *.pdf to only process files ending with the .PDF or .pdf extension. Multiple filters can be combined using the pipe (|) character, such as *.doc|*.pdf to process only Word and PDF files.

Hidden and system files are ignored, and the search pattern filters files based on a regular expression match of the long name of a file. The filter defaults to all files in the folder (*.*) if *String.Empty* or *null* are passed for the filter.

String ExcludeFilter

After the *Filter* pattern is used to get the list of files to convert from the *InputFolder*, the exclude filter can then be applied to that list to remove files that match the exclude pattern. Multiple excluded filters are combined using the pipe (|) character, such as *.pdf|*.xml to process all files returned except PDF and XML files.

If *String.Empty* or *null* is passed then no files are excluded.

String OutputFolder

Full path to the save file location. If this argument is not specified, a .new folder named *.converted* is created in the same location as the source file and all output files are saved there.

If the path doesn't exist, the conversion will fail, or if a file of the same name already exists in the save file location, the conversion will fail. Pass **True** for *OverwriteExisting* to allow file overwriting.

String OverwriteExisting

Set to **True** to overwrite existing files, or **False** to fail conversion when a file of the same name already exists in the save location.

String RemoveFileExtension

Set this to **True** if you do not want the original file name extension as part of your output file name. Normally the name of the each output file is created using the base name and file extension of the original file to prevent name collision when you have two files in the folder with the same base name.

String CreateResultsLogFile

Pass **True** to create a results log file containing a complete snapshot of the conversion information for each file. This log file is saved with each output file. The name of the results log file is based on the name of the original file and also indicates the conversion status. For example, when converting *Sample.doc*, a successful conversion will create *Sample.doc.succeeded.dcsresults* and if the conversion did not succeed, the file would be named *Sample.doc.failed.dcsresults*.

These log files can later be read from disk using the [DeserializeFromXML](#) method of the [PNConversionItem](#) class.

String SettingsProfile

The name of the profile to use, with or without the XML extension. Document Conversion Service includes several sample profiles for common types of output files for your use, or you can create your own and pass in a full path to your custom profile. See [Creating and Customizing Profiles](#) for a list of included profiles and how to create your own.

IDictionary<String, String> SettingsList

A dictionary of name/value pairs of settings that describes the conversion options. The name/value pairs that make up this dictionary are the same settings that are used to create the XML-formatted profiles included with Document Conversion Service. See [Conversion Settings](#) for a list of all of the settings that are available.

String ExtensionsProfile

Name of the file mapping profile XML file, with or without the XML extension. Providing this parameter is optional and an internal default mapping is provided. You would only need to provide this file if you wanted to override the default file extension to converter mappings provided.

String MimeProfile

Reserved for future use - pass String.Empty.

IDictionary<String, String> UserSettings

Optional. Pass a dictionary of additional conversion settings. These settings will override any matching settings in either *SettingsProfile* or *SettingsList*. Pass *null* if not using.

String RemoteComputerName

Optional. Pass *String.Empty* if you are converting locally or the name of the remote computer where Document Conversion Service is running. When converting remotely, a *ConversionWorkingFolder* must also be provided.

String ConversionWorkingFolder

Used to provide a shared path to be used when doing remote conversion or an alternate temporary working instead of our default of the Windows TEMP folder.

This setting is required when *RemoteComputerName* is provided for remote conversion (DCOM) as both the local and the remote computer need access to a shared path in which to do the conversion. Pass *String.Empty* if you are not using this setting.

When not doing remote conversion, this setting is not required in most cases but can be useful when dealing with folder and file names longer than 255 characters. When converting a file, the conversion tool copies the file and performs the conversion in temporary staging and working

folders created on demand in the default Windows temp folder. These folders need to be less than 255 characters as required by the underlying programs used by Document Conversion Service to perform conversions. When dealing with these long path and file names the default folders created can occasionally cause path names that are too long for Document Conversion Service to process. When this happens this switch can be used to set the temporary folder to a shorter path to allow processing. Again, pass *String.Empty* if you are not using this setting.

String ConvertFileProcessLoggingPath

Optional. Specify a path to a folder in which to store the SmartInspect logs files of any failed conversions. These files are stored in the temp folder by default and can be viewed using the SmartInspect Redistributable Console. These log files are a tracing of the entire conversion process and are not the same as the conversion results log files created when a conversion fails. See [Controlling the SmartInspect Logging Files](#) to change where these files are stored, how they are named, or to disable creation of these files.

[PNFileSortMode](#) *SortMode*

Optional, controls the sort order of the list of files returned from the *InputFolder*. Files can be sorted by name, date created or date modified. Default is *None* when not specified.

[PNFileSortOrder](#) *SortOrder*

Optional, returns the files in *Ascending* (0-9, A-Z) or *Descending* (Z-A, 9-0) order. Default is *Ascending* when not specified.

Remarks

If conversion of any of the files in the folder does not succeed, a folder named *.failed* is created in the same location as that file. Inside the *.failed* folder is a timestamped folder that contains the conversion results log file that is always created with each failed file. The results log file named based on the source file's name and its conversion status. For example, if converting *Document.doc* failed the results log file would be named *Document.doc.failed.dcsresults*. See [Controlling the Failed Results File Location](#) to store these files in a different location or to disable the creation of these file.

Exceptions

Exception	Condition
ArgumentException	An empty, or badly formatted profile was passed for <i>SettingsProfile</i> An empty list was passed for <i>SettingsList</i> An empty, or badly formatted profile was passed for <i>ExtensionsProfile</i> . Null or empty string passed for <i>InputFile</i> . A name for <i>RemoteComputerName</i> was passed but no corresponding <i>ConversionWorkingFolder</i> specified.
FileNotFoundException	<i>InputFile</i> doesn't exist.
DirectoryNotFoundException	The path to <i>InputFolder</i> is specified but does not exist or is invalid. The path to <i>OutputFolder</i> is specified but does not exist or is invalid. The path to <i>ConversionWorkingFolder</i> is specified but does not exist or is invalid.

See Also:

[ConvertFile](#) [ConvertFileList](#) [CombineFiles](#) [CombineFolder](#) [IsConversionServiceRunning](#)

**Code Sample - C#**

```

IList<PNConversionItem> results = new List<PNConversionItem>();

// Convert all files in C:\Test\Input except TIFF images, include subfolders
results = PNConverter.ConvertFolder(@"C:\Test\Input\", true,
    "*.\"", "*.tif",
    @"C:\Test\Output\",
    true, // overwrite existing
    false, // do not remove file ext
    false, // do not create log
    "TIFF 200dpi OptimizedColor",
    String.Empty,
    String.Empty,
    null, // no custom user settings
    String.Empty, // not using DCOM
    String.Empty, // use default working folder
    String.Empty); // do not use custom log folder

```

**Code Sample - VB.NET**

```

Dim results As IList(Of PNConversionItem)

' Convert all files in C:\Test\Input except TIFF images, include subfolders
resulta = PNConverter.ConvertFolder("C:\Test\Input\", _
    "*.\"", "*.tif", _
    "C:\Test\Output\"
    True, _
    False, _
    False, _
    "TIFF 200dpi OptimizedColor", _
    String.Empty, _
    String.Empty, _
    Nothing, _
    String.Empty, _
    String.Empty, _
    String.Empty)

```

CombineFiles**Description**

Static method.

Converts and combines the list of files using the requested conversion settings. The files are combined in the order in which they are given.

The conversion settings passed in determine how the files are combined. For instance, passing conversion settings to create a multipaged PDF file will combine all input files into a single, multipage PDF file, while passing in the conversion settings to create serialized TIFF images will result in a serialized sequence of TIFF images, one for each page of each file.

Syntax

```
PNConverter.CombineFiles(FileList, OutputFolder, OutputName,  
    OverwriteExisting, CreateResultsLogFiles,  
    SettingsProfile, ExtensionsProfile, MIMEProfile, UserSettings,  
    RemoteComputerName, ConversionWorkingFolder,  
    ConvertFileProcessLoggingPath)
```

```
PNConverter.CombineFiles(FileList, OutputFolder, OutputName,  
    OverwriteExisting, CreateResultsLogFiles,  
    SettingsList, ExtensionsProfile, MIMEProfile, UserSettings,  
    RemoteComputerName, ConversionWorkingFolder,  
    ConvertFileProcessLoggingPath)
```

```
PNConverter.CombineFiles(FileInfoList, OutputFolder, OutputName,  
    OverwriteExisting, CreateResultsLogFiles,  
    SettingsProfile, ExtensionsProfile, MIMEProfile, UserSettings,  
    RemoteComputerName, ConversionWorkingFolder,  
    ConvertFileProcessLoggingPath)
```

```
PNConverter.CombineFiles(FileInfoList, OutputFolder, OutputName,  
    OverwriteExisting, CreateResultsLogFiles,  
    SettingsList, ExtensionsProfile, MIMEProfile, UserSettings,  
    RemoteComputerName, ConversionWorkingFolder,  
    ConvertFileProcessLoggingPath)
```

Returns a [PNCombineItem](#) object which contains a collection of [PNConversionResult](#) objects, one for each file in the supplied list of files added to the combined file. The [PNCombineItem](#) object contains a list of files used in the combine process, and a list of the resulting combined files as well as other information about the original combine request. Each inner [PNConversionResult](#) object contains information about the conversion results for a file in the combine set passed.

Parameters

IList<PNConvertFileInfo> FileInfoList

A list of [PNConvertFileInfo](#) objects, in the desired order, to convert and add to the output file. Each [PNConvertFileInfo](#) object describes a single input file to be converted and an optional collection of converter settings to use when converting the file. The [PNConvertFileInfo](#) *OutputPath* property is ignored, and the *OutputFolder* argument used instead.

Only the following converter settings are valid when combining files:

- [General Converter Options](#)
- [Endorsement Options](#)
- [Word Converter Options](#)
- [Excel Converter Options](#)
- [PowerPoint Converter Options](#)
- [Adobe Reader Options](#)
- [Internet Explorer Options](#)
- [Ghostscript Converter Options](#)
- [Image Converter Options](#)
- [OutsideIn AX Options](#)

IList<String> FileList

A list of strings, in the desired order, representing the full paths of each file to convert and add to the output file. The files can be on the local computer, on a shared location using a mapped drive letter or by passing a UNC formatted file path.

String OutputFolder

Full path to the save file location. This folder must be specified and it must be created before the call to `CombineFiles` is made. If the path doesn't exist or a file of the same name already exists in the output folder location, the conversion will fail. Pass `True` for *OverwriteExisting* to allow file overwriting.

String OutputName

The name to use for the output file, without extension. The default file extension for the type of multipaged file being created will always be added to the name provided here. This argument must be provided.

String OverwriteExisting

Set to **True** to overwrite existing files, or **False** to fail conversion when a file of the same name already exists in the save location.

String CreateResultsLogFile

Pass **True** to create a results log file containing a complete snapshot of the conversion information for each file. This file is saved with each output file. The name of the results log file is based on the name of the original file and also indicates the conversion status for that file. For example, when converting *Sample.doc*, a successful conversion will create *Sample.doc.succeeded.dcsresults* and if the conversion did not succeed, the file would be named *Sample.doc.failed.dcsresults*.

These log files can later be read from disk using the [DeserializeFromXML](#) method of the [PNConversionItem](#) class.

String SettingsProfile

The name of the profile to use, with or without the XML extension. Settings in the profile that do not apply to the type of output being created are ignored. Document Conversion Service includes several sample profiles for common types of output files for your use, or you can create your own and pass in a full path to your custom profile. See [Creating and Customizing Profiles](#) for a list of included profiles and how to create your own.

IDictionary<String, String> SettingsList

A dictionary of name/value pairs of settings that describes the conversion options. Used instead of *SettingsProfile* above. The name/value pairs that make up this dictionary are the same settings that are used to create the XML-formatted profiles included with Document Conversion Service. See [Conversion Settings](#) for a list of all of the settings that are available.

String ExtensionsProfile

Name of the file mapping profile XML file, with or without the XML extension. Providing this parameter is optional and an internal default mapping is provided. You would only need to provide this file if you wanted to override the default file extension to converter mappings provided.

String MimeProfile

Reserved for future use - pass String.Empty.

IDictionary<String, String> UserSettings

Optional. Pass a dictionary of additional conversion settings. These settings will override any matching settings passed in for *SettingsProfile* or *SettingsList*. Pass *null* if not using.

String RemoteComputerName

Optional. Pass *String.Empty* if you are converting locally or the name of the remote computer where Document Conversion Service is running. When converting remotely, a *ConversionWorkingFolder* must also be provided.

String ConversionWorkingFolder

Used to provide a shared path to be used when doing remote conversion or an alternate temporary working instead of our default of the Windows TEMP folder.

This setting is required when *RemoteComputerName* is provided for remote conversion (DCOM) as both the local and the remote computer need access to a shared path in which to do the conversion. Pass *String.Empty* if you are not using this setting.

When not doing remote conversion, this setting is not required in most cases but can be useful when dealing with folder and file names longer than 255 characters. When converting a file, the conversion tool copies the file and performs the conversion in temporary staging and working folders created on demand in the default Windows temp folder. These folders need to be less than 255 characters as required by the underlying programs used by Document Conversion Service to perform conversions. When dealing with these long path and file names the default folders created can occasionally cause path names that are too long for Document Conversion Service to process. When this happens this switch can be used to set the temporary folder to a shorter path to allow processing. Again, pass *String.Empty* if you are not using this setting.

String ConvertFileProcessLoggingPath

Optional. Specify a path to a folder in which to store the SmartInspect logs files of any failed conversions. These files are stored in the temp folder by default and can be viewed using the SmartInspect Redistributable Console. These log files are a tracing of the entire conversion process and are not the same as the conversion results log files created when a conversion fails. See [Controlling the SmartInspect Logging Files](#) to change where these files are stored, how they are named, or to disable creation of these files.

Remarks

In the case of a failed combine, the combine results log file is always created. When the combine does not succeed, a *.failed* folder is created in the save folder location specified by *OutputFolder* argument and the results log files are stored there.

The name of the results log when the combine does not succeed will be similar to the following:

```
PNCombineFiles_2013_05_31_2_50_05_PM_3.failed.dcsresults
```

The bold text in the name will change for each file and is based on the date and time of the run and an internal counter. See [Controlling the Failed Results File Location](#) to store these files in a different location, disable the use of the date and time in the name, or to disable the creation of these file.

Exceptions

Exception	Condition
ArgumentException	An empty, or badly formatted profile was passed for <i>SettingsProfile</i> . An empty list was passed for <i>SettingsList</i> . An empty, or badly formatted profile was passed for <i>ExtensionsProfile</i> . An empty list was passed for <i>FileList</i> . An empty name was passed for <i>OutputName</i> .

	A name for <i>RemoteComputerName</i> was passed but no corresponding <i>ConversionWorkingFolder</i> specified.
FileNotFoundException	One of the input files in the <i>FileList</i> does not exist or cannot be accessed.
DirectoryNotFoundException	The output path for <i>OutputFolder</i> is specified but the path does not exist or is invalid. When <i>ConversionWorkingFolder</i> is specified but does not exist or is invalid.

See Also:

[ConvertFile](#) [ConvertFileList](#) [ConvertFolder](#) [CombineFolder](#) [IsConversionServiceRunning](#)

**Code Sample - C# - Combine both files into a multipage TIFF image**

```
PNCombineItem resultItem = null;
IList<String> filesToTIFF = new List<String>();

filesToTIFF.Add(@"C:\Test\File1.pdf");
filesToTIFF.Add(@"C:\Test\File2.pdf");

resultItem = PNConverter.CombineFiles(filesToTIFF,
    @"C:\Test\Output\",
    @"CombinedPDF",
    true, // overwrite existing
    false, // do not create log
    "TIFF 200dpi OptimizedColor",
    String.Empty,
    String.Empty,
    null, // no custom user settings
    String.Empty, // not using DCOM
    String.Empty, // use default working folder
    String.Empty); // do not use custom log folder
```

**Code Sample - VB.NET - Combine both files into a multipage TIFF image**

```
Dim resultItem As PNCombineItem
Dim filesToTIFF As IList(Of String)

resultItem = Nothing

filesToTIFF.Add("C:\Test\File1.pdf")
filesToTIFF.Add("C:\Test\File2.pdf")

resultItem = PNConverter.CombineFiles(filesToTIFF, _
    "C:\Test\Output\", _
    "CombinedPDF", _
    True, _
    False, _
    "TIFF 200dpi OptimizedColor", _
    String.Empty, _
    String.Empty, _
    Nothing, _
    String.Empty, _
    String.Empty, _
    String.Empty)
```

CombineFolder

Description

Static method.

Converts and combines all files in the folder, and optionally all subfolders, using the requested conversion settings.

The order of the files in the combined file cannot be guaranteed and is dependent on the file system. In most cases they are alphabetical but can also be by creation time. Files from the root of the input folder are listed first, then all files from the subfolders when enabled. Subfolders are listed in alphabetical or creation time order, again dependent on the file system.

A filter pattern can be used to only process files in the folder that match the provided pattern, such as *.doc to process all Word documents, or ABC* to process all files that start with the letters ABC. An exclude filter is also provided, to allow you to skip files that match the exclude pattern. The exclude filter is applied to the list of files returned by the include filter.

The conversion settings passed in determine how the files are combined. For instance, passing conversion settings to create a multipaged PDF file will combine all input files into a single, multipage PDF file, while passing in the conversion settings to create serialized TIFF images will result in a serialized sequence of TIFF images, one for each page of each file.

Syntax

```
PNConverter.CombineFolder(InputFolder, IncludeSubFolders,
                           FileFilter, ExcludeFileFilter,
                           OutputFolder, OutputName, OverwriteExisting,
                           CreateResultsLogFiles, SettingsProfile,
                           ExtensionsProfile, MIMEProfile, UserSettings,
                           RemoteComputerName, ConversionWorkingFolder,
                           CombineFilesProcessLoggingPath)
```

```
PNConverter.CombineFolder(InputFolder, IncludeSubFolders,
                           FileFilter, ExcludeFileFilter,
                           OutputFolder, OutputName, OverwriteExisting,
                           CreateResultsLogFiles, SettingsProfile,
                           ExtensionsProfile, MIMEProfile, UserSettings,
                           RemoteComputerName, ConversionWorkingFolder,
                           CombineFilesProcessLoggingPath,
                           SortMode, SortOrder)
```

```
PNConverter.CombineFolder(InputFolder, IncludeSubFolders,
                           FileFilter, ExcludeFileFilter,
                           OutputFolder, OutputName, OverwriteExisting,
                           CreateResultsLogFiles, SettingsList,
                           ExtensionsProfile, MIMEProfile, UserSettings,
                           RemoteComputerName, ConversionWorkingFolder,
                           CombineFilesProcessLoggingPath)
```

```
PNConverter.CombineFolder(InputFolder, IncludeSubFolders,
```

```

FileFilter, ExcludeFileFilter,
OutputFolder, OutputName, OverwriteExisting,
CreateResultsLogFiles, SettingsList,
ExtensionsProfile, MIMEProfile, UserSettings,
RemoteComputerName, ConversionWorkingFolder,
CombineFilesProcessLoggingPath,
SortMode, SortOrder)

```

Returns a [PNCombineItem](#) object which contains a collection of [PNConversionResult](#) objects, one for each file in the folder (and subfolders, if selected) that matched the filter pattern. The [PNCombineItem](#) object contains a list of files used in the combine process, and a list of the resulting combined files as well as other information about the original combine request. Each inner [PNConversionResult](#) object contains information about the conversion results for a file in the combine set passed.

Parameters

String InputFolder

The full path to the folder containing the files to convert and combine together. This can be on the local computer, on a shared location using a mapped drive letter or by passing a UNC formatted file path.

String IncludeSubFolders

Set to **True** to include the subfolders under the folder when building the list of files to be converted and combined.

String FileFilter

A filter to process only the files matching the filter pattern, such as using *.pdf to only process files ending with the .PDF or .pdf extension. Multiple filters can be combined using the pipe (|) character, such as *.doc|*.pdf to process only Word and PDF files.

Hidden and system files are ignored, and the search pattern filters files based on a regular expression match of the long name of a file. The filter defaults to all files in the folder (*.*) if *String.Empty* or *null* are passed for the filter.

String ExcludeFileFilter

After the *Filter* pattern is used to get the list of files to convert from the *InputFolder*, the exclude filter can then be applied to that list to remove files that match the exclude pattern. Multiple excluded filters are combined using the pipe (|) character, such as *.pdf|*.xml to process all files returned except PDF and XML files.

If *String.Empty* or *null* is passed then no files are excluded.

String OutputFolder

Full path to the save file location. This folder must be specified and it must be created before the call to *CombineFolder* is made. If the path doesn't exist or a file of the same name already exists in the output folder location, the conversion will fail. Pass **True** for *OverwriteExisting* to allow file overwriting.

String OutputName

The name to use for the output file, without extension. The default file extension for the type of multipaged file being created will always be added to the name provided here. This argument must be provided.

String OverwriteExisting

Set to **True** to overwrite existing files, or **False** to fail conversion when a file of the same name already exists in the save location.

String CreateResultsLogFile

Pass **True** to create a results log file containing a complete snapshot of the conversion information for each file. This file is saved with each output file. The name of the results log file is based on the name of the original file and also indicates the conversion status for that file. For example, when converting *Sample.doc*, a successful conversion will create *Sample.doc.succeeded.dcsresults* and if the conversion did not succeed, the file would be named *Sample.doc.failed.dcsresults*.

These log files can later be read from disk using the [DeserializeFromXML](#) method of the [PNConversionItem](#) class.

String SettingsProfile

The name of the profile to use, with or without the XML extension. Settings in the profile that do not apply to the type of output being created are ignored. Document Conversion Service includes several sample profiles for common types of output files for your use, or you can create your own and pass in a full path to your custom profile. See [Creating and Customizing Profiles](#) for a list of included profiles and how to create your own.

IDictionary<String, String> SettingsList

A dictionary of name/value pairs of settings that describes the conversion options. Used instead of *SettingsProfile* above. The name/value pairs that make up this dictionary are the same settings that are used to create the XML-formatted profiles included with Document Conversion Service. See [Conversion Settings](#) for a list of all of the settings that are available.

String ExtensionsProfile

Name of the file mapping profile XML file, with or without the XML extension. Providing this parameter is optional and an internal default mapping is provided. You would only need to provide this file if you wanted to override the default file extension to converter mappings provided.

String MimeProfile

Reserved for future use - pass String.Empty.

IDictionary<String, String> UserSettings

Optional. Pass a dictionary of additional conversion settings. These settings will override any matching settings passed in for *SettingsProfile* or *SettingsList*. Pass *null* if not using.

String RemoteComputerName

Optional. Pass *String.Empty* if you are converting locally or the name of the remote computer where Document Conversion Service is running. When converting remotely, a *ConversionWorkingFolder* must also be provided.

String ConversionWorkingFolder

Used to provide a shared path to be used when doing remote conversion or an alternate temporary working instead of our default of the Windows TEMP folder.

This setting is required when *RemoteComputerName* is provided for remote conversion (DCOM) as both the local and the remote computer need access to a shared path in which to do the conversion. Pass *String.Empty* if you are not using this setting.

When not doing remote conversion, this setting is not required in most cases but can be useful when dealing with folder and file names longer than 255 characters. When converting a file, the conversion tool copies the file and performs the conversion in temporary staging and working folders created on demand in the default Windows temp folder. These folders need to be less

than 255 characters as required by the underlying programs used by Document Conversion Service to perform conversions. When dealing with these long path and file names the default folders created can occasionally cause path names that are too long for Document Conversion Service to process. When this happens this switch can be used to set the temporary folder to a shorter path to allow processing. Again, pass *String.Empty* if you are not using this setting.

String ConvertFileProcessLoggingPath

Optional. Specify a path to a folder in which to store the SmartInspect logs files of any failed conversions. These files are stored in the temp folder by default and can be viewed using the SmartInspect Redistributable Console. These log files are a tracing of the entire conversion process and are not the same as the conversion results log files created when a conversion fails. See [Controlling the SmartInspect Logging Files](#) to change where these files are stored, how they are named, or to disable creation of these files.

[PNFileSortMode](#) *SortMode*

Optional, controls the sort order of the list of files returned from the *InputFolder*. Files can be sorted by name, date created or date modified. Default is *None* when not specified.

[PNFileSortOrder](#) *SortOrder*

Optional, returns the files in *Ascending* (0-9, A-Z) or *Descending* (Z-A, 9-0) order. Default is *Ascending* when not specified.

Remarks

In the case of a failed combine, the combine results log file is always created. When the combine does not succeed, a *.failed* folder is created in the save folder location specified by *OutputFolder* argument and the results log files are stored there.

The name of the results log when the combine does not succeed will be similar to the following:

```
PNCombineFolder_2013_05_31_2_50_05_PM_3.failed.dcsresults
```

The bold text in the name will change for each file and is based on the date and time of the run and an internal counter. See [Controlling the Failed Results File Location](#) to store these files in a different location, disable the use of the date and time in the name, or to disable the creation of these file.

Exceptions

Exception	Condition
ArgumentException	An empty, or badly formatted profile was passed for <i>SettingsProfile</i> . An empty list was passed for <i>SettingsList</i> . An empty, or badly formatted profile was passed for <i>ExtensionsProfile</i> . The folder did not contain any files to process, or the filtered list of files returned an empty list. An empty name was passed for <i>OutputName</i> . A name for <i>RemoteComputerName</i> was passed but no corresponding <i>ConversionWorkingFolder</i> specified.
FileNotFoundException	One of the files found to process was removed or cannot be accessed when conversion was attempted.
DirectoryNotFoundException	The output path for <i>OutputFolder</i> is specified but the path does not exist or is invalid.

	When <i>ConversionWorkingFolder</i> is specified but does not exist or is invalid.
--	--

See Also:

[ConvertFile](#) [ConvertFileList](#) [ConvertFolder](#) [CombineFiles](#) [IsConversionServiceRunning](#)



Code Sample - C# - Combine files in C:\Input to multipage PDF document

```
PNCombineItem resultItem = null;

resultItem = PNConverter.CombineFolder(@"C:\Test\Input\",
    false, // do not include subfolders
    @"*.*", // process all files
    String.Empty, // do not exclude any files
    @"C:\Test\Output\",
    @"CombinedPDF",
    true, // overwrite existing
    false, // do not create log
    "PDF 200dpi OptimizedColor",
    String.Empty,
    String.Empty,
    null, // no custom user settings
    String.Empty, // not using DCOM
    String.Empty, // use default working folder
    String.Empty, // do not use custom log folder
    PNFileSortMode.DateCreated,
    PNFileSortOrder.Ascending);
```



Code Sample - VB.NET - Combine files in C:\Input to multipage PDF document

```
Dim resultItem As PNCombineItem
resultItem = Nothing

resultItem = PNConverter.ConvertFolder("C:\Test\Input", _
    False, _
    "*.*", _
    String.Empty, _
    "C:\Test\Output", _
    "CombinedPDF", _
    True, _
    False, _
    "PDF 200dpi OptimizedColor", _
    String.Empty, _
    String.Empty, _
    Nothing, _
    String.Empty, _
    String.Empty, _
    String.Empty, _
    )
```

IsConversionServiceRunning

Description

Test if Document Conversion Service is running and ready to convert.

Syntax

```
PNConverter.IsConversionServiceRunning(ComputerName)
```

Returns **True** if Document Conversion Service is running and ready to convert a file, **False** otherwise.

Parameters

String ComputerName

If you are running Document Conversion Service locally, pass *String.Empty* to test if the service is running. If Document Conversion Service is running on a remote computer, pass the name of that computer to test the state of the conversion service on that computer.

See Also:



[ConvertFile](#) [ConvertFileList](#) [ConvertFolder](#) [CombineFiles](#)

PNConvertFileInfo




Description

The PNConvertFileInfo class describes a single input file to be converted, the output path for that file and an optional collection of settings to use when converting the file. It is used to pass collections of files to the [ConvertFileList](#) method to be converted.

Methods

 PNConvertFileInfo	Initializes a new instance of the PNConvertFileInfo object.
 AddSetting	Adds a setting to a PNConvertFileInfo object.

Properties

 InputFile	Gets or sets the full path to the input file to be converted.
 OutputPath	Gets or sets the full path to the output folder in which to save the new file.
 Settings	<i>Optional.</i> A collection of conversion settings that will apply only to this file.

Methods

AddSetting

Description

Add a [PNSetting](#) object into the IList collection of PNSetting objects.. You can add any number of settings into the collection. If the same setting is added more than once, the last setting in the collection is the one that will be used.

Syntax

```
expression.AddSetting(setting)
```

where *expression* is a [PNConvertFileInfo](#) object.

Parameters

PNSetting setting

The setting to add.

See Also:

[PNConvertFileInfo](#) [PNSetting](#)

PNConvertFileInfo

Description

Initializes an instance of the [PNConvertFileInfo](#) object with an input file, the desired output folder and an optional collection of conversion settings to use when converting the input file. This class is used to pass collections of files to the [ConvertFileList](#) method to be converted.

Syntax

```
PNConvertFileInfo(inputFile,outputPath)
```

```
PNConvertFileInfo(inputFile, outputPath, settings)
```

Parameters

String inputFile

The full path to the input file to be converted. This can be on the local computer, on a shared location using a mapped drive letter or by passing a UNC formatted file path.

String outputPath

Full path to the save file location, or *String.Empty* to create the new file in the same location as *inputFile*.

This folder must be created before the call to [ConvertFileList](#) is made. If the path doesn't exist or a file of the same name already exists in the save file location, the conversion will fail. Pass **True** for *OverwriteExisting* to allow file overwriting.

IList<[PNSetting](#)> settings

A collection of conversion settings to use when converting the file. These conversion settings will apply only to *inputFile*.

See Also:

[AddSetting](#)

Properties

InputFile

Description

Gets or sets the full path to the input file to be converted. This can be on the local computer, on a shared location using a mapped drive letter or by passing a UNC formatted file path.

Syntax

`expression.InputFile`

where *expression* is a [PNConvertFileInfo](#) object.

Returns a **String**.

See Also:

[OutputPath Settings](#)

OutputPath

Description

Gets or sets the full path to the output folder in which to save the converted file. This can be on the local computer, on a shared location using a mapped drive letter or by passing a UNC formatted file path.

Syntax

`expression.OutputPath`

where *expression* is a [PNConvertFileInfo](#) object.

Returns a **String**.

See Also:

[InputFile Settings](#)

Settings

Description

An list of conversion options that will apply only to this file. This collection is optional and can be empty or null.

Syntax

`expression.Settings`

where *expression* is a [PNConvertFileInfo](#) object.

Returns an **ICollection<[PNSetting](#)>** collection.

See Also:

[InputFile OutputPath](#)





PNConversionItem

Description




The PNConversionItem class contains information about the original conversion request and the results of the conversion in an inner [PNConversionResult](#) property. This class is used by [ConvertFile](#), [ConvertFileList](#) and [ConvertFolder](#) to return the results of document conversion.

This is also the class that is serialized to disk to create the results log files that can optionally be created by the ConvertFile, ConvertFileList and ConvertFolder methods. Several static methods for extracting information from the results log files on disk are provided.




Static Methods








 DeserializeFromXML	Deserialize the conversion results from a PNConversionItem serialized to disk as XML.
 GetCreatedFiles	Returns a list of the files created from a PNConversionItem serialized to disk as XML.
 GetErrors	Returns a list of the errors from a PNConversionItem serialized to disk as XML.
 GetSourceFileName	Returns the source file used from a PNConversionItem serialized to disk as XML.

Methods

 GetConversionStatus	Returns the conversion status as one of PNConvertResultStatus strings.
 HasErrors	Returns True if errors occurred during the conversion, False otherwise.
 SerializeToXML	Serialize the conversion results to a file on disk.

Properties

 ConversionResult	Read-only; A PNConvertResultStatus string enumeration of the conversion status.
 ConversionLogFilePath	Read-only; the path to the logging file for this conversion item.
 ConversionResultsFilePath	Read-only; the path to the .dcsresults file for this conversion item.

 ConverterPlugInList	Read-only; The list of converters that Document Conversion Service chose from to convert the file.
 OutputBaseName	Read-only; The base name used to name the converted files.
 OutputDirectory	Read-only; The directory in which the converted files were created.
 Settings	Read-only; A List< PNSetting > collection of the conversion settings used to create the output files.
 SourceFileExtension	Read-only; The extension of the source file that was used to determine what converter Document Conversion Service used to convert the file.
 SourceFileMimeType	<i>Reserved for future use.</i>
 SourceFilePath	Read-only; The source file that was converted.

Methods

DeserializeFromXML

Description

Static method.

Deserializes a PNConversionItem serialized to disk as XML.

The file passed can be a results log file ending in the *.dcsresults* extension created by enabling the results log file option when calling [ConvertFile](#), [ConvertFileList](#) or [ConvertFolder](#), or a file on disk created by calling [SerializeToXML](#).

Syntax

```
PNConversionItem.DeserializeFromXML(FilePath)
```

Returns a [PNConversionItem](#) object.

Parameters

String FilePath

Full path to the file.

See Also:

[GetCreatedFiles](#) [GetErrors](#) [GetSourceFileName](#)

GetConversionStatus

Description

Returns the conversion status.

Syntax

expression.GetConversionStatus(path)

where *expression* is a [PNConversionItem](#) object.

Returns conversion status as a [PNConvertResultStatus](#).

See Also:

[HasErrors](#) [SerializeToXML](#)

GetCreatedFiles

Description

Static method.

Given a path to a PNConversionItem serialized to disk as XML, returns a list of the files created. This list can be empty if no files were created.

The file passed can be a results log file ending in the *.dcsresults* extension created by enabling the results log file option when calling [ConvertFile](#), [ConvertFileList](#) or [ConvertFolder](#), or a file on disk created by calling [SerializeToXML](#).

Syntax

PNConversionItem.GetCreatedFiles(path)

Returns **List<String>** of the paths to the created files.

Parameters

String path

Full path to the file.

See Also:

[DeserializeFromXML](#) [GetErrors](#) [GetSourceFileName](#)

GetErrors

Description

Static method.

Given a path to a PNConversionItem serialized to disk as XML, returns a list of any errors encountered during conversion. This list can be empty if no errors occurred.

The file passed can be a results log file ending in the *.dcsresults* extension created by enabling the results log file option when calling [ConvertFile](#), [ConvertFileList](#) or [ConvertFolder](#), or a file on disk created by calling [SerializeToXML](#).

Syntax

```
PNConversionItem.GetErrors(path)
```

Returns **List<String>** of error messages.

Parameters

String path

Full path to the file.

See Also:

[DeserializeFromXML](#) [GetCreatedFiles](#) [GetSourceFileName](#)

GetSourceFileName

Description

Static method.

Given a path to a PNConversionItem serialized to disk as XML, returns the name of the file that was converted.

The file passed can be a results log file ending in the *.dcsresults* extension created by enabling the results log file option when calling [ConvertFile](#), [ConvertFileList](#) or [ConvertFolder](#), or a file on disk created by calling [SerializeToXML](#).

Syntax

```
PNConversionItem.GetSourceFileName(path)
```

Returns a **String**.

Parameters

String path

Full path to the file.

See Also:

[DeserializeFromXML](#) [GetCreatedFiles](#) [GetSourceFileName](#)

HasErrors

Description

Returns **True** if errors occurred during the conversion, **False** otherwise.

Syntax

```
expression.HasErrors()
```

where *expression* is a [PNConversionItem](#) object.

Returns a **Boolean**.

See Also:

[GetConversionStatus](#) [SerializeToXML](#)

SerializeToXML

Description

Serializes the PNConversionItem to an XML file on disk.

Syntax

```
expression.SerializeToXML(FilePath)
```

where *expression* is a [PNConversionItem](#) object.

Parameters

String FilePath

Full path to the file to create, including the filename.

See Also:

[GetConversionStatus](#) [HasErrors](#)

Properties

ConversionResult

Description

Gets the [PNConversionResult](#) object describing the results of the conversion.

Read-only.

Syntax

expression.ConversionResult

where *expression* is a [PNConversionItem](#) object.

Returns [PNConversionResult](#).

See Also:

[ConversionLogFilePath](#) [ConversionResultsFilePath](#) [ConverterPlugInList](#)
[OutputBaseName](#) [OutputDirectory](#) [Settings](#)
[SourceFileExtension](#) [SourceFileMimeType](#) [SourceFilePath](#)

ConversionLogFilePath

Description

The path to the Smart Inspect console logging file (*.sil). This file is always created when a conversion runs. If the conversion is successful, the log file is normally deleted. If it fails, it is kept and copied to the Windows temp folder. The [General Converter Options](#) variables *KeepFailedProcessingLoggingFiles* and *AlwaysKeepProcessingLoggingFiles* allow you to control if this file is always kept or always deleted. See [Controlling the SmartInspect Logging Files](#) to change where these files are stored, how they are named, or to disable creation of these files.

Read-only.

Syntax

expression.ConversionLogFilePath

where *expression* is a [PNConversionItem](#) object.

Returns **String**.

See Also:

[ConversionResult](#) [ConversionResultsFilePath](#) [ConverterPlugInList](#)
[OutputBaseName](#) [OutputDirectory](#) [Settings](#)
[SourceFileExtension](#) [SourceFileMimeType](#) [SourceFilePath](#)

ConversionResultsFilePath

Description

The path to the results file (*.dcsresults) that is created when a conversion fails. The [General Converter Options](#) variables *KeepFailedItemResultsFiles* control if this file is kept for failed items. See [Controlling the Failed Results File Location](#) to change where these files are stored, how they are named, or to disable creation of these files.

Read-only.

Syntax

expression.ConversionResultsFilePath

where *expression* is a [PNConversionItem](#) object.

Returns **String**.

See Also:

[ConversionResult](#) [ConversionLogFilePath](#) [ConverterPlugInList](#)
[OutputBaseName](#) [OutputDirectory](#) [Settings](#)
[SourceFileExtension](#) [SourceFileMimeType](#) [SourceFilePath](#)

ConverterPlugInList

Description

The list of converters that Document Conversion Service chose from to convert the file. This can be a single converter, or as some file types can be converted using more than one converter, it can be a list of converters.

Read-only.

Syntax

expression.ConverterPlugInList

where *expression* is a [PNConversionItem](#) object.

Returns **String**.

See Also:

[ConversionResult](#) [ConversionLogFilePath](#) [ConversionResultsFilePath](#)
[OutputBaseName](#) [OutputDirectory](#) [Settings](#)
[SourceFileExtension](#) [SourceFileMimeType](#) [SourceFilePath](#)

OutputBaseName

Description

The base name used to name the output files.

Read-only.

Syntax

expression.OutputBaseName

where *expression* is a [PNConversionItem](#) object.

Returns **String**.

See Also:

[ConversionResult](#) [ConversionLogFilePath](#) [ConversionResultsFilePath](#)
[ConverterPlugInList](#) [OutputDirectory](#) [Settings](#)
[SourceFileExtension](#) [SourceFileMimeType](#) [SourceFilePath](#)

OutputDirectory

Description

Gets the directory in which the converted files were created. This can be an empty string if no output directory was specified.

Read-only.

Syntax

expression.OutputDirectory

where *expression* is a [PNConversionItem](#) object.

Returns **String**.

See Also:

[ConversionResult](#) [ConversionLogFilePath](#) [ConversionResultsFilePath](#)
[ConverterPlugInList](#) [OutputBaseName](#) [Settings](#)
[SourceFileExtension](#) [SourceFileMimeType](#) [SourceFilePath](#)

Settings

Description

A collection of the conversion settings used to create the output files.

Read-only.

Syntax

`expression.Settings`

where *expression* is a [PNConversionItem](#) object.

Returns an **List<[PNSetting](#)>** collection.

See Also:

[ConversionResult](#) [ConversionLogFilePath](#) [ConversionResultsFilePath](#)
[ConverterPlugInList](#) [OutputBaseName](#) [OutputDirectory](#)
[SourceFileExtension](#) [SourceFileMimeType](#) [SourceFilePath](#)

SourceFileExtension

Description

Gets the extension of the source file that was used to determine what converter Document Conversion Service used to convert the file.

Read-only.

Syntax

`expression.SourceFileExtension`

where *expression* is a [PNConversionItem](#) object.

Returns a **String**.

See Also:

[ConversionResult](#) [ConversionLogFilePath](#) [ConversionResultsFilePath](#)
[ConverterPlugInList](#) [OutputBaseName](#) [OutputDirectory](#)
[Settings](#) [SourceFileMimeType](#) [SourceFilePath](#)

SourceFileMimeType

Description

Reserved for future use.

SourceFilePath

Description

The full path to the source file that was converted.

Read-only.

Syntax

expression.SourceFilePath

where *expression* is a [PNConversionItem](#) object.

Returns a **String**.

See Also:

[ConversionResult](#) [ConversionLogFilePath](#) [ConversionResultsFilePath](#)

[ConverterPlugInList](#) [OutputBaseName](#) [OutputDirectory](#)

[Settings](#) [SourceFileExtension](#) [SourceFileMimeType](#)

PNCombineItem





Description

The PNCombineItem class contains information about the original file combine (append) request, a list of the output files created, and an inner list of [PNConversionResult](#) items for each file included as part of the combine operation.



This class is used by the [CombineFiles](#) method to return the results of document conversion and combination.

This is also the class that is serialized to disk to create the results log files that can optionally be created by the CombineFiles method. Several static methods for extracting information from the results log files on disk are provided.




Static Methods







 DeserializeFromXML	Deserialize the conversion results from a PNCombineItem serialized to disk as XML.
 GetCreatedFiles	Given a PNCombineItem serialized to disk as XML, returns a list of the files created in the results.
 GetErrors	Given a PNCombineItem serialized to disk as XML, returns a list of any errors in the results.
 GetInputFileNames	Given a PNCombineItem serialized to disk as XML, returns the list of source files passed to be combined together.

Methods

 HasErrors	Returns True if errors occurred during the conversion, False otherwise.
 SerializeToXML	Serialize the file combine results to a file on disk.

Properties

 CombinedOutputFileList	Read-only; The list of files created; this can be one or more depending on the output format chosen.
 ConversionItems	Read-only; The list of PNConversionResult items for each file in the combine set.
 ConversionLogFilePath	Read-only; the path to the logging file for this combination item.

 ConversionResultsFilePath	Read-only; the path to the .dcsresults file for this combination item.
 Errors	Read-only; A collection of any errors that occurred during the convert and combine process.
 InputFiles	Read-only; The collection of source files used as input into the combine call.
 OutputBaseName	Read-only; The base name used to name the combined file or files.
 OutputDirectory	Read-only; The directory in which the combined file or files were created.
 Settings	Read-only; A List< PNSetting > collection of the conversion settings used to create the output files.

Methods

DeserializeFromXML

Description

Static method.

Deserializes a PNCombineItem object that was serialized to disk as XML.

The XML file passed in must be a results log file ending in the .dcsresults extension created by enabling the results log file option when calling [CombineFiles](#), or a file on disk created by calling [SerializeToXML](#).

Syntax

```
PNCombineItem.DeserializeFromXML(FilePath)
```

Returns a [PNCombineItem](#) object.

Parameters

String FilePath

Full path to the file.

See Also:

[GetCreatedFiles](#) [GetErrors](#) [GetInputFileNames](#)

GetCreatedFiles

Description

Static method.

Given a path to a PNCombineItem serialized to disk as XML, returns a list of the files created. This list can be empty if no files were combined.

The XML file passed in must be a results log file ending in the *.dcsresults* extension created by enabling the results log file option when calling [CombineFiles](#), or a file on disk created by calling [SerializeToXML](#).

Syntax

```
PNCombineItem.GetCreatedFiles(path)
```

Returns **List<String>** of the paths to the created files.

Parameters

String path

Full path to the file.

See Also:

[DeserializeFromXML](#) [GetErrors](#) [GetInputFileNames](#)

GetErrors

Description

Static method.

Given a path to a PNConversionItem, serialized to disk as XML, returns a list of any errors encountered during conversion. This list can be empty if no errors occurred.

The XML file passed in must be a results log file ending in the *.dcsresults* extension created by enabling the results log file option when calling [CombineFiles](#), or a file on disk created by calling [SerializeToXML](#).

Syntax

```
PNConversionItem.GetErrors(path)
```

Returns **List<String>** of error messages.

Parameters

String path

Full path to the file.

See Also:

[DeserializeFromXML](#) [GetCreatedFiles](#) [GetInputFileNames](#)

GetInputFileNames

Description

Static method.

Given a path to a PNCombineItem serialized to disk as XML, returns a list of the source files that were passed to be combined together.

The XML file passed in must be a results log file ending in the *.dcsresults* extension created by enabling the results log file option when calling [CombineFiles](#), or a file on disk created by calling [SerializeToXML](#).

Syntax

```
PNCombineItem.GetSourceFileName(path)
```

Returns **List<String>** of the paths to the input files used for the combine.

Parameters

String path

Full path to the file.

See Also:

[DeserializeFromXML](#) [GetCreatedFiles](#) [GetErrors](#)

HasErrors

Description

Returns **True** if errors occurred during the conversion, **False** otherwise.

Syntax

```
expression.HasErrors()
```

where *expression* is a [PNCombineItem](#) object.

Returns a **Boolean**.

See Also:

[SerializeToXML](#)

SerializeToXML**Description**

Serializes the PNCombineltem to an XML file on disk.

Syntax

```
expression.SerializeToXML(FilePath)
```

where *expression* is a [PNCombineltem](#) object.

Parameters

String FilePath

Full path to the file to create, including the filename.

See Also:

[HasErrors](#)

Properties**CombinedOutputFileList****Description**

The list of the files created. This list can be empty if no files were combined.

Read-only.

Syntax

```
expression.CombinedOutpoutFileList
```

where *expression* is an [PNCombineltem](#) object.

Returns a **List<String>** collection.

See Also:

[ConversionItems](#) [ConversionLogFilePath](#) [ConversionResultsFilePath](#)
[Errors](#) [InputFiles](#) [OutputBaseName](#) [OutputDirectory](#) [Settings](#)

ConversionItems

Description

The collection of [PNConversionResult](#) objects, one for each of the files in the combine set. This list can be empty if no files were combined.

Read-only.

Syntax

`expression.ConversionItems`

where *expression* is an [PNCombineltem](#) object.

Returns a **List**<[PNConversionResult](#)> collection.

See Also:

[CombinedOutputFileList](#) [ConversionLogFilePath](#) [ConversionResultsFilePath](#)
[Errors](#) [InputFiles](#) [OutputBaseName](#) [OutputDirectory](#) [Settings](#)

ConversionLogFilePath

Description

The path to the Smart Inspect console logging file (*.sil). This file is always created when the convert and combine runs. If the convert and combine is successful, the log file is normally deleted. If it fails, it is kept and copied to the Windows temp folder. The [General Converter Options](#) variables *KeepFailedProcessingLoggingFiles* and *AlwaysKeepProcessingLoggingFiles* allow you to control if this file is always kept or always deleted. See [Controlling the SmartInspect Logging Files](#) to change where these files are stored, how they are named, or to disable creation of these files.

Read-only.

Syntax

`expression.ConversionLogFilePath`

where *expression* is an [PNCombineltem](#) object.

Returns **String**.

See Also:

[CombinedOutputFileList](#) [ConversionItems](#) [ConversionResultsFilePath](#)
[Errors](#) [InputFiles](#) [OutputBaseName](#) [OutputDirectory](#) [Settings](#)

ConversionResultsFilePath

Description

The path to the results file (*.dcsresults) that is created when a conversion fails. The [General Converter Options](#) variables *KeepFailedItemResultsFiles* control if this file is kept for failed items. See [Controlling the Failed Results File Location](#) to change where these files are stored, how they are named, or to disable creation of these files.

Read-only.

Syntax

`expression.ConversionResultsFilePath`

where *expression* is an [PNCombinelItem](#) object.

Returns **String**.

See Also:

[CombinedOutputFileList](#) [ConversionItems](#) [ConversionLogFilePath](#)
[Errors](#) [InputFiles](#) [OutputBaseName](#) [OutputDirectory](#) [Settings](#)

Errors

Description

A collection of any errors that occurred during conversion.

Read-only.

Syntax

`expression.Errors`

where *expression* is an [PNCombinelItem](#) object.

Returns a **List**<[PNConversionResultError](#)> collection.

See Also:

[CombinedOutputFileList](#) [ConversionItems](#) [ConversionLogFilePath](#) [ConversionResultsFilePath](#)
[InputFiles](#) [OutputBaseName](#) [OutputDirectory](#) [Settings](#)

InputFiles

returns a list of the source files that were passed to be combined together.

Description

A list of the input files passed in to be combined together.

Read-only.

Syntax

`expression.InputFiles`

where *expression* is an [PNCombineltem](#) object.

Returns a **List<String>** collection.

See Also:

[CombinedOutputFileList](#) [ConversionItems](#) [ConversionLogFilePath](#) [ConversionResultsFilePath](#)
[Errors](#) [OutputBaseName](#) [OutputDirectory](#) [Settings](#)

OutputBaseName

Description

The base name used to name the output files.

Read-only.

Syntax

`expression.OutputBaseName`

where *expression* is a [PNCombineltem](#) object.

Returns **String**.

See Also:

[CombinedOutputFileList](#) [ConversionItems](#) [ConversionLogFilePath](#) [ConversionResultsFilePath](#)
[Errors](#) [InputFiles](#) [OutputDirectory](#) [Settings](#)

OutputDirectory

Description

Gets the directory in which the combined file was created.

Read-only.

Syntax

`expression.OutputDirectory`

where *expression* is a [PNCombineltem](#) object.

Returns **String**.

See Also:

[CombinedOutputFileList](#) [ConversionItems](#) [ConversionLogFilePath](#) [ConversionResultsFilePath](#)
[Errors](#) [InputFiles](#) [OutputBaseName](#) [Settings](#)

Settings

Description

A collection of the conversion settings used to create the combined file.

Read-only.

Syntax

expression.Settings

where *expression* is a [PNCombineItem](#) object.

Returns an **List<[PNSetting](#)>** collection.

See Also:










[CombinedOutputFileList](#) [ConversionItems](#) [ConversionLogFilePath](#) [ConversionResultsFilePath](#)
[Errors](#) [InputFiles](#) [OutputBaseName](#) [OutputDirectory](#)

PNConversionResult

Description

The PNConversionResult class describes the results of the conversion. This class contains the list of files created, any informational messages and any error messages that occurred during conversion.

Properties

 Completed	Read-only; The time at which the conversion was completed.
 ConverterPlugInUsed	Read-only; The converter that was used by Document Conversion Service to convert the file.
 Errors	Read-only; A collection of any errors that occurred during conversion.
 Messages	Read-only; A collection of any informational messages returned.
 OutputFileRenderedPages	Read-only; A collection of information about each page created in the converted file.
 OutputFiles	Read-only; A list of the files created.
 PrintJobPrintedPages	Read-only; A collection of information about each page that was printed to create the converted file.
 PrintJobs	Read-only; A list of all print jobs that resulted from converting this file.
 Submitted	Read-only; The time at which the conversion request was submitted to Document Conversion Service.

Properties

Completed

Description

Returns the time this document conversion was completed.

Read-only.

Syntax

expression.Completed

where *expression* is an [PNConversionResult](#) object.

Returns a **DateTime** object.

See Also:

[ConverterPlugInUsed](#) [Errors](#) [Messages](#) [OutputFileRenderedPages](#)
[OutputFiles](#) [PrintJobPrintedPages](#) [PrintJobs](#) [Submitted](#)

ConverterPlugInUsed

Description

Returns the name of the converter that was used by Document Conversion Service to convert the file. This will be one of the converters listed in the [ConverterPlugInList](#) property of the [PNConversionItem](#) parent object.

Read-only.

Syntax

expression.ConverterPlugInUsed

where *expression* is an [PNConversionResult](#) object.

Returns a **String**.

See Also:

[Completed](#) [Errors](#) [Messages](#) [OutputFileRenderedPages](#)
[OutputFiles](#) [PrintJobPrintedPages](#) [PrintJobs](#) [Submitted](#)

Errors

Description

A collection of any errors that occurred during conversion.

Read-only.

Syntax

expression.Errors

where *expression* is an [PNConversionResult](#) object.

Returns a **List**<[PNConversionResultError](#)> collection.

See Also:

[Completed](#) [ConverterPlugInUsed](#) [Messages](#) [OutputFileRenderedPages](#)
[OutputFiles](#) [PrintJobPrintedPages](#) [PrintJobs](#) [Submitted](#)

Messages

Description

A collection of any informational messages returned.

Read-only.

Syntax

`expression.Messages`

where *expression* is an [PNConversionResult](#) object.

Returns a **List**<[PNConversionResultMessage](#)> collection.

See Also:

[Completed](#) [ConverterPluginUsed](#) [Errors](#) [OutputFileRenderedPages](#)
[OutputFiles](#) [PrintJobPrintedPages](#) [PrintJobs](#) [Submitted](#)

OutputFiles

Description

A collection of [PNConversionResultOutputFile](#) objects. There will be one object in the collection for every file created. The [PNConversionResultOutputFile](#) object contains the full path to the output file.

Read-only.

Syntax

`expression.OutputFiles`

where *expression* is an [PNConversionResult](#) object.

Returns a **List**<[PNConversionResultOutputFile](#)> collection.

See Also:

[Completed](#) [ConverterPluginUsed](#) [Errors](#) [Messages](#) [OutputFileRenderedPages](#)
[PrintJobPrintedPages](#) [PrintJobs](#) [Submitted](#)

PrintJobPrintedPages

Description

A collection of [PNConversionResultPrintJobPrintedPage](#) objects. This page object represents the print settings of the page when a converter from Document Conversion Service uses the Document Conversion Service to convert the file.

Read-only.

Syntax

`expression.PrintJobPrintedPages`

where *expression* is an [PNConversionResult](#) object.

Returns a **List**<[PNConversionResultPrintJobPrintedPage](#)> collection.

See Also:

[Completed](#) [ConverterPluginUsed](#) [Errors](#) [Messages](#)
[OutputFileRenderedPages](#) [OutputFiles](#) [PrintJobs](#) [Submitted](#)

OutputFileRenderedPages

Description

A collection of [PNConversionResultOutputFileRenderedPage](#) objects. A [PNConversionResultOutputFileRenderedPage](#) object contains information about each individual page for this converted file, including what file it was created in and what page it is in the new file.

Read-only.

Syntax

`expression.OutputFileRenderedPages`

where *expression* is an [PNConversionResult](#) object.

Returns a **List**<[PNConversionResultOutputFileRenderedPage](#)> collection

See Also:

[Completed](#) [ConverterPluginUsed](#) [Errors](#) [Messages](#)
[OutputFiles](#) [PrintJobPrintedPages](#) [PrintJobs](#) [Submitted](#)

PrintJobs

Description

A collection of [PNConversionResultPrintJob](#) objects. This print job object represents a single print job created when a converter from Document Conversion Service uses the Document Conversion Service to convert the file. Most documents will only create a single print job, but there are certain converters, such as Excel, that do create multiple print jobs for a single document.

Read-only.

Syntax

`expression.PrintJobs`

where *expression* is an [PNConversionResult](#) object.

Returns a **List**<[PNConversionResultPrintJob](#)> objects.

See Also:

[Completed](#) [ConverterPluginUsed](#) [Errors](#) [Messages](#) [OutputFileRenderedPages](#)
[OutputFiles](#) [PrintJobPrintedPages](#) [Submitted](#)

Submitted

Description

The time at which the conversion request was submitted to Document Conversion Service.
Read-only.

Syntax

expression.Submitted

where *expression* is an [PNConversionResult](#) object.

Returns a **DateTime** object.

See Also:


[Completed](#) [ConverterPluginUsed](#) [Errors](#) [Messages](#) [OutputFileRenderedPages](#)
[OutputFiles](#) [PrintJobPrintedPages](#) [PrintJobs](#)

PNConversionResultError

Description

The PNConversionResultError class wraps a single error message returned as part of a collection of errors in a [PNConversionResult](#) object.

Properties

 Value	Gets the error message.
---	-------------------------

Properties

Value

Description

Gets the error message.

Syntax

expression.Value

where *expression* is an [PNConversionResultError](#) object.


Returns a **String**.

PNConversionResultMessage

Description

The PNConversionResultMessage class wraps a single information message returned as part of a collection of messages in a [PNConversionResult](#) object.

Properties

 Value	Gets the informational message.
---	---------------------------------

Properties

Value

Description

Gets the informational message.

Syntax

expression.Value

where *expression* is an [PNConversionResultMessage](#) object.




Returns a **String**.

PNConversionResultOutputFile


Description

A `PNConversionResultOutputFile` object is created for every physical file created on disk. It contains the full output filename of the created file and three collections: a [PNConversionResultOutputFileRenderedPage](#) collection of pages representing each page in the file on disk, a [PNConversionResultPrintJobPrintedPage](#) collection of each printed page that was used to create the file, and a [PNConversionResultPrintJob](#) collection of print jobs that were used to create the output file.

Methods

 GetOutputFileRenderedPages	Read-only; Returns a collection of PNConversionResultOutputFileRenderedPage objects.
 GetPrintJobPrintedPages	Read-only; Returns a collection of PNConversionResultPrintJobPrintedPage objects.
 GetPrintJobs	Read-only; Returns a collection of PNConversionResultPrintJob objects.

Properties

 OutputFilePath	Read-only; The filename of the file created.
--	--

Methods

`GetOutputFileRenderedPages`

Description

Returns a collection of [PNConversionResultOutputFileRenderedPage](#) objects. A `PNConversionResultOutputFileRenderedPage` object contains information about each individual page for this converted file, including what file it was created in and what page number it is in the new file.

Read-only.

Syntax

`expression.GetOutputFileRenderedPages`

where `expression` is an [PNConversionResultOutputFile](#) object.

Returns a `List<PNConversionResultOutputFileRenderedPage>` collection.

See Also:

[GetPrintJobPrintedPages](#) [GetPrintJobs](#)

GetPrintJobPrintedPages

Description

Returns a collection of [PNConversionResultPrintJobPrintedPage](#) objects. This page object represents the print settings of the page when a converter from Document Conversion Service uses the Document Conversion Service to convert the file.

Read-only.

Syntax

expression.GetPrintJobPrintedPages

where *expression* is an [PNConversionResultOutputFile](#) object.

Returns a List<[PNConversionResultPrintJobPrintedPage](#)> collection.

See Also:

[GetOutputFileRenderedPages](#) [GetPrintJobs](#)

GetPrintJobs

Description

Returns a collection of [PNConversionResultPrintJob](#) objects. This print job object represents a single print job created when a converter from Document Conversion Service uses the Document Conversion Service to convert the file. Most documents will only create a single print job, but there are certain converters, such as Excel, that do create multiple print jobs for a single document.

Read-only.

Syntax

expression.GetPrintJobs

where *expression* is an [PNConversionResultOutputFile](#) object.

Returns a List<[PNConversionResultPrintJob](#)> collection.

See Also:

[GetOutputFileRenderedPages](#) [GetPrintJobPrintedPages](#)

Properties

OutputFilePath

Description

The name of the file created. This is the fully qualified path, including directory and filename.

Read-only.

Syntax

expression.OutputFilePath

where *expression* is an [PNConversionResultOutputFile](#) object

Returns a **String**.




PNConversionResultOutputFileRenderedPage

Description









A `PNConversionResultOutputFileRenderedPage` object represents a single page in the physical file on disk. From this object you can get the full output filename of the created file in which it is located and other information about this page such as orientation and page width and height.

There are also two collections: a [PNConversionResultPrintJobPrintedPage](#) collection of each printed page that was used to create the file, and a [PNConversionResultPrintJob](#) collection of print jobs that were used to create the output file.

Methods

 GetOutputFile	Read-only; Returns a PNConversionResultOutputFile object.
 GetPrintJobPrintedPages	Read-only; Returns a collection of PNConversionResultPrintJobPrintedPage objects.
 GetPrintJobs	Read-only; Returns a collection of PNConversionResultPrintJob objects.

Properties

 BitsPerPixel	Read-only; The bits per pixel, or color depth of the page on disk.
 HeightInPixels	Read-only; The height of the page in pixels.
 Orientation	Read-only; The orientation of the page, either <i>Portrait</i> or <i>Landscape</i> .
 PageNumber	Read-only; The page number of the page in the file on disk.
 RotationInDegrees	Read-only; The rotation of the page in the file on disk.
 WidthInPixels	Read-only; The weight of the printed page in pixels.
 XPixelsPerInch	Read-only; The vertical dots per inch, or resolution, of the page.
 YPixelsPerInch	Read-only; The horizontal dots per inch, or resolution, of the page.

Methods

GetOutputFile

Description

Returns a [PNConversionResultOutputFile](#) object. From this object you can get the full output filename of the created file.

Read-only.

Syntax

expression.GetOutputFile

where *expression* is an [PNConversionResultOutputFileRenderedPage](#) object.

Returns a [PNConversionResultOutputFile](#) object.

See Also:

[GetPrintJobPrintedPages](#) [GetPrintJobs](#)

GetPrintJobPrintedPages

Description

Returns a collection of [PNConversionResultPrintJobPrintedPage](#) objects. This page object represents the print settings of the page when a converter from Document Conversion Service uses the Document Conversion Service to convert the file.

Read-only.

Syntax

expression.GetPrintJobPrintedPages

where *expression* is an [PNConversionResultOutputFileRenderedPage](#) object.

Returns a **List**<[PNConversionResultPrintJobPrintedPage](#)> collection.

See Also:

[GetOutputFile](#) [GetPrintJobs](#)

GetPrintJobs

Description

Returns a collection of [PNConversionResultPrintJob](#) objects. This print job object represents a single print job created when a converter from Document Conversion Service uses the Document

Conversion Service to convert the file. Most documents will only create a single print job, but there are certain converters, such as Excel, that do create multiple print jobs for a single document.

Read-only.

Syntax

expression.GetPrintJobs

where *expression* is an [PNConversionResultOutputFileRenderedPage](#) object.

Returns a **List**<[PNConversionResultPrintJobPrintedPage](#)> collection.

See Also:

[GetOutputFile](#) [GetPrintJobPrintedPages](#)

Properties

BitsPerPixel

Description

This is the color depth, or bit depth of the page.

Read-only.

Syntax

expression.BitsPerPixel

where *expression* is an [PNConversionResultOutputFileRenderedPage](#) object.

Returns a **UInt32**.

See Also:

[HeightInPixels](#) [Orientation](#) [PageNumber](#) [RotationInDegrees](#)
[WidthInPixels](#) [XPixelsPerInch](#) [YPixelsPerInch](#)

HeightInPixels

Description

This is the height of the output page in pixels.

Read-only.

Syntax

expression.HeightInPixels

where *expression* is an [PNConversionResultOutputFileRenderedPage](#) object.

Returns a **UInt32**.

See Also:

[BitsPerPixel](#) [Orientation](#) [PageNumber](#) [RotationInDegrees](#)
[WidthInPixels](#) [XPixelsPerInch](#) [YPixelsPerInch](#)

Orientation

Description

This is the orientation, either *Portrait* or *Landscape*, of the output page.

Read-only.

Syntax

`expression.Orientation`

where `expression` is an [PNConversionResultOutputFileRenderedPage](#) object.

Returns a **UInt32** where Portrait = 0 and Landscape = 1.

See Also:

[BitsPerPixel](#) [HeightInPixels](#) [PageNumber](#) [RotationInDegrees](#)
[WidthInPixels](#) [XPixelsPerInch](#) [YPixelsPerInch](#)

PageNumber

Description

This is the number of the page in the output file.

Read-only.

Syntax

`expression.PageNumber`

where `expression` is an [PNConversionResultOutputFileRenderedPage](#) object.

Returns a **UInt32**.

See Also:

[BitsPerPixel](#) [HeightInPixels](#) [Orientation](#) [RotationInDegrees](#)
[WidthInPixels](#) [XPixelsPerInch](#) [YPixelsPerInch](#)

RotationInDegrees

Description

This is the rotation, one of 0° , 90° , 180° , 270° , of the page. Pages are always rotated counter-clockwise.

Read-only.

Syntax

`expression.BitsPerPixel`

where *expression* is an [PNConversionResultOutputFileRenderedPage](#) object.

Returns a **UInt32**, one of 0, 90, 180 or 270.

See Also:

[BitsPerPixel](#) [HeightInPixels](#) [Orientation](#) [PageNumber](#)
[WidthInPixels](#) [XPixelsPerInch](#) [YPixelsPerInch](#)

WidthInPixels

Description

This is the width of the page in pixels.

Read-only.

Syntax

`expression.WidthInPixels`

where *expression* is an [PNConversionResultOutputFileRenderedPage](#) object.

Returns a **UInt32**.

See Also:

[BitsPerPixel](#) [HeightInPixels](#) [Orientation](#) [PageNumber](#)
[RotationInDegrees](#) [XPixelsPerInch](#) [YPixelsPerInch](#)

XPixelsPerInch

Description

This is the vertical dots per inch (DPI), or resolution, of the output page when it is an image.

Read-only.

Syntax

expression.XPixelsPerInch

where *expression* is an [PNConversionResultOutputFileRenderedPage](#) object.

Returns a **UInt32**.

See Also:

[BitsPerPixel](#) [HeightInPixels](#) [Orientation](#) [PageNumber](#)
[RotationInDegrees](#) [WidthInPixels](#) [YPixelsPerInch](#)

YPixelsPerInch

Description

This is the horizontal dots per inch, or resolution, of the page.

Read-only.

Syntax

expression.YPixelsPerInch

where *expression* is an [PNConversionResultOutputFileRenderedPage](#) object.

Returns a **UInt32**.

See Also:

[BitsPerPixel](#) [HeightInPixels](#) [Orientation](#) [PageNumber](#)
[RotationInDegrees](#) [WidthInPixels](#) [XPixelsPerInch](#)

PNConversionResultPrintJob



Description

Many of the converters used by Document Conversion Service will use the Document Conversion Service 3.0 printer to do the conversion. For most documents there is only a single print job created when the document is printed, but some applications can send multiple jobs when printing a single file. One example of this is Excel when printing a workbook containing multiple worksheets at different print quality settings. Excel will create a separate print job for each group of worksheets with different print qualities.










Each PNConversionResultPrintJob object represents one print job. The job object is identified by a unique identifier, the [GUID](#) and contains information about the job such as the job status and the number of pages spooled and printed.




There are also two collections: a [PNConversionResultOutputFile](#) collection of files created by this job, and a [PNConversionResultPrintJobPrintedPage](#) collection of the printed pages belonging to this print job.

Methods

 GetOutputFiles	Read-only; Returns a collection of PNConversionResultOutputFile objects.
 GetPrintJobPrintedPages	Read-only; Returns a collection of PNConversionResultPrintJobPrintedPage objects.

Properties

 BytesPrinted	Read-only; How much of the document, in bytes, has been printed.
 BytesSpooled	Read-only; Size of the document (in bytes) in the printer queue.
 Title	Read-only; Name of the document printed.
 GUID	Read-only; Unique identifier for this object.
 JobID	Read-only; non-unique identifier used by the Windows printing sub-system.
 PagesPrinted	Read-only; count of the number of pages printed.
 PagesSpooled	Read-only; count of the number of pages spooled.
 Status	Read-only; current print status of the job as an Integer value.
 StatusMessage	Read-only; current print status of the job as a string value.

 Submitted	Read-only; The date and time this document was spooled.
 Title	Read-only; Name of the document printed.
 UserName	Read-only; name of the user who printed the document.

Methods

GetOutputFiles

Description

Returns a collection of [PNConversionResultOutputFile](#) objects, one for each file created. From this object you can get the full output filename of the created file.

Read-only.

Syntax

expression.GetOutputFiles

where *expression* is an [PNConversionResultPrintJob](#) object.

Returns a List< [PNConversionResultOutputFile](#)> collection.

See Also:

[GetPrintJobPrintedPages](#)

GetPrintJobPrintedPages

Description

Returns a collection of [PNConversionResultPrintJobPrintedPage](#) objects, one for page printed by this job.

Read-only.

Syntax

expression.GetPrintJobPrintedPages

where *expression* is an [PNConversionResultPrintJob](#) object.

Returns a List<[PNConversionResultPrintJobPrintedPage](#)> collection.

See Also:

[GetOutputFiles](#)

Properties

BytesPrinted

Description

Returns the size of the printed job in bytes. This can be different from [BytesSpooled](#).

Read-only.

Syntax

expression.BytesPrinted

where *expression* is an [PNConversionResultPrintJob](#) object.

Returns a **UInt64**.

See Also:

[BytesSpooled](#) [GUID](#) [JobID](#) [PagesPrinted](#) [PagesSpooled](#)
[Status](#) [StatusMessage](#) [Submitted](#) [Title](#) [UserName](#)

BytesSpooled

Description

The size of the spooled job in bytes.

Read-only.

Syntax

expression.BytesSpooled

where *expression* is an [PNConversionResultPrintJob](#) object.

Returns a **UInt64**.

See Also:

[BytesPrinted](#) [GUID](#) [JobID](#) [PagesPrinted](#) [PagesSpooled](#)
[Status](#) [StatusMessage](#) [Submitted](#) [Title](#) [UserName](#)

GUID

Description

A string based unique identifier for this object.

Read-only.

Syntax

expression.GUID

where *expression* is an [PNConversionResultPrintJob](#) object.

Returns a **String**.

See Also:

[BytesPrinted](#) [BytesSpooled](#) [JobID](#) [PagesPrinted](#) [PagesSpooled](#)
[Status](#) [StatusMessage](#) [Submitted](#) [Title](#) [UserName](#)

JobId**Description**

This is a non-unique numerical identifier used by the Windows printing sub-system.

Read-only.

Syntax

expression.JobId

where *expression* is an [PNConversionResultPrintJob](#) object.

Returns a **UInt32**.

See Also:

[BytesPrinted](#) [BytesSpooled](#) [GUID](#) [PagesPrinted](#) [PagesSpooled](#)
[Status](#) [StatusMessage](#) [Submitted](#) [Title](#) [UserName](#)

PagesPrinted**Description**

Returns the number of pages printed. This can be different from [PagesSpooled](#).

Read-only.

Syntax

expression.PagesPrinted

where *expression* is an [PNConversionResultPrintJob](#) object.

Returns a **UInt32**.

See Also:

[BytesPrinted](#) [BytesSpooled](#) [GUID](#) [JobID](#) [PagesSpooled](#)
[Status](#) [StatusMessage](#) [Submitted](#) [Title](#) [UserName](#)

PagesSpooled

Description

Returns the number of pages spooled. This can be different from [PagesPrinted](#).

Read-only.

Syntax

expression.PagesSpooled

where *expression* is an [PNConversionResultPrintJob](#) object.

Returns a **UInt32**.

See Also:

[BytesPrinted](#) [BytesSpooled](#) [GUID](#) [JobID](#) [PagesPrinted](#)
[Status](#) [StatusMessage](#) [Submitted](#) [Title](#) [UserName](#)

Status

Description

The print status of the job as a numerical value. See the **Remarks** section for a list of the status values and what they mean.

Read-only.

Syntax

expression.Status

where *expression* is an [PNConversionResultPrintJob](#) object.

Returns a **UInt32**.

Remarks

The status can be one or more of the values in the table below. These are the same values used by the *JOB_INFO_2* structure in Microsoft's Win32 Printing and Print Spooler functions and structures. See the Microsoft documentation for more details.

The values are OR'd together to define the current status of the job. To determine which values, the hexadecimal values must be examined:

If Status = 388, which is 0x00000184

JOB_STATUS_DELETED	0x00000100
--------------------	------------

```

JOB_STATUS_PRINTED      0x00000080
JOB_STATUS_DELETING     0x00000004
-----
                        0x00000184

```

Job Status	Hexadecimal Value	Integer Value
JOB_STATUS_PAUSED	0x00000001	1
JOB_STATUS_ERROR	0x00000002	2
JOB_STATUS_DELETING	0x00000004	4
JOB_STATUS_SPOOLING	0x00000008	8
JOB_STATUS_PRINTING	0x00000010	16
JOB_STATUS_OFFLINE	0x00000020	32
JOB_STATUS_PAPEROUT	0x00000040	64
JOB_STATUS_PRINTED	0x00000080	128
JOB_STATUS_DELETED	0x00000100	256
JOB_STATUS_BLOCKED_DEVQ	0x00000200	512
JOB_STATUS_USER_INTERVENTION	0x00000400	1024
JOB_STATUS_RESTART	0x00000800	2048
JOB_STATUS_COMPLETE	0x00001000	4096
JOB_STATUS_RETAINED	0x00002000	8192
JOB_STATUS_RENDERING_LOCALLY	0x00004000	16384

See Also:

[BytesPrinted](#) [BytesSpooled](#) [GUID](#) [JobID](#) [PagesPrinted](#)
[PagesSpooled](#) [StatusMessage](#) [Submitted](#) [Title](#) [UserName](#)

StatusMessage**Description**

The current print status of the job as a string value. This value can be an empty string.
Read-only.

Syntax

expression.StatusMessage

where *expression* is an [PNConversionResultPrintJob](#) object.

Returns an **String**.

See Also:

[BytesPrinted](#) [BytesSpooled](#) [GUID](#) [JobID](#) [PagesPrinted](#)
[PagesSpooled](#) [Status](#) [Submitted](#) [Title](#) [UserName](#)

Submitted**Description**

Returns the size of the printed job in bytes. This can be different from [BytesSpooled](#).

Read-only.

Syntax

expression.Submitted

where *expression* is an [PNConversionResultPrintJob](#) object.

Returns an **DateTime**.

See Also:

[BytesPrinted](#) [BytesSpooled](#) [GUID](#) [JobID](#) [PagesPrinted](#)
[PagesSpooled](#) [Status](#) [StatusMessage](#) [Title](#) [UserName](#)

Title

Description

The name of the document printed that created this print job. This is the name the printing application uses in the print queue. It can be different from the actual document name.

Read-only.

Syntax

expression.Title

where *expression* is an [PNConversionResultPrintJob](#) object.

Returns an **String**.

See Also:

[BytesPrinted](#) [BytesSpooled](#) [GUID](#) [JobID](#) [PagesPrinted](#)
[PagesSpooled](#) [Status](#) [StatusMessage](#) [Submitted](#) [UserName](#)

UserName

Description

Returns the name of the user who printed the document.

Read-only.

Syntax

expression.UserName

where *expression* is an [PNConversionResultPrintJob](#) object.

Returns an **String**.

See Also:

[BytesPrinted](#) [BytesSpooled](#) [GUID](#) [JobID](#) [PagesPrinted](#)
[PagesSpooled](#) [Status](#) [StatusMessage](#) [Submitted](#) [Title](#)

PNConversionResultPrintJobPrintedPage




Description

A `PNConversionResultPrintJobPrintedPage` object is created for every page of the document or file that is printed.








The page object represents the print settings of the page when spooled to the Document Conversion Service printer. These settings are different from the [PNConversionResultOutputFileRenderedPage](#) settings, which are the settings of the output file created. For instance, printing a single page document in color and creating a fax resolution TIFF image will give a `PNConversionResultPrintJobPrintedPage` object with a *BitsPerPixel* = 24, and a `PNConversionResultOutputFileRenderedPage` object with *BitsPerPixel* = 1.


There are also two collections: a [PNConversionResultOutputFileRenderedPage](#) collection of pages, currently only a collection of one, representing this page in the final output on disk, and a [PNConversionResultOutputFile](#) collection of files that contain this pages as a `PNConversionResultOutputFileRenderedPage` object.

Methods

 GetOutputFileRenderedPages	Read-only; Returns a collection of PNConversionResultOutputFileRenderedPage objects.
 GetOutputFiles	Read-only; Returns a collection of PNConversionResultOutputFile objects.
 GetPrintJob	Read-only; Returns a PNConversionResultPrintJob object.

Properties

 BitsPerPixel	Read-only; The bits per pixel, or color depth of the printed page.
 HeightInPixels	Read-only; The height of the printed page in pixels.
 Orientation	Read-only; The orientation of the page, either <i>Portrait</i> or <i>Landscape</i> .
 PageNumber	Read-only; The page number of the page.
 Skipped	Read-only; Boolean value True if the page was skipped.
 WidthInPixels	Read-only; The weight of the printed page in pixels.
 XPixelsPerInch	Read-only; The vertical dots per inch, or resolution, of the page.

 YPixelsPerInch	Read-only; The horizontal dots per inch, or resolution, of the page.
--	--

Methods

GetOutputFileRenderedPages

Description

Returns a collection of [PNConversionResultOutputFileRenderedPage](#) objects, one for every page in the output physical file on disk.

Read-only.

Syntax

expression.GetOutputFileRenderedPages

where *expression* is an [PNConversionResultPrintJobPrintedPage](#) object.

Returns a **List**<[PNConversionResultOutputFileRenderedPage](#)> collection.

See Also:

[GetOutputFiles](#) [GetPrintJob](#)

GetOutputFiles

Description

Returns a collection of [PNConversionResultOutputFile](#) objects, one for each file created. From this object you can get the full output filename of the created file.

Read-only.

Syntax

expression.GetOutputFiles

where *expression* is an [PNConversionResultPrintJobPrintedPage](#) object.

Returns a **List**<[PNConversionResultOutputFile](#)> collection.

See Also:

[GetOutputFileRenderedPages](#) [GetPrintJob](#)

GetPrintJob

Description

Returns the [PNConversionResultPrintJob](#) object that created this [PNConversionResultPrintJobPrintedPage](#).

Read-only.

Syntax

expression.GetPrintJob

where *expression* is an [PNConversionResultPrintJob](#) object.

Returns a [PNConversionResultPrintJob](#) object.

See Also:

[GetOutputFileRenderedPages](#) [GetOutputFiles](#)

Properties

BitsPerPixel

Description

This is the color depth, or bit depth of the page. This can be different from the [BitsPerPixel](#) values in any [PNConversionResultOutputFileRenderedPage](#) objects in the collection returned from [GetOutputFileRenderedPages](#) method. It is commonly 1 for black and white, or monochrome printing, and 24 when printing in color.

Read-only.

Syntax

expression.BitsPerPixel

where *expression* is an [PNConversionResultPrintJobPrintedPage](#) object.

Returns a **UInt32**.

See Also:

[HeightInPixels](#) [Orientation](#) [PageNumber](#) [Skipped](#)
[WidthInPixels](#) [XPixelsPerInch](#) [YPixelsPerInch](#)

HeightInPixels

Description

This is the height of the page in pixels.

Read-only.

Syntax

`expression.HeightInPixels`

where *expression* is an [PNConversionResultPrintJobPrintedPage](#) object.

Returns a **UInt32**.

See Also:

[BitsPerPixel](#) [Orientation](#) [PageNumber](#) [Skipped](#)
[WidthInPixels](#) [XPixelsPerInch](#) [YPixelsPerInch](#)

Orientation

Description

This is the orientation, either *Portrait* or *Landscape*, of the page when printed.

Read-only.

Syntax

`expression.Orientation`

where *expression* is an [PNConversionResultPrintJobPrintedPage](#) object.

Returns a **UInt32** where Portrait = 0 and Landscape = 1..

See Also:

[BitsPerPixel](#) [HeightInPixels](#) [PageNumber](#) [Skipped](#)
[WidthInPixels](#) [XPixelsPerInch](#) [YPixelsPerInch](#)

PageNumber

Description

This is the page number of the printed page. This can be different from the page number of the page in the resulting file on disk.

Read-only.

Syntax

`expression.PageNumber`

where *expression* is an [PNConversionResultPrintJobPrintedPage](#) object.

Returns a **UInt32**.

See Also:

[BitsPerPixel](#) [HeightInPixels](#) [Orientation](#) [Skipped](#)
[WidthInPixels](#) [XPixelsPerInch](#) [YPixelsPerInch](#)

Skipped

Description

This property is **True** if the page was skipped.

Read-only.

Syntax

expression.Skipped

where *expression* is an [PNConversionResultPrintJobPrintedPage](#) object.

Returns a **UInt32**.

See Also:

[BitsPerPixel](#) [HeightInPixels](#) [Orientation](#) [PageNumber](#)
[WidthInPixels](#) [XPixelsPerInch](#) [YPixelsPerInch](#)

WidthInPixels

Description

This is the width of the page in pixels.

Read-only.

Syntax

expression.WidthInPixels

where *expression* is an [PNConversionResultPrintJobPrintedPage](#) object.

Returns a **UInt32**.

See Also:

[BitsPerPixel](#) [HeightInPixels](#) [Orientation](#) [PageNumber](#)
[Skipped](#) [XPixelsPerInch](#) [YPixelsPerInch](#)

XPixelsPerInch

Description

This is the vertical dots per inch (DPI), or resolution, of the page.

Read-only.

Syntax

expression.XPixelsPerInch

where *expression* is an [PNConversionResultPrintJobPrintedPage](#) object.

Returns a **UInt32**.

See Also:

[BitsPerPixel](#) [HeightInPixels](#) [Orientation](#) [PageNumber](#)

[Skipped](#) [WidthInPixels](#) [YPixelsPerInch](#)

YPixelsPerInch

Description

This is the horizontal dots per inch (DPI), or resolution, of the page.

Read-only.

Syntax

expression.YPixelsPerInch

where *expression* is an [PNConversionResultPrintJobPrintedPage](#) object.

Returns a **UInt32**.

See Also:

[BitsPerPixel](#) [HeightInPixels](#) [Orientation](#) [PageNumber](#)

[Skipped](#) [WidthInPixels](#) [XPixelsPerInch](#)


PNProfile

Description


The PNProfile class provides an interface for working with the profiles that Document Conversion Service uses to convert documents. Profiles control both the type of file created and optionally the behavior of the converters.

A profile is a structured XML file on disk that contains the list of settings. The settings are organized as a list of name\value pairs in the XML document. See [Creating and Customizing Profiles](#) for more information on the default profiles that come with Document Conversion Service, where they are stored, and how to modify existing or create new profiles.

Static Methods

 GetListofProfileNames	Returns a list of the existing profiles in specified location.
---	--

Enumerations

 PNProfileSearchLocation	The location in which to look for the profiles.
---	---

Methods

GetListofProfileNames

Description

Static method.

Returns a list of profile names from the location specified. Profiles are stored as XML document on disk. A profile name is the file name of the XML document without the *.xml* extension.

Syntax

```
PNProfile.GetListofProfileNames(searchLevel)
```

```
PNProfile.GetListofProfileNames(AlternatePath)
```

Returns an **ICollection<String>** collection of profiles names from the specified location.

Parameters

PNProfileSearchLocation searchLevel

The location in which to search, one of [PNProfileSearchLocation](#).values.

String AlternatePath

The full path to an alternate location to search for profiles.

Enumerations

PNProfileSearchLocation

Description

Where to look for profile files.

Details

Name	Value	Description
DefaultProfiles	0	Returns profiles included with the Document Conversion Service install. The default profiles are stored in a global location available for all users on the computer.
UserProfiles	1	Returns profiles that are stored in the user's local data folder.
DefaultAndUserProfiles	2	Returns all profiles found in both the default profile location and the user's local data folder.


PNSetting

Description



A PNSetting class defines a name/value pair that describes a conversion setting. The name/value pairs that can be used are the same settings that are used to create the XML-formatted profiles included with Document Conversion Service. See [Conversion Settings](#) for a list of all of the settings that are available.

The PNSetting class is used to hold collections of settings in the following classes: [PNConversionItem](#), [PNConvertFileInfo](#) and [PNProfile](#).

Methods

 PNSetting	Initializes a new instance of a PNSetting object.
---	---

Properties

 Name	Gets or sets the name in the name/value pair.
 Value	Gets or sets the value in the name/value pair.

Methods

PNSetting

Description

Initializes an instance of the [PNSetting](#) object with the specified name and value.

Syntax

```
PNSetting(name, value)
```

Parameters

String name

The name of the conversion setting. See [Conversion Settings](#) for a list of all the name/value pairs of settings that are available.

String value

The value associated with *name*.

Properties

Name

Description

Gets or sets the name in the name/value pair. See [Conversion Settings](#) for a list of names for all of the settings that are available.

Syntax

expression.Name

where *expression* is an [PNSetting](#) object.

Returns a **String**.

See Also:

[Value](#)

Value

Description

Gets or sets the value in the name/value pair. See [Conversion Settings](#) for the list of names and the values that can be set for each.

Syntax

expression.Name

where *expression* is an [PNSetting](#) object.




Returns a **String**.

See Also:

[Name](#)

Enumerations

The following enumerations are used in the PEERNET.ConvertUtility namespace.

 PNConvertResultStatus	Conversion status result as a short string message.
 PNFileSortMode	Sort the files none (system default), name, date created or date modified when picking up files from system.
 PNFileSortOrder	The order of the files, either Ascending or Descending.

PNConvertResultStatus

Description

Conversion status result as a short string message.

Details

Name	Value (String)
ResultStatus_SUCCESS Conversion was successful.	SUCCESS
ResultStatus_FAIL Generic failure error.	FAIL
ResultStatus_FILECOPY_FAIL_MAXATTEMPTS File copy failed after max attempts (20) to copy it to the destination.	FILECOPY_FAIL_MAXATTEMPTS
ResultStatus_OVERWRITE_FAIL_EXISTING File could not be overwritten because a file of the same name exists.	OVERWRITE_FAIL_EXISTING
ResultStatus_DIRCREATE_FAIL Directory could not be created.	DIRCREATE_FAIL
ResultStatus_EXCEPTION An exception was thrown during conversion.	EXCEPTION

PNFileSortMode

Description

Determines the sorting mode, if any, applied when picking up files from an input folder.

Details

Name	Value (int)
None	0
No sorting is performed, files are listed as returned from the system. <i>Default.</i>	
Name	1
Files are sorted by file name.	
DateCreated	2
Files are sorted using the file creation date on the file.	
DateModified	3
Files are sorted using the last modified date on the file.	

PNFileSortOrder

Description

Determines the sorting order, if any, applied when picking up files from an input folder.

Details

Name	Value (int)
Ascending	0
Sorts the files from low to high: 0-9, A-Z.	
Descending	1
Sorts the files from high to low: Z-A, 9-0.	

Setting up Client-Server Conversion

Document Conversion Service supports client-server conversion using DCOM (*Distributed Component Object Model*).

This scenario would be commonly used when running a web service that converts files running on one computer with Document Conversion Service running on another computer. When the web server needs to convert a file it will "talk" to the computer that the conversion service is running on and tell it to convert the files. This is referred to as a *client-server* relationship where the web server is the *client* and the computer running Document Conversion Service is the *server*.

Another example of this is when the Document Conversion Service is running on a server and a small application to convert files is installed on each user's machine. This keeps all the heavy work of document conversion on the server and not on the user's machine. In this case each user's machine is the client.

Typical Client-Server Configurations

When setting up client-server conversion, you will need to know if your computers are running on a *domain*, on a *workgroup*, or a mix of the two.

The most common usage scenarios are explained below.

- The simplest setup is when both the clients and the server running Document Conversion Service are on the same *domain*. In this case, nothing needs to be done to allow conversion.
- If the server is on a *domain*, and the client is a local user, a matching local account with the same user name and password must be created on the server.
- If the server is on a *workgroup*, a matching local account with the same user name and password as will need to exist on both the client and the server computer.

Best Practices

For remote, or client-server conversion, Document Conversion Service is first installed on the computer that will be performing the conversions, then an additional setup component, the *Document Conversion Service Client Redistributable* is installed on the client machines to allow the client machines to talk to the server. The client redistributable is included in the Document Conversion Service install.

Document Conversion Service is initially installed so that all authenticated users are able to communicate in client-server conversion. This is done during installation by creating a local group, *Document Conversion Service Users* and adding the *Authenticated Users* group as a member of the local group. Permissions on the required Document Conversion Service components and folders are all set using the *Document Conversion Service Users* group to allow client-server conversion to work out of the box with little or no additional configuration needed.

The client install also creates the *Document Conversion Service Users* local group and adds the *Authenticated Users* group to it's member list, as well as setting permissions on any required components and folders using this group.

For situations where you only want certain users to be able to perform remote conversion, you can edit the members list of the *Document Conversion Service Users* group on the server and the client to remove the *Authenticated Users* group and add your desired users, or an existing group of users you may already have set up.

When setting up Document Conversion Service for client-server communication, we also recommend the following best practices below.

- You will need to have access to an account with *Administrative* rights to run Document Conversion Service on the server and the permissions to add local accounts and groups on the server machine and to modify a group's membership.
- When installing Document Conversion Service and Document Conversion Service Client Redistributable, let the setups create the local administrative account, DCSAdmin, with the same user name and password on both the server and any clients.

Setting up the Server

The first step is to install and configure the Document Conversion Service on the computer that will be the conversion server.

Install Document Conversion Service

Document Conversion Service is initially installed so that all authenticated users are able to communicate in client-server conversion using the created local group *Document Conversion Service Users*. This allows remote conversion to work out of the box with little to no configuration needed.

A Logon Account to be used for Document Conversion Service is needed during the install. We recommend allowing the install to create the local administrative user account, *DCSAdmin*. You can instead choose to use a different local or domain account, but it must have administrative privileges.



Note

If you create the local DCSAdmin account during the install, make sure you write down or store the password you used when creating this account. You will need to use the same password when installing the client software.

If you want to restrict access to only certain user accounts or groups, you can edit the members list of the *Document Conversion Service Users* group on the server and any clients to remove the *Authenticated Users* group and add your desired users or group. Instructions for this are included below in the [Editing the Document Conversion Service Users Group Member List](#) section.

See [Installing Document Conversion Service Silently](#) for command line parameters for silent installation of both Document Conversion Service and Document Conversion Service Client Redistributable.

1. Install Document Conversion Service on the computer that will be the server. The install creates the following group and network share and uses them for client-server communication:
 - a. A local group named *Document Conversion Service Users* is added. The account set as the Logon Account during the install and the *Authenticated Users* group are added to this local group.
 - b. The folder *C:\PEERNET\DCSREMOTE* is created and assigned the share name of **DCSREMOTE**. This share folder will be used by the clients to allow the conversion server access to the files being converted.
 - c. Full permissions for anyone part of the *Document Conversion Service Users* group are added to the *C:\PEERNET* root of the network share folder.
 - d. The *Document Conversion Service Users* is used to apply permissions on the DCOM components needed for remote conversion.
2. Once installed, configure Document Conversion Service for the file types you want to convert. See [Starting and Stopping the Service](#) and [Advanced Configuration](#) to configure your server for the file types you want to convert.
3. Test your server configuration with the sample program as shown in [The Convert File Sample](#). At this point all conversion is local; you will not need to set any of the Remote Conversion Settings in the sample at this point.

4. Review the next section [Additional Client-Server Configuration](#) for any further setup that may be needed in your environment.

Additional Client-Server Configuration

The following sections outline any additional configuration needed for different domain/workgroup client-server scenarios, as well as how to remove the *Authenticated Users* membership from the *Document Conversion Service Users* group and lock down the client-server access by adding your own groups or user accounts.

My Server is on a Workgroup...

When the server is on a workgroup, the client can use accounts that are on the domain, or local user accounts on the client. In both cases local accounts with matching user names and passwords must be created on the server.

Local accounts already have the required DCOM access through the *Authenticated Users* group added to the local group *Document Conversion Service Users* and no more changes should be needed.

To lock down the accounts that have access, do the following:

1. On the server:
 - a. If needed, add the new local account with the matching user name and password as the account on the client.
 - b. Once created, add this user as a member of the *Document Conversion Service Users* group and remove the *Authenticated Users* group.
2. On the client:
 - a. Install the Document Conversion Service Client Redistributable setup using the same account and password used when installing Document Conversion Service on the server.
 - b. Edit the member list of the *Document Conversion Service Users* group to add the account you are using to perform conversion on the client and remove the *Authenticated Users* group.

My Server is on a Domain...

When the server is on a domain the client can use accounts that are also on the domain, or local user accounts on the client.

Clients are Using Domain Accounts

The simplest configuration is when both the server and the client machines are all part of the same domain, and you are using domain accounts. In this scenario, the install should have already added the *Authenticated Users* group to the local group *Document Conversion Service Users* and no further setup should be needed.

If you want tighter control over your accounts and permissions, you can remove the *Authenticated Users* group from the member list of *Document Conversion Service Users* group and add the specific domain users as members of this group on both the server and the client machine as shown in the [Editing the Document Conversion Service Users Group Member List](#) section.

Clients are Using Local Accounts

If the clients and server are all part of the same domain, but the accounts you are using to perform the conversion on the client machines are local accounts, you will need to create a matching account on the server with the same user name and password as the account on the client machine.

Local accounts already have the required DCOM access through the *Authenticated Users* group added to the local group *Document Conversion Service Users*.

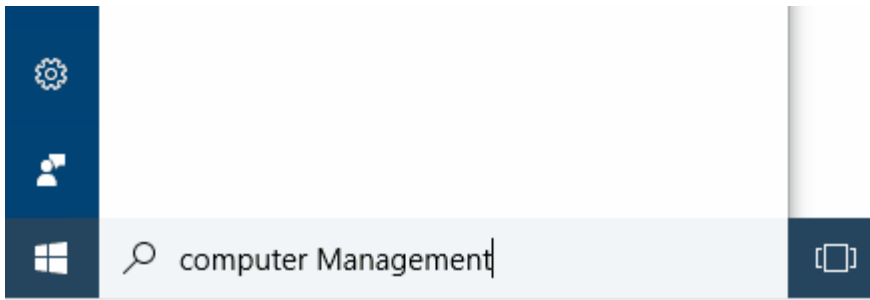
If you want more granular control over the accounts that have access, do the following:

1. On the server:
 - a. If needed, add the new new local account with the matching user name and password as the account on the client.
 - b. Once created, add this user as a member of the *Document Conversion Service Users* group and remove the *Authenticated Users* group.
2. On the client:
 - a. Install the Document Conversion Service Client Redistributable setup using the same account and password used when installing Document Conversion Service on the server.
 - b. Edit the member list of the *Document Conversion Service Users* group to add the account you are using to perform conversion on the client and remove the *Authenticated Users* group.

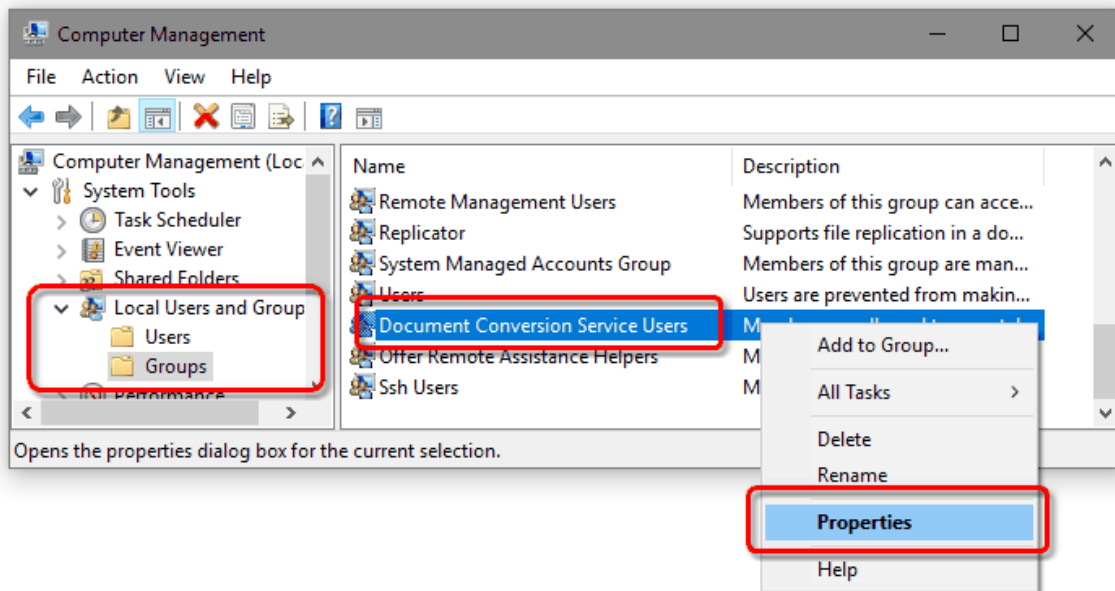
Editing the Document Conversion Service Users Group Member List

Which users are allowed to communicate in client-server conversion are controlled using the membership list of our local group, *Document Conversion Service Users*. The install initially sets this group to include all authenticated users by including the *Authenticated Users* group as a member. By removing this group and adding your own users and groups you can lock down which accounts have access.

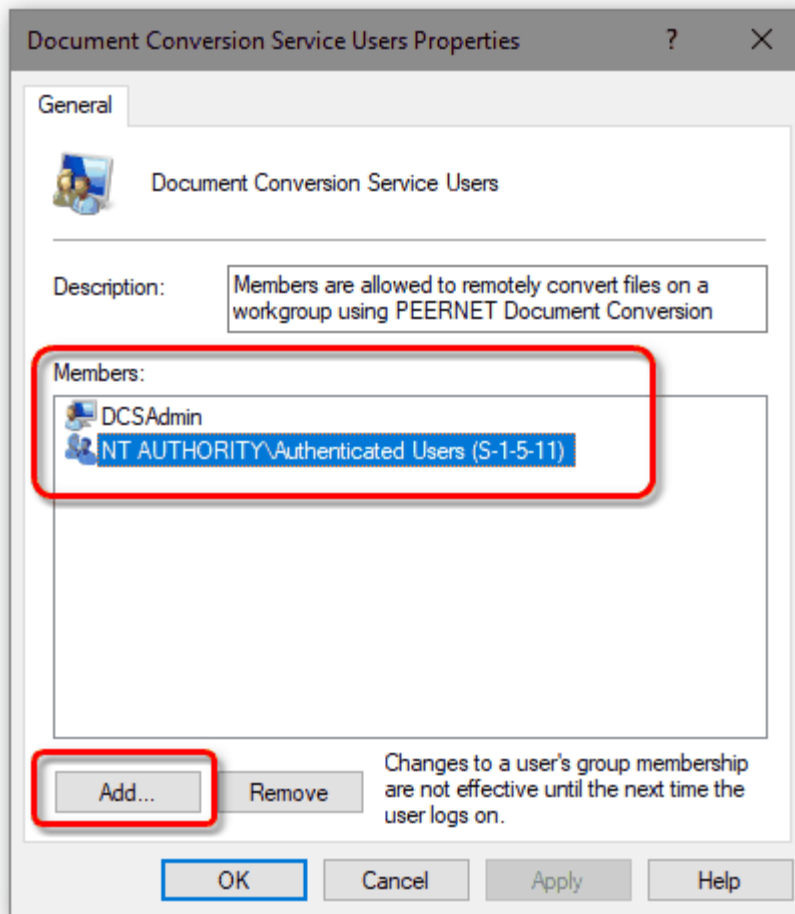
1. Type "Computer Management" into the search field on the Start menu to open the **Computer Management** tool and edit the local groups. You can also get access to this tool from the **Administrative Tools** section of the Control Panel.



2. Under **Local Users and Group** select **Groups** to see all local groups. Right-click on the *Document Conversion Service Users* group and select *Properties*.



3. On the Properties dialog, select the *Authenticated Users* group and delete it from the group using the **Remove** button. Then use the **Add** button to add the desired users or company user groups.



4. Click **Apply** and **OK** to save your changes.

Setting up the Client

After Document Conversion Service has been installed and configured on the machine that you want to use as the server you need to install the redistributable client program on each client computer to make the connection between the client and the server. If you have created any [custom conversion profiles](#) that you are using, they too will need to be copied to the client machine.

Installing the Client Redistributable

The client redistributable, PNDocConvClientSetup_3.0.exe, is included as part of the Document Conversion Service install.

It can be found in the \Samples\Redist folder under your Document Conversion Service installation path. Copy the client setup program, PNDocConvClientSetup_3.0.exe, from the server where you have installed Document Conversion Service to the client computer, or a location that can be accessed from the client computer and run the setup on the client computer.

A Logon Account to be used for Document Conversion Service by the client is needed during the install. We recommend allowing the install to create the local administrative user account, DCSAdmin. You can instead choose to use a different local or domain account if desired.



Note

If you create the local DCSAdmin account during the install, you will need to use the same password you used installing Document Conversion Service on the server.

The client setup will install the following:

- the client component that allows the client to communicate with the server
- the default set of conversion profiles included with Document Conversion Service; if you have created any custom conversion profiles you will need to copy them over to the client machine as well.
- a local group named *Document Conversion Service Users* is added, and two members, the account set as the Logon Account during the client install and the *Authenticated Users* group are added to this local group.

A *Minimum* install is the default and installs the above components. If a *Complete* install is chosen, the Watch Folder Service and sample code, the command line conversion tools and all additional sample code is also installed. You can choose exactly what parts are installed by selecting a *Custom* install.

For future installs, bundling the client with your own install, or push software installation, see [Installing PNDocConvClientSetup_3.0.exe Silently](#) for command line parameters for silent installation.

Editing Client Permissions

The client install is initially configured so that all authenticated users are able to communicate in client-server conversion using the created local group *Document Conversion Service Users*. This allows remote conversion to work out of the box with little to no configuration needed.

To restrict permissions to only specific users, or to use an existing group to apply permissions, remove *Authenticated Users* from the *Document Conversion Service Users* group members and add your desired user or group as shown in the section [Editing the Document Conversion Service Users Group Member List](#).

Running the Convert File Sample



Before you begin...

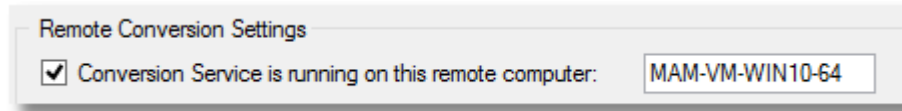
Before you can test on the client you need to have Document Conversion Service running on the remote server as per the steps in [Starting and Stopping the Service](#).

When the service has started, the Convert File sample application can be run to test the client-server communication.

1. Open the C# sample by going to Start - All Programs - PEERNET Document Conversion Service Client 3.0 – Samples - C# - Run Convert File Sample.

2. Choose a file to convert using the **Browse** button or by typing in the file name. The *Output File Name* field will be populated from the chosen file name.
3. Choose a folder in which to store the new file.

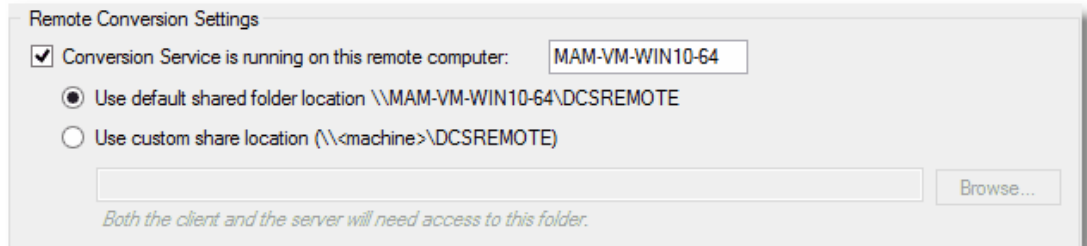
4. Choose the [profile](#) to use to create the file. The sample defaults to TIFF images but PDF or JPEG can be created as well.
5. Enable the *Conversion Service is running on this remote computer* checkbox.
 - a. Type in the name of the server where Document Conversion Service is installed and running. If this field is not filled in the conversion will not succeed.



Remote Conversion Settings

☒ Conversion Service is running on this remote computer:

- b. For client-server conversion a temporary conversion folder that is accessible to both the client and the server is required. A network shared folder named *DCSREMOTE* is automatically created on the server as part of the Document Conversion Service installation and already has the required permissions for any users who are part of the *Document Conversion Service Users* group. You can leave this folder selected, or use your own custom share folder. If you use a different share folder, you will need to give the *Document Conversion Service Users* group full permissions in that folder.



Remote Conversion Settings

☒ Conversion Service is running on this remote computer:

☒ Use default shared folder location \\MAM-VM-WIN10-64\DCSREMOTE

☐ Use custom share location (\\<machine>\DCSREMOTE)

Both the client and the server will need access to this folder.

6. Click Convert to convert the chosen file. The file will be created in the output folder selected and when the conversion process is finished, the results are displayed in the list box at the bottom.

Setting up a Client-Server Watch Folder

In this configuration the complete install of Document Conversion Service Client Redistributable, which includes the Watch Folder Service, would be installed on the *client* computer and the Document Conversion Service would be installed on a separate computer, the *server* computer.



Note

The Document Conversion Service Client Redistributable installs only the basic required components by default. To also install the Watch Folder service, choose the *Complete* install, or select *Custom* and then choose which samples and tools to install.

We want the input and output folders to be local to the *client* computer and the actual conversion done on the *server*, here a computer named *DOC-CONV-SRV1*.

To accomplish this, the server needs access to the staging and working folders used by the Watch Folder Service.

The simplest way to do this is to use the network share folder, **DCSREMOTE**, that was created when Document Conversion Service was installed on *DOC-CONV-SRV1*.

If you want to use a different network share, you will need to add full permissions for the *Document Conversion Service Users* group to the shared folder.



Sample Watch Folder Configuration

```
<WatchFolders>

  <WatchFolder Name="Shared DCOM Folder Watch FAX TIFF" >
    <Settings>
      <!-- Folder options -->
      <add Name="InputFolder" Value="C:\PEERNET\InputFax\"/>
      <add Name="SearchFilter" Value="*.*/>
      <add Name="IncludeSubFolders" Value="True"/>
      <add Name="StagingFolder" Value="\\DOC-CONV-SRV1\DCSREMOTE\WatchFolder\Staging\"/>
      <add Name="WorkingFolder" Value="\\DOC-CONV-SRV1\DCSREMOTE\WatchFolder\Working\"/>
      <add Name="FailedFolder" Value="C:\PEERNET\Failed\"/>
      <add Name="CompletedFolder" Value="C:\PEERNET\Completed\"/>
      <add Name="OutputFolder" Value="C:\PEERNET\FAX TIFF OUT\"/>
      <add Name="PollingInterval" Value="15000"/>
      <add Name="DCOMComputerName" Value="DOC-CONV-SRV1"/>
      <add Name="TestMode" Value="false" />
      ....
      <add Name="Devmode settings;Resolution" Value="300"/>
      <add Name="Save;Output File Format" Value="TIFF Serialized" />
      <add Name="Save;Append" Value="0"/>
      <add Name="Save;Color reduction" Value="BW"/>
      <add Name="TIFF File Format;BW compression" Value="Group4" />

      <add Name="Image Options;Fax" Value="1" />
      <add Name="Image Options;Fax Profile" Value="0" />
      <add Name="Image Options;Fax Resolution" Value="4" />
      <add Name="Processing;Rotate landscape" Value="90" />
    </Settings>
  </WatchFolder>
</WatchFolders>
```

Microsoft IIS and Document Conversion Service

Starting with Document Conversion Service 3.0.015, integrating conversion using a web service in Microsoft IIS has been simplified to make it easier to add file conversion into your web services. If you are running an earlier version of Document Conversion Service, see the section [IIS and Previous Versions of Document Conversion Service](#) below.

Local File Conversion with IIS

The addition of the *Authenticated Users* group to our local group *Document Conversion Service Users* automatically gives any IIS application pools access to all of the COM objects needed and to our default conversion folder *C:\PEERNET*. Adding access to other folders is as simple as adding full permissions for the *Document Conversion Service Users* group to that folder.

For situations where you only want certain users to be able to perform conversion, you can edit the members list of the *Document Conversion Service Users* group on the server and the client to remove the *Authenticated Users* group and add your desired users, or an existing group of users you may already have set up.

When calling the *PEERNET.ConvertUtility* methods or any of the command line utilities in a web service, the *ConversionWorkingFolder* argument must be specified and the *Document Conversion Service Users* group must have *Full control* or at least *Modify* permissions on that folder or the conversion will fail.

The sample code below is from a WCF service hosted on an IIS website.

```
public String ConvertToTIFF(String InputFile, String OutputFolder,
                          String ConversionWorkingFolder)
{
    String resultsStr = String.Empty;
    String convworkingFolder = @"C:\PEERNET\";

    if (!String.IsNullOrEmpty(ConversionWorkingFolder)) {
        convworkingFolder = ConversionWorkingFolder;
    }

    PNConversionItem convItem = null;
    try
    {
        convItem = PEERNET.ConvertUtility.PNConverter.ConvertFile(
            InputFile, OutputFolder, Guid.NewGuid().ToString(),
            true, true, true, "TIFF 200dpi Monochrome",
            String.Empty, String.Empty, null,
            String.Empty, convworkingFolder, String.Empty);

        if (convItem == null) {
            resultsStr += "Null item";
        }
        else {
            if (convItem.HasErrors()) {
                resultsStr += "There were errors";

                foreach (PNConversionResultError error in convItem.ConversionResult.Errors) {
                    resultsStr += error.Value;
                    resultsStr += "; ";
                }
            }
            else {
                resultsStr += "Converted file.";
            }
        }
    }
    catch (Exception ex) {
        resultsStr += String.Format("Exception occurred converting {0} to folder {1}. [{2}]",
                                   InputFile, OutputFolder, ex.Message);
    }

    return resultsStr;
}
```

Conversion can be called from another application (the service *consumer*) as follows:

```
static void Main(string[] args)
{
    ConverterService.ConverterServiceClient client =
        new ConverterService.ConverterServiceClient();

    String results = client.ConvertToTIFF(@"C:\PEERNET\files.txt",
                                         @"C:\PEERNET\", String.Empty);

    Console.WriteLine(results);

    // Always close the client.
    client.Close();

    Console.WriteLine("Press any key to exit...");
    Console.ReadKey();
}
```

Remote Conversion with IIS

If you plan on running your IIS server and Document Conversion Service on separate computers and converting remotely (client-server conversion), a few extra steps are needed.

For remote conversion, the application pool that the web service is running under will need access to the shared folder DCSREMOTE on the remote machine running Document Conversion Service. The easiest way to do this is to have the web service use an application pool that is configured to use an identity (an account) that has access to the remote machine. This identity can be a local account with matching passwords that exists on both machines or, if the computers are on the domain, it can be a domain account.

This account also needs access to the input, or the location of the file being converted, as well as the output location where the file is being saved.

The sample code below is for remote conversion from a WCF service hosted on an IIS website.

```
public String ConvertToTIFFRemote(String InputFile, String OutputFolder,
                                String RemotedCOMName, String RemoteWorkingFolder)
{
    String resultsStr = String.Empty;
    PNConversionItem convItem = null;

    try
    {
        convItem = PEERNET.ConvertUtility.PNConverter.ConvertFile(
            InputFile, OutputFolder, Guid.NewGuid().ToString(),
            true, true, true, "TIFF 200dpi Monochrome",
            String.Empty, String.Empty, null,
            RemotedCOMName, RemoteWorkingFolder, String.Empty);

        if (convItem == null) {
            resultsStr += "Null item";
        }
        else {
            if (convItem.HasErrors()) {
                resultsStr += "There were errors";

                foreach (PNConversionResultError error in convItem.ConversionResult.Errors) {
                    resultsStr += error.Value;
                    resultsStr += "; ";
                }
            }
            else {
                resultsStr += "Converted file.";
            }
        }
    }
    catch (Exception ex) {
        resultsStr += String.Format("Exception occurred converting {0} to folder {1}. [{2}]",
                                   InputFile, OutputFolder, ex.Message);
    }

    return resultsStr;
}
```

Conversion can be called from another application (the service *consumer*) as follows:

```
static void Main(string[] args)
{
    ConverterService.ConverterServiceClient client =
        new ConverterService.ConverterServiceClient();

    String results = client.ConvertToTIFFRemote(@"C:\PEERNET\Bliss.bmp", @"C:\PEERNET\",
                                              @"DCS-CONV-SRV", @"\\DCS-CONV-SRV\DCSRemote");

    Console.WriteLine(results);

    // Always close the client.
    client.Close();

    Console.WriteLine("Press any key to exit...");
    Console.ReadKey();
}
```

IIS and Previous Versions of Document Conversion Service

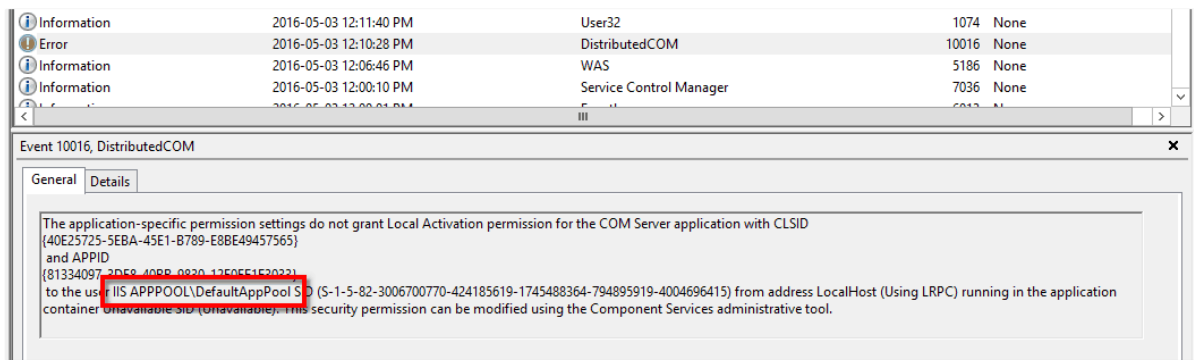
These steps demonstrate the changes needed to allow previous versions of Document Conversion Service to be called from an IIS web service application.

To perform the conversion the user (or identity) that the web service runs as needs to be added to the Document Conversion Service Users local user group to have access to Document Conversion Service.

The user group Document Conversion Service Users is created when Document Conversion Service or the Document Conversion Service Client Redistributable is installed.

Identify the User

The first step to be done is to identify the user that IIS is using to run the web service. The default user is normally IIS APPPOOL\DefaultAppPool. If you are using a different user, you can find out this information by locating the *DistributedCOM* error in the System log of the Event Viewer.

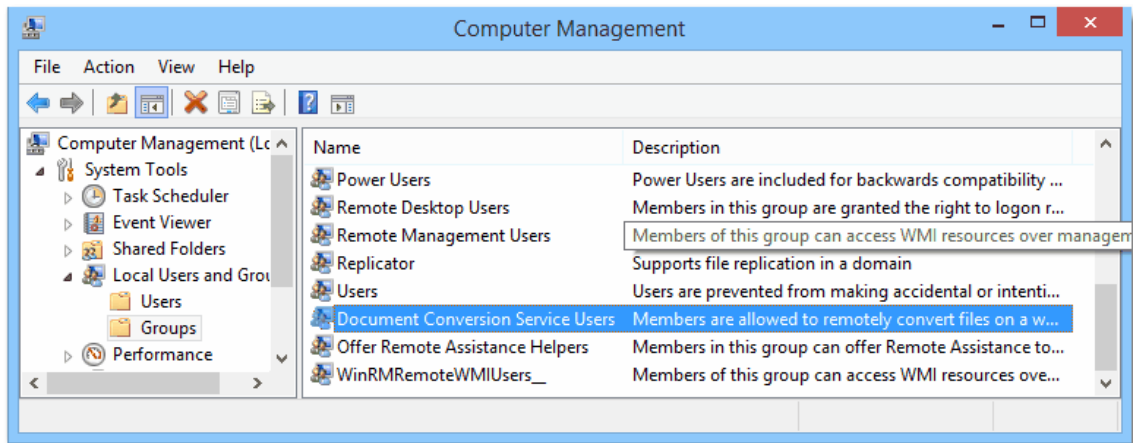


Add the User to Document Conversion Service Users

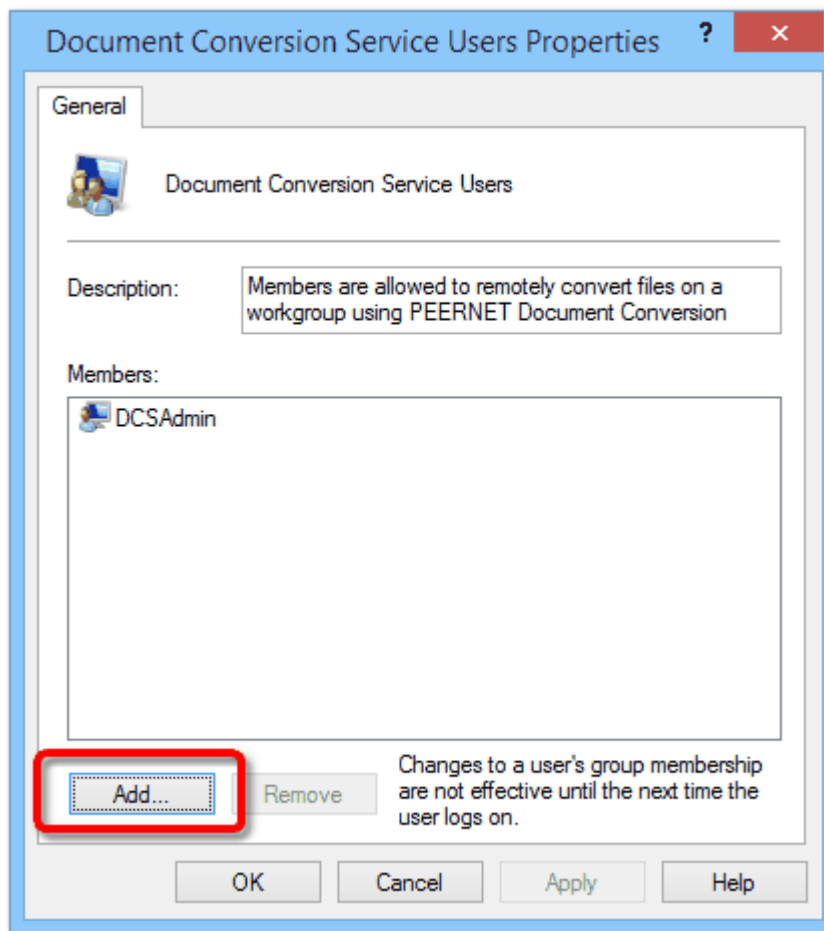
Once you know the user, you will need to add this user to the Document Conversion Service Users group so that it will have the necessary permissions. Once you have done the following steps, you must restart your computer to have the changes take effect.

1. From the **Control Panel**, go to **System and Security** and then to **Administrative Tools**. From here, open the *Computer Management* console.

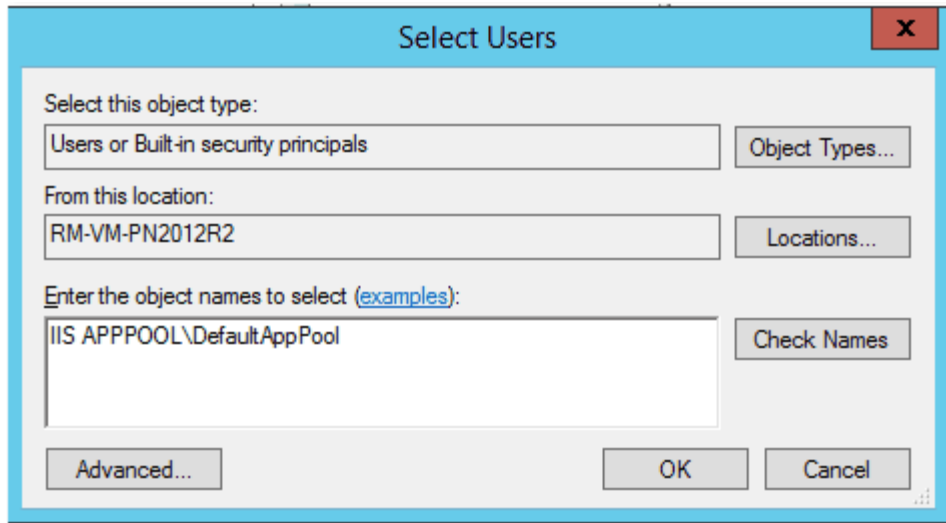
2. In the console, under **Computer Management (local) - Local Users and Groups - Groups**, locate and select the *Document Conversion Service Users* group.



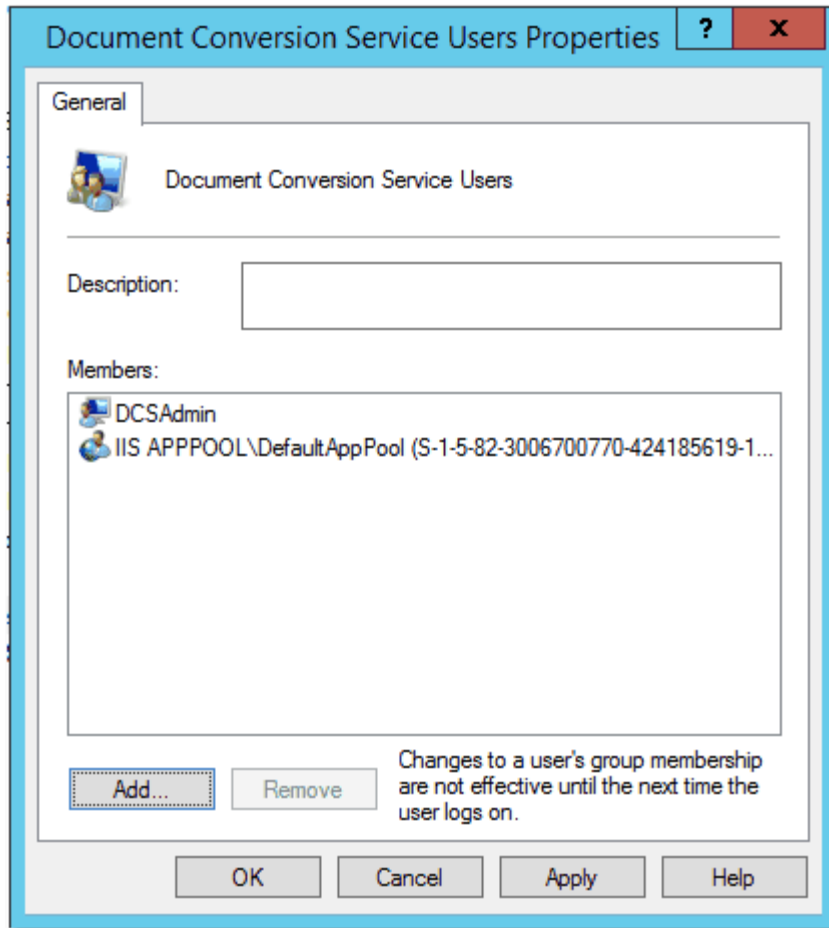
3. Double-click the group to open the *Document Conversion Service Users Properties* dialog, then click the **Add** button.



4. In the *Select Users* dialog change the **From this location:** to be the local computer (it normally defaults to the domain if you are on one) and add the desired user in the list at the bottom. Here we have added the default user IIS APPPOOL\DefaultAppPool; your actual user may be different. When done, press the **OK** button.



5. The *Document Conversion Service Users Properties* dialog should now look like the one below. Press the **Apply** button to save the changes.



6. This is the **LAST AND MOST IMPORTANT STEP**. You need to restart the computer to have the changes take effect. If you do not restart the computer you will still get the DistributedCOM error when trying to use Document Conversion Service from within the IIS environment.

Adding Document Conversion Service Users Permissions to Folders

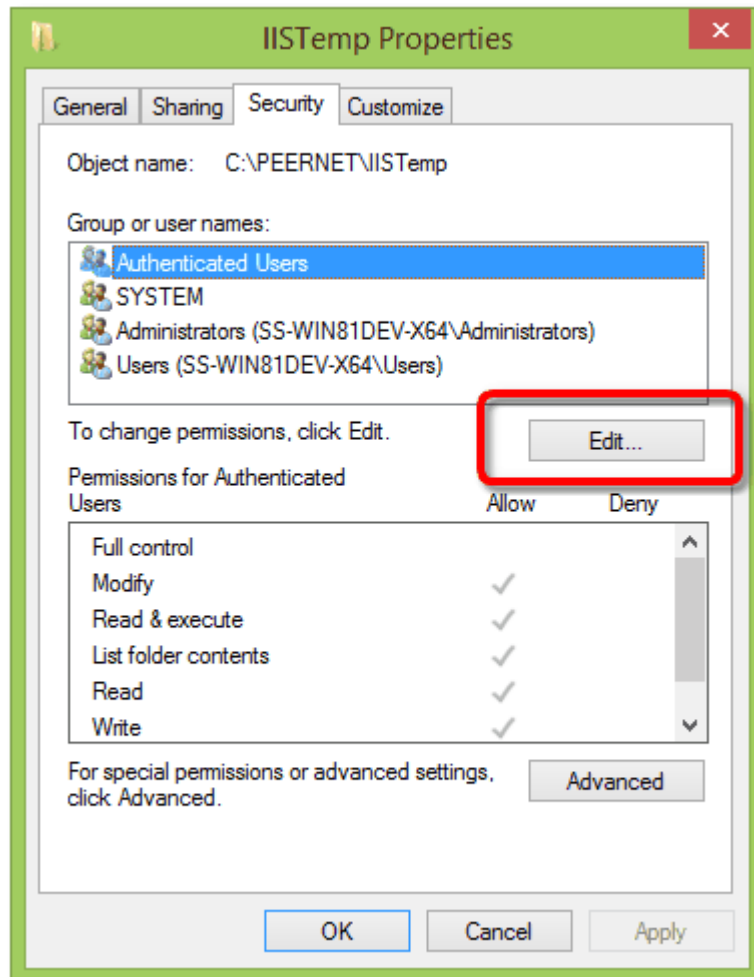
When calling PEERNET.ConvertUtility methods or the command line conversion utilities from within a web service, the Document Conversion Service Users local group sometimes needs to be added to certain folders to give the conversion process the required permissions to access the folders.

One such scenario is if you are passing a custom folder for the *ConversionWorkingFolder*, this Document Conversion Service Users group needs Full Control, or at least Modify access on that folder. Not having this level of access will cause the conversion process to take upwards of an extra 90 seconds to complete as the utility attempts to clean up interim files and folders created as part of the conversion. Once access is granted, the cleanup is instant.

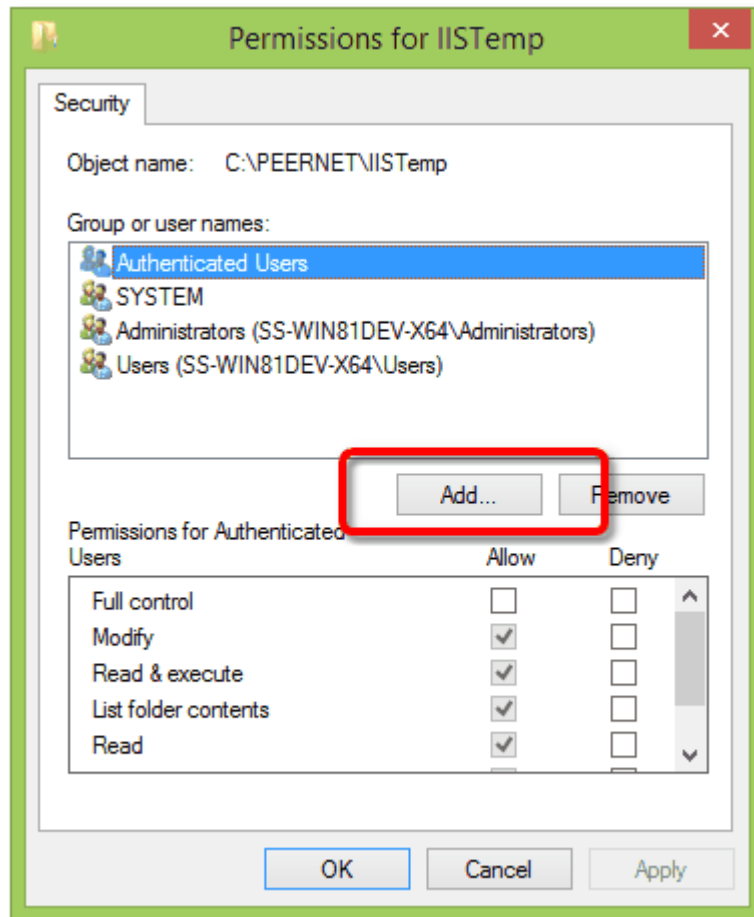
Other folders that may need permissions include the input and output folders, custom paths for results files and the SIL logging files.

1. On the folder you want to use as the *ConversionWorkingFolder*, right-click and select **Properties** from the context menu.

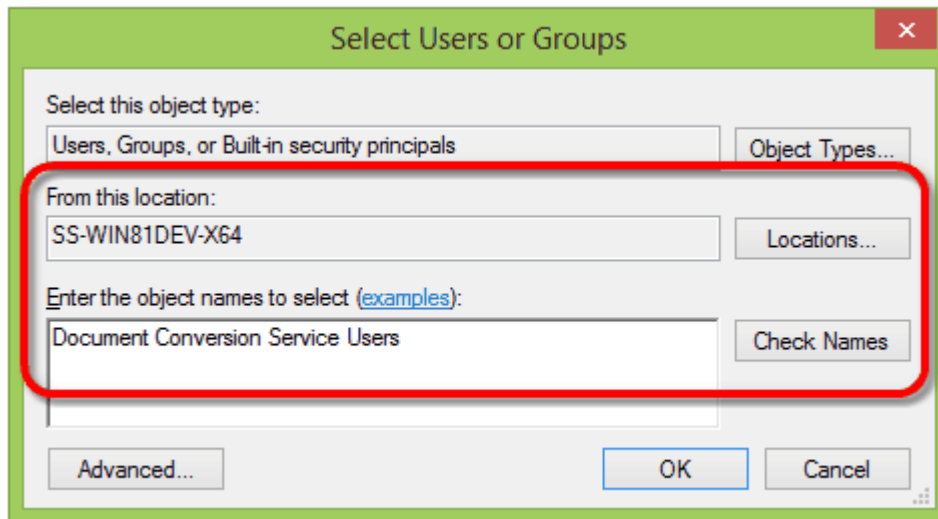
2. On the *Properties* dialog box, select the **Security** tab and then the **Edit...** button to open the Permissions dialog.



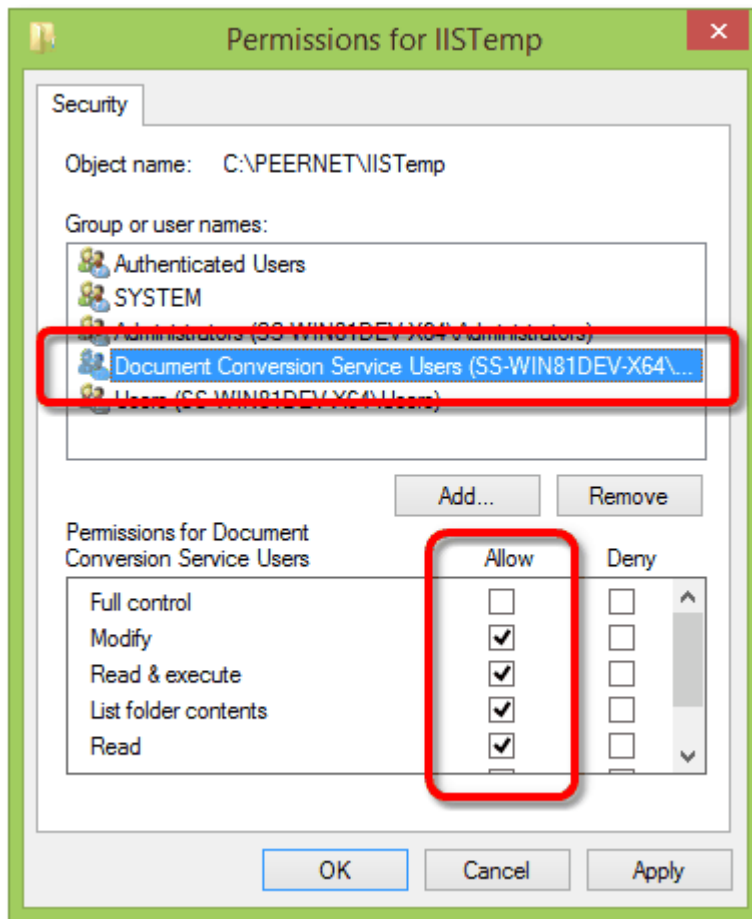
3. On the *Permissions* dialog, click the **Add...** button.



4. In the *Select Users or Groups* dialog change the **From this location:** to be the local computer (it normally defaults to the domain if you are on one). Then add the Document Conversion Service Users group in the list at the bottom. When done, press the **OK** button.



- Back on the *Permissions* dialog, make sure the new group Document Conversion Service Users is selected. In the permissions section below under the **Allow** column, make sure there is a check mark in the *Modify* option. You can also check *Full control* to grant the group full access.



- Click **OK** to apply the changes.

Creating and Customizing Profiles

Document Conversion Service includes several sample profiles for common types of output files for your use. The default set of profiles are installed into the following location:

Default profile location:

C:\ProgramData\PEERNET\Document Conversion Service\Profiles

Custom Profiles

You can use the sample profiles above as a base to edit and create your own custom profiles. Custom profiles can be stored per user in the user's application data folder. Both the local and roaming data folders are searched when looking for user profiles. If a profile is found in a user location, that profile will be used. If no matching profiles are found in the user profile locations, the default profile location is searched.

User profile locations searched in this order:

C:\Users\<user>\AppData\Roaming\Document Conversion Service\Profiles
C:\Users\<user>\AppData\Local\Document Conversion Service\Profiles

When using the PEERNET.ConvertUtility.dll and the command line tools, the full path to a profile stored elsewhere on disk can also be passed instead of the base name of the profile.

See the section [Conversion Settings](#) for information on the contents and structure of the profile files, and the [Name-Value Tables for Conversion Settings](#) for the conversion setting strings to use to get various output formats.

Included Sample Profiles

The profiles included with the Document Conversion Service install are listed below.

See below for [E-discovery specific profiles](#), and [Optical Character Recognition \(OCR\) profiles](#).

Profile Name	Profile Description
Adobe PDF Multipage	Creates Adobe PDF files. The PDF files created using this profile are, where possible, <i>vector</i> PDF files. Vector PDF files are also known as <i>searchable</i> PDF files. The other PDF profiles provided create <i>raster</i> , or non-searchable PDF files. What this profile cannot do is create a vector PDF from an existing raster PDF (scanned PDF) or other image formats such as TIFF or JPEG. A vector PDF is only created if the source document contains text or vector graphics already.
Adobe PDF To RTF	Creates a Rich Text Format document from a PDF file. Does not work with other input file types. only PDF documents.
BMP 100dpi Color	Creates Windows Bitmap images (one image for each page) at 100dpi. Bitmap images are always serialized.

Profile Name	Profile Description
JPEG 60dpi Color JPEG 120dpi Color JPEG 200dpi Color JPEG 300dpi Color JPEG 600dpi Color	Creates color JPEG images (one image for each page) at the dots per inch (dpi) specified. JPEG files are always serialized.
PDF 200dpi OptimizedColor Serialized PDF 300dpi OptimizedColor Serialized	Creates serialized (one file per page) PDF documents at the dots per inch (dpi) specified. Color is optimized per page to reduce file size.
PDF 200dpi OptimizedColor PDF 300dpi OptimizedColor	Creates a multipaged PDF document at the dots per inch (dpi) specified. Color is optimized per page to reduce file size.
PDF A-1b 200dpi OptimizedColor Serialized PDF A-1b 300dpi OptimizedColor Serialized	Creates serialized (one file per page) PDF/A-1b compliant PDF documents at the dots per inch (dpi) specified. Color is optimized per page to reduce file size.
PDF A-1b 200dpi OptimizedColor PDF A-1b 300dpi OptimizedColor	Creates a multipaged PDF/A-1b compliant PDF document at the dots per inch (dpi) specified. Color is optimized per page to reduce file size.
Text to Adobe PDF Multipage Text to TIFF 120dpi Monochrome	Profiles for use when converting text files to a searchable PDF or TIFF. Uses the Text - Builtin converter. Page size is auto-detected; see Built-in Text Converter Options for all text settings. Works for all text sizes including wide format (landscape oriented) text files such as those generated on mainframe systems or other reporting systems.
Text to A3 sized TIFF 120dpi Monochrome Text to A3 sized PDF 120dpi Monochrome	<i>Deprecated; use Text to Adobe PDF Multipage or Text to TIFF 120dpi Monochrome with the Text - Builtin converter instead.</i> Profiles for use when converting text files using Word to a specific size of paper. These profiles target wide format (landscape oriented) text files such as those generated on mainframe systems or other reporting systems.
TIFF 120dpi Color LowJPEG TIFF 150dpi Color LowJPEG TIFF 200dpi Color LowJPEG TIFF 300dpi Color LowJPEG TIFF 600dpi Color LowJPEG	Creates multipaged color TIFF images at the dots per inch (dpi) specified. Images are compressed using low quality JPEG compression. This can give a smaller file size but a lower quality image.
TIFF 120dpi Color HighPEG TIFF 150dpi Color HighPEG TIFF 200dpi Color HighPEG TIFF 300dpi Color HighPEG TIFF 600dpi Color HighPEG	Creates multipaged color TIFF images at the dots per inch (dpi) specified. Images are compressed using high quality JPEG compression. This can give a higher quality image but also a larger size file.

Profile Name	Profile Description
TIFF 120dpi Grayscale TIFF 150dpi Grayscale TIFF 200dpi Grayscale TIFF 300dpi Grayscale TIFF 600dpi Grayscale	Creates multipaged grayscale TIFF images at the dots per inch (dpi) specified.
TIFF 120dpi OptimizedColor TIFF 150dpi OptimizedColor TIFF 200dpi OptimizedColor TIFF 300dpi OptimizedColor TIFF 600dpi OptimizedColor	Creates a single multipage TIFF image at the dots per inch (dpi) specified. Color is optimized per page to reduce file size. File is compressed using Group 4 compression for monochrome and LZW for all other color types.
TIFF 200dpi OptimizedColor HighJPEG	Creates a single multipage TIFF image at the dots per inch (dpi) specified. Color is optimized per page to reduce file size. File is compressed using Group 4 compression for monochrome and high quality JPEG compression for all other color types.
TIFF 200dpi Monochrome Serialized	Creates serialized (one file per page) black and white TIFF images at 200dpi.
TIFF 200dpi Monochrome	Creates a single multipage black and white TIFF image at 200dpi.
TIFF 204x196dpi Monochrome Fax	Creates a single multipage black and white fax format TIFF image at 204 x 196dpi.
TIFF 204x196dpi Monochrome Fax ReverseBitOrder	Creates a single multipage black and white Group 4 fax format TIFF image at 204 x 196dpi with a reverse bit order of least significant bit to most significant bit (LSB2MSB). Often needed for fax boards.
TIFF 204x196dpi Monochrome Fax Group3 256GreyPalette	Creates a single multipage Group 3 fax format TIFF image at 204 x 196dpi using a grayscale palette.
TIFF 204x196dpi Monochrome Fax Group3 256GreyPalette ReverseBitOrder	Creates a single multipage Group 3 fax format TIFF image at 204 x 196dpi using a grayscale palette with a reverse bit order of least significant bit to most significant bit (LSB2MSB).
TIFF 204x196dpi Monochrome Fax Compatible with FCC	Created fax TIFF images matching the format created by the Fax(TIFF) profile used in PEERNET File Conversion Center. Provided for use by clients migrating from File Conversion Center to Document Conversion Service.
TIFF 300dpi Allow Javascript PDF	This profile is the same as the <i>TIFF 300dpi Optimized Color</i> above but also enables the processing of Javascript, if present, in PDF files when they are converted using this profile.
TIFF 300dpi Color Fax	Creates a single multipage color fax format TIFF image at 300dpi.

Profile Name	Profile Description
TIFF 300dpi OptimizedColor ExtractText Serialized	Creates serialized (one file per page) TIFF images at 300dpi. Color is optimized per page to reduce file size. Text content, if available, is extracted and saved as separate files with the same base name as the output images.
TIFF 300dpi OptimizedColor ExtractText	Creates a single multipage TIFF image at 300dpi. Color is optimized per page to reduce file size. Text content, if available, is extracted and saved as a separate file with the same base name as the output image.
TIFF 300dpi OptimizedColor Serialized	Creates serialized (one file per page) TIFF images at 300dpi. Color is optimized per page to reduce file size.
TIFF 300dpi OptimizedColor SplitByPageCount	Creates a sequence of multipaged 300 dots per inch TIFF images. A new file in the sequence is started based on the page count set by the <i>SplitFileEveryNPages</i> setting. When auto-splitting files, serialized naming profile is always used to name each file in the sequence.
TIFF 300dpi OptimizedColor SplitByFileSize	Creates a sequence of multipaged 300 dots per inch TIFF images. A new file in the sequence is started when the current file exceeds the file size set by the <i>SplitFileSizeThresholdInBytes</i> setting. When auto-splitting files, serialized naming profile is always used to name each file in the sequence.
Text to A3 sized TIFF 120dpi Monochrome Text to A3 sized PDF 120dpi Monochrome	Profiles for use when converting text files in Word to a specific size of paper. These profiles target wide format (landscape oriented) text files such as those generated on mainframe systems or other reporting systems.

Optical Character Recognition (OCR) Profiles	Profile Description
<p>Adobe PDF OCR to Searchable Adobe PDF OCR to Searchable Serialized</p>	<p>These profiles can be used to perform OCR when converting to PDF with any of the PEERNET built-in converters installed with Document Conversion Service. Optical Character Recognition (OCR for short), searches for and recognizes text (characters) on scanned pages or images and extracts it as digital text included in the PDF file.</p>
E-Discovery Profiles	Profile Description
<p>eDiscovery - Excel - PDF 300dpi Convert Charts Only eDiscovery - Excel - TIFF 300dpi Convert Charts Only</p>	<p>For use with Excel documents, these profiles will print only the embedded charts and any chart tabs in the document.</p>
<p>eDiscovery - Excel - PDF 300dpi Show Formulas eDiscovery - Excel - TIFF 300dpi Show Formulas</p>	<p>For use with Excel documents, these profiles will print any formulas from any cells as a comment at the end of each sheet. If a comment already exists, the formula is inserted before the existing text. For Excel documents, a tracked changes history sheet is created if tracking is enabled, background colors are removed, text is changed to black and conditional formatting is removed.</p>
<p>eDiscovery PDF 300dpi AutoField Replace eDiscovery TIFF 300dpi AutoField Replace</p>	<p>For use with Word, Excel and PowerPoint e-discovery, these profiles will show all data in the documents and where possible, replace any auto data, time and file fields in headers, footers, and in the case of Excel, in cells too. For Excel documents, a tracked changes history sheet is created if tracking is enabled, background colors are removed, text is changed to black and conditional formatting is removed.</p>
<p>eDiscovery PDF 300dpi Monochrome Fit On Page eDiscovery TIFF 300dpi Monochrome Fit On Page</p>	<p>For use with Word, Excel and PowerPoint e-discovery, these profiles will show all data in the documents. The output created is black and white. For Excel documents, each sheet is fit to a single output page, a tracked changes history sheet is created if tracking is enabled, background colors are removed, text is changed to black and conditional formatting is removed.</p>
<p>eDiscovery PDF 300dpi Span Pages eDiscovery TIFF 300dpi Span Pages</p>	<p>For use with Word, Excel and PowerPoint e-discovery, these profiles will show all data in the documents. For Excel documents, tracked changes history sheet is created if tracking is enabled, background colors are removed, text is changed to black and conditional formatting is removed.</p>

Conversion Settings

Conversion settings are used to describe the output created by Document Conversion Service and consist of a collection of name-value pairs. These settings can also be used to control the behavior of the individual converters, such as configuring Word to pass a password or telling Excel to ignore the print areas when printing worksheets. The technique you are using to convert your files (command line utilities, the PEERNET.ConvertUtility.dll .NET library or the PNDocConvQueueServiceLib COM object) will determine how you will pass this collection of settings to Document Conversion Service.

Command Line Utilities

When using the command line utilities the settings are passed by supplying the name of a *profile file*, a structured XML file that contains the list of settings. Below is a sample command line using a profile file named *TIFF 300dpi Optimized Color.xml*, followed by a listing of the XML file itself. Note that the *.xml* extension is not needed when using the command line utilities. Several sample profiles are included for your use, or to use as a base to [customize](#) to your needs.



Passing setting using a profile

```
DCSConvertFile /P="TIFF 300dpi OptimizedColor" /NE "C:\Test\File.pdf"
```



Sample Profile

```
<?xml version="1.0" encoding="utf-8"?>
<Profile Type="0"
  DisplayName="TIFF 300dpi OptimizedColor"
  Description="Creates a single TIFF image at 300dpi.">
  <Settings>

    <!-- Output file options -->
    <add Name="Devmode settings;Resolution" Value="300"/>
    <add Name="Save;Output File Format" Value="TIFF Multipaged"/>
    <add Name="Save;Color reduction" Value="Optimal"/>
    <add Name="Save;Dithering method" Value="Halftone"/>

    <!-- TIFF Compression Options -->
    <add Name="TIFF File Format;BW compression" Value="Group4"/>
    <add Name="TIFF File Format;Color compression" Value="LZW RGB"/>
    <add Name="TIFF File Format;Indexed compression" Value="LZW"/>
    <add Name="TIFF File Format;Greyscale compression" Value="LZW"/>
    <add Name="JPEG File Format;Color compression" Value="Medium Quality"/>
    <add Name="JPEG File Format;Greyscale compression" Value="High Quality"/>
    <add Name="Image Options;Fax" Value="0"/>

  </Settings>
</Profile>
```

PEERNET.ConvertUtility .NET Library

When using the PEERNET.ConvertUtility .NET library methods from your own managed code you have the choice of supplying the name of a *profile file*, an XML file that contains the list of settings, or by passing in an *IDictionary<String,String>* collection of name-value pairs directly. Several sample profiles are included for your use, or to use as a base to [customize](#) to your needs.

**Code Sample - Calling ConvertFile with a Profile**

```
using PEERNET.ConvertUtility;

// conversion results returned, use to find files created or errors
PNConversionItem resultItem =
    PNConverter.ConvertFile(@"C:\Input\Memo.doc",
        @"C:\Output\",
        "ConvertedMemo",
        true, // overwrite existing
        false, // remove file extension
        false, // create log file
        "TIFF 300dpi OptimizedColor", // profile
        String.Empty,
        String.Empty,
        null, // no extra settings
        String.Empty, //remote computer
        String.Empty);
```

**Code Sample - Calling ConvertFile with a settings collection**

```
using PEERNET.ConvertUtility;

IDictionary<String, String> settings =
    new Dictionary<String, String>();

settings.Add("Devmode settings;Resolution", "300");
settings.Add("Save;Output File Format", "TIFF Multipaged");
settings.Add("Save;Color reduction", "Optimal");
settings.Add("Save;Dithering method", "Halftone");

// conversion results returned, use to find files created or errors
PNConversionItem resultItem =
    PNConverter.ConvertFile(@"C:\Input\Memo.doc",
        @"C:\Output\",
        "ConvertedMemo",
        true, // overwrite existing
        false, // remove file extension
        false, // create log file
        settings,
        String.Empty,
        String.Empty,
        null, // no extra settings
        String.Empty, //remote computer
        String.Empty);
```

PNDocConvQueueServiceLib COM Object

The PNDocConvQueueServiceLib COM object uses a list of name-value pairs of conversion settings to configure the output that is created. These settings are passed into the COM object directly through its `IPNDocConvQueueItem.Set` method before calling `IPNDocConvQueueItem.Convert`.

The following code sample show the conversion settings strings for setting the resolution to 200 DPI and creating multipaged black and white TIFF files. The Resolution setting is part of the [Devmode settings](#) configuration options, while Output File Format , Append, Color reduction, and Dithering method are part of the [Save](#) configuration options.

You can find more sample output configurations by looking at the name and value pairs used in the sample [conversion profiles](#) included with Document Conversion Service.



Code Sample - C#

```
PNDocConvQueueServiceLib.PNDocConvQueueItem item = null;

// Create the conversion item
item = new PNDocConvQueueServiceLib.PNDocConvQueueItem();

// Set conversion settings
item.Set("Devmode settings;Resolution", "200");
item.Set("Save;Output File Format", "TIFF Multipaged");
item.Set("Save;Color reduction", "BW");
item.Set("Save;Dithering method", "Floyd");
item.Set("TIFF File Format;BW compression", "Group4");
item.Set("TIFF File Format;Color compression", "LZW RGB");
item.Set("TIFF File Format;Indexed compression", "LZW");
item.Set("TIFF File Format;Greyscale compression", "LZW");

// convert the file
item.Convert("Microsoft Word",
            "C:\\Test\\Report.docx",
            "C:\\Test\\Out\\ConvertedReport");
```



Code Sample - VB.NET

```
Dim item As PNDocConvQueueServiceLib.IPNDocConvQueueItem

' Create the conversion item
item = New PNDocConvQueueServiceLib.PNDocConvQueueItem()

' Set conversion settings
item.Set("Devmode settings;Resolution", "200")
item.Set("Save;Output File Format", "TIFF Multipaged")
item.Set("Save;Color reduction", "BW")
item.Set("Save;Dithering method", "Floyd")
item.Set("TIFF File Format;BW compression", "Group4")
item.Set("TIFF File Format;Color compression", "LZW RGB")
item.Set("TIFF File Format;Indexed compression", "LZW")
item.Set("TIFF File Format;Greyscale compression", "LZW")

' convert the file
item.Convert("Microsoft Word", _
            "C:\\Test\\Report.docx", _
            "C:\\Test\\Out\\ConvertedReport")
```

Name-Value Tables for Conversion Settings

The table below lists the different conversion settings separated out into categories with a description of the settings available in each. Click the link for that category to view all available settings for that option.

Options	Description of Settings
General Converter Options	These are general options that can be applied to the conversion process itself or to all converters.
Built-in Converter OCR Options	These options can be used to perform OCR when converting to PDF with any of the PEERNET built-in converters installed with Document Conversion Service. Optical Character Recognition (OCR for short), searches for and recognizes text (characters) on scanned pages or images and extracts it as digital text.
Built-in Cadd Converter Options	Convert DWF, DWFX, PLT and GBX files using the builtin Cadd conversion utility. This converter does not require any third-party applications.
Built-in Image Converter Options	The built-in image converter is a faster, more efficient converter designed to replace our original Image Converter. This converter has no options.
Built-in PDF Converter Options	Convert PDF files using the built-in PDF conversion utility instead of Adobe Reader. This converter has no options.
Built-in Text Converter Options	Convert text files using this built-in conversion utility instead of Microsoft Word. This converter has options for font name and size. It auto-detects page size but also offers font and page size settings.
Word Converter Options	These options are specific to the behavior of the Word converter.
Excel Converter Options	These options are specific to the behavior of the Excel converter.
PowerPoint Converter Options	These options are specific to the behavior of the PowerPoint converter.
Ghostscript Converter Options	These options are specific to the behavior of the Ghostscript converter.
Image Converter Options	These options are specific to the behavior of the Image converter.

Options	Description of Settings
OutsideIn AX Options	These options are specific to the behavior of the OutsideIn converter.
Advanced Features	Advanced settings such as custom paper size and text extraction.
Advanced File Naming	Settings to configure the file naming profiles (preset file naming schemes) for multipaged, multipaged with JobID, serialized and serialized with JobID.
Devmode settings	Resolution (DPI), page size and color mode settings.
Image Options	Image output options such as creating fax mode images and page rotation settings.
JPEG File Format	Compression settings for color and greyscale JPEG images.
PDF File Format	PDF file format settings for compression, content encoding and PDF/A-1b compliant PDF files.
PDF Security	PDF encryption and file permissions.
Processing	Settings to adjust the image during conversion such as trimming, cropping, copying to a new page size, resampling and brightness adjustment.
Save	Settings for output file format, color reduction, dithering and file name prompting.
TIFF File Format	Compression settings for black and white, color, indexed and greyscale TIFF images.
Endorsement Options	Endorsements are header and footer information that can be stamped onto each page of the output created by Document Conversion Service.
Watermark Stamping	Settings to create a text watermark diagonally across the page.

File Extension to Converter Mapping

The file extension of each file is used to determine what converter is used when Document Conversion Service converts that file.

When using the PEERNET.ConvertUtility.dll or the command line tools to convert files, a default file extension mapping profile, *File Extension To Converter Map.xml*, is used to determine this mapping. This file can be edited and file extensions can be added, removed and changed as needed.

If desired, the file itself can be copied and renamed and the new mapping file passed to the PEERNET.ConvertUtility methods or the command line tools as needed.

A simpler approach is to customize the file extension mapping by adding the setting into a profile file. This allows you to set the file extension mapping at a file level instead of at the application level. Any file extension mappings found in a profile will override the settings in the base *File Extension To Converter Map.xml* file.

A common use of this would be to have a profile that uses the PEERNET Passthrough Converter to skip processing TIFF files, or one that uses Ghostscript to process PDF files instead of Adobe Reader.

For the Watch Folder Service, the service's configuration file contains its own file extension to converter mapping section. The extension to converter mapping listed in the configuration file has the same format as in the mapping profile.

Customizing the File Extension Mapping Profile

File mapping profiles are stored in the same location as the conversion profiles. The default file extension mapping profile, *File Extension To Converter Map.xml*, is installed as part of Document Conversion Service. The difference between a conversion profile and a mapping profile is detected using the Type attribute on the Profile element. It is 0 for a conversion profile and 1 for a file extension mapping profile.

The mapping consists of the extension (the suffix of the file name past the last dot or period in file's name) and a semi-colon separated list of converter names. There are two things to remember when modifying this file:

1. Each file extension can only be listed once.
2. The file extensions must be added in lower case and must include both the dot (.) and the extension.

In some cases the file extension may only have one converter associated with it. Others, such as PDF which can be converted using either our Adobe PDF - Builtin converter, Adobe Reader, Ghostscript or Outside-In AX, can potentially have more than one converter, in order of preference, associated with it. The code sample below shows a small snippet of the file mapping in the provided file mapping profile.

You can edit the File Extension Mapping profile in the DCS Editor by going to **DCS Dashboard - DCS Settings - Edit File Extension Map**. You can also open the XML file using your favorite text or code editor.



Code Sample - File Extension to Converter Mapping

```
<?xml version="1.0" encoding="utf-8"?>
<Profile Type="1"
  DisplayName="File Extension To Converter Map"
  Description="Maps file extensions to the converter to use for that document.">
  <Settings>
    <add Name=".doc" Value="Microsoft Word;Outside-In AX"/>
    <add Name=".docx" Value="Microsoft Word;Outside-In AX"/>
    ...
    <add Name=".xlsx" Value="Microsoft Excel;Outside-In AX"/>
    <add Name=".xlsm" Value="Microsoft Excel;Outside-In AX"/>
    ...

    <add Name=".pdf" Value="Adobe PDF - Builtin;Adobe Acrobat Reader;Ghostscript;Outside-In
    ...
  </Settings>
</Profile>
```

The table below lists the available converters and their default file extensions.

Converter Name	Supported Document Types
Adobe Acrobat Reader	Adobe PDF Documents (*.pdf)
Adobe PDF - Builtin	Adobe PDF Documents (*.pdf)
Autodesk Design Review	Design Review Drawings (*.dwf) AutoCAD Drawings (*.dwg)
Microsoft Excel	Excel Workbooks (*.xlsx, *.xlsm, *.xls) Excel Templates (*.xltx, *.xltm, *.xlt) Excel Binary Workbook (*.xlsb)
Cadd - Builtin	Design Review Drawings (*.dwf, *.dwfx) HPGL Plot files (*.plt) Gerber RS-274D, Gerber RS-274X, Gerber X2 (*.gbr)
Ghostscript	Postscript Files (*.ps) Encapsulated Postscript Files (.eps) Adobe PDF Documents (*.pdf)
Image - Builtin (replaces PEERNET Image Converter and PEERNET WIC Image Converter)	JPEG images (*.jpg) TIFF images (*.tif) High Efficiency Image Files (*.heif, *.heic) Google WebP Images (*.webp) AVIF Images (*.avif) Windows Bitmap images (*.bmp) ZSoft PCX images (*.pcx) ZSoft DCX images (*.dcx) Cserve Portable Network Graphics images (*.png) Graphics Interchange Format image files (*.gif) Icon Format (*.ico) Windows Media Photo images (*.wdp, *.hdp, *.jxr) ImageMagick images (100+ image formats)
PEERNET Image Converter	JPEG images (*.jpg) TIFF images (*.tif) Windows Bitmap images (*.bmp) ZSoft PCX images (*.pcx) ZSoft DCX images (*.dcx) Cserve Portable Network Graphics images (*.png) Graphics Interchange Format image files (*.gif) Icon Format (*.ico) Windows Media Photo images (*.wdp, *.hdp, *.jxr)
PEERNET WIC Image Converter	Icon Format (*.ico) Windows Media Photo images (*.wdp, *.hdp, *.jxr) Works with other Windows Imaging Component (WIC) third-party add-ons such as: DjVu Shell Extension Pack (*.djvu) FastPicture Viwer Codec Pack adds support for over 45+ image formats and over 500 raw digital camera formats

Converter Name	Supported Document Types
Internet Explorer	HTML Files (*.htm, *.html) Secure HTML (*.shtm, *.shtml) Web Archive (*.mht)
Microsoft Outlook	Outlook Message Files (*.msg) Outlook Templates (*.oft)
Outside-In AX	Oracle Outside In Viewer Technology (ActiveX) supports over 500 common file formats; see the documentation that came with your Outside In Technology product.
Microsoft PowerPoint	PowerPoint Presentations (*.pptx, *.pptm, *.ppt) PowerPoint Shows (*.ppsx, *.ppsm, *.pps) PowerPoint Templates (*.potx, *.potm, *.pot)
Microsoft Publisher	Publisher Files (*.pub)
Text - Builtin	Text Files (*.txt) Logging Files (*.log) Source Files(*.c, *.cpp, *.cs, etc.) Config and INI Files (.config, *.ini) PowerShell (*.ps1) Batch Files (*.cmd, *.bat) Any other text-based file
Microsoft Visio	Visio Drawings (*.vsd)
Microsoft Word	Word Documents (*.docx, *.docm, *.doc) Word Templates (*.dotx, *.dotm, *.dot) Rich Text Documents (*.rtf) Plain Text Files (*.txt) Plain Text Log Files (*.log)
Microsoft XPS	XPS Documents (*.xps) Open XPS Documents (*.oxps)
PEERNET Passthrough	Any file type. Passes any file matching the extension through the system without converting.

General Converter Options

These options can be used with any of the converters installed with Document Conversion Service. Table values in **bold** text are the default value for that setting.



Sample Profile

```
<?xml version="1.0" encoding="utf-8"?>
<Profile Type="0"
    DisplayName="TIFF 300dpi First3PagesOnly "
    Description ="Prints only the first three pages.">
  <Settings>

    <!-- Print first three pages only -->
    <add Name="PageRange" Value="1-3"/>

    <!-- Output file options -->
    <add Name="Devmode settings;Resolution" Value="300"/>
    <add Name="Save;Output File Format" Value="TIFF Multipaged"/>
    ...

  </Settings>
</Profile>
```



Code Sample - C#

```
PNDocConvQueueServiceLib.PNDocConvQueueItem item = null;

// Create the conversion item
item = new PNDocConvQueueServiceLib.PNDocConvQueueItem();

// Set conversion settings
item.Set("PageRange", "1-3");
item.Set("Devmode settings;Resolution", "300");
item.Set("Save;Output File Format", "TIFF Multipaged");
...
// convert the file
item.Convert("Microsoft Word",
    @"C:\Test\Report.docx",
    @"C:\Test\Out\ConvertedReport");
```



Code Sample - VB.NET

```
Dim item As PNDocConvQueueServiceLib.IPNDocConvQueueItem

' Create the conversion item
item = New PNDocConvQueueServiceLib.PNDocConvQueueItem()

' Set conversion settings
item.Set("PageRange", "1-3")
item.Set("Devmode settings;Resolution", "300")
item.Set("Save;Output File Format", "TIFF Multipaged")
...
' convert the file
item.Convert("Microsoft Word", _
    "C:\Test\Report.docx", _
    "C:\Test\Out\ConvertedReport")
```

Conversion Settings

Name: **PageRange**

The page numbers and page ranges to include in the output file. Separate each number and range with a comma. For example, "1, 3, 5-7" prints page 1 and 3 and pages 5 through 7. Numbers in the page range exceeding the page count of the source document are ignored.

Values: The string representing the page range.

Name: **MaxSpooledPagesAllowed**

Sets the maximum number of pages that are allowed to be printed/spooled. Documents larger than this set page limit will not convert.

Values: The string representing the maximum number of pages allowed.

Name: **MaxSpooledPagesGreaterThanPageCount**

Sets the maximum number of spooled pages greater than the document page count that is allowed to be printed. Documents larger than this set page limit will not convert. This is often used to manage a single extra page created by duplex printing forced by the document. It can also occur with mail merge documents and PDF files that use Javascript. For PDF files, use the **Adobe.PDF.Javascript.Enable** setting instead.

When not set (default), or set to 0, the conversion will fail if the number of spooled pages is greater than the document page count.

Values: The string representing the maximum number of pages allowed.

Name: **NormalizeFileNames**

When *true*, file names passed in will be checked for normalization and normalized when necessary. This means that the new output file name, if not specified, will be the normalized filename.
The default is to not normalize the filename.

This is needed for foreign file name where some international characters are represented using diacritics. A diacritic is a *glyph* added to a letter; they are used to change the sound of the letter to which they are added. Some examples of a diacritic are the accent grave (') and acute (') in the French language.

Values: Pass *true* to normalize file names if necessary.

Conversion Settings

Name: **SecondsToWaitForRunningConversionService**

Applies only when using the command line tools (/D switch) and the PEERNET.ConvertUtility methods.

The Document Conversion Service must be running, either locally or on a remote computer for files or folders of files to be converted. If it is not running the PEERNET.ConvertUtility methods or command line tools it will all return immediately with an error. To wait for Document Conversion Service to be running instead of failing to convert the files, use this setting to pass the desired wait timeout value down. If Document Conversion Service hasn't started after waiting the supplied amount of time, an error is returned.

Values: The number of seconds to wait for Document Conversion Service to be running and ready to convert files.

Name: **KeepFailedItemResultsFiles**

Applies only when using the command line tools (/D switch) and when passing custom settings to the PEERNET.ConvertUtility methods.

By default when a conversion fails, a results file ending with .failed.dcsresults for the file that failed will be created in a .failed folder. To suppress the automatic creation of these files pass this setting as *true*. When using the PEERNET.ConvertUtility methods, the resultant items that are returned will contain the path to the results file.

Values: Pass *true* to suppress the creation of these files.

Name: **FailedFolder**

Applies only when passing custom settings to the PEERNET.ConvertUtility methods.

By default when a conversion fails, a results file ending with .failed.dcsresults for the file that failed will be created in a .failed folder. Specifying a folder for this custom setting will override the default use of the .failed folder and store the failed results log files if the specified folder.

Values: Pass the path to the folder in which to store the failed conversion results files.

Conversion Settings

Name: AlwaysKeepProcessingLoggingFiles

Applies only when using the command line tools (/D switch) and the PEERNET.ConvertUtility methods.

By default a Smart Inspect console logging file (*.sil) is always created when a conversion runs. If the conversion is successful, the log file is normally deleted. If it fails, it is kept and copied to the Windows temp folder. To always keep this file, pass this setting as true. Overrides the variable

KeepFailedProcessingLoggingFiles. When using the PEERNET.ConvertUtility methods, the results items that are returned will contain the path to the results file.

Values: Pass *true* to always keep the logging file.

Name: KeepFailedProcessingLoggingFiles

Applies only when using the command line tools (/D switch) and the PEERNET.ConvertUtility methods.

By default when a conversion fails, the Smart Inspect console logging file (*.sil) created as part of the conversion process is kept and copied to the Windows temp folder. To have these files deleted even when the conversion fails, pass this setting as *true*. When using the PEERNET.ConvertUtility methods, the results items that are returned will contain the path to the results file.

Values: Pass *true* to delete these files when the conversion is finished even if the conversion has failed.

Name: UseCompressedDateTimeFormat

Applies only when using the command line tools (/D switch) and the PEERNET.ConvertUtility methods.

Controls the formatting of the name of the date and time subfolder used internally by the conversion utility in the staging and working folders for file conversion, as well as in naming the internal logging files (*.sil). This setting would only need to be altered if you are dealing with very long folder and file path names that exceed the 255 character path limit, as a way of reducing the internally created paths so that they do not exceed the maximum path length.

When set to FALSE, or not provided, the folder name follows the pattern '2016_03_31_2_38_46_PM'. The compressed format is shorter, and uses a 24-hour time format, giving a folder following the pattern '20160331143846'.

Values: Pass *true* to use the shorter, numerical format.

Built-in Converter OCR Options

These options are used when converting to PDF with any of the PEERNET built-in converters installed with Document Conversion Service. These settings do not apply to any other output format.



Caution

This feature is not supported on Microsoft® Windows Server 2008 R2 and Microsoft® Windows 7.

Optical Character Recognition (OCR) for short), searches for and recognizes text (characters) on scanned pages or images and extracts it as digital text. When recognizing text, the OCR engine has to know which languages to look for on the page. OCR works by analyzing the patterns, shapes, and curves of the text characters on the page and matching them to predefined information for different characters in each language.

OCR will increase the processing time for file conversion. Outside factors such as image quality, the font used, and any image background on the pages will all affect the validity of the OCR results.

They are used by the following converters:

- Built-in PDF Converter
- Built-in Image Converter
- Built-in Cadd Converter

Table values in **bold** text are the default value for that setting.



Sample Profile

```
<?xml version="1.0" encoding="utf-8"?>
<Profile Type="0"
  DisplayName="Adobe PDF OCR Searchable"
  Description="Converts to OCR (searchable) PDF.">
  <Settings>

    <add Name ="ConverterPlugIn.PNBuiltinsOCRPDF.Enabled" Value="1"/>
    <add Name ="ConverterPlugIn.PNBuiltinsOCRPDF.Languages" Value="eng+fra"/>
    <add Name ="ConverterPlugIn.PNBuiltinsOCRPDF.FirstPageOnly" Value="0"/>

    <!-- Output file options -->
    <add Name="Devmode settings;Resolution" Value="300"/>
    <add Name="Save;Output File Format" Value="Adobe PDF Multipaged"/>
    ...

  </Settings>
</Profile>
```


**Code Sample - C#**

```
PNDocConvQueueServiceLib.PNDocConvQueueItem item = null;

// Create the conversion item
item = new PNDocConvQueueServiceLib.PNDocConvQueueItem();

// Set conversion settings
item.Set("ConverterPlugIn.PNBuiltinsOCRPDF.Enabled", "1");
item.Set("ConverterPlugIn.PNBuiltinsOCRPDF.Languages", "eng+fra");
item.Set("ConverterPlugIn.PNBuiltinsOCRPDF.FirstPageOnly", "0");

item.Set("Devmode settings;Resolution", "300");
item.Set("Save;Output File Format", "Adobe PDF Multipaged");
...
// convert the file
item.Convert("Cadd - Builtin", _
            @"C:\Test\BuildingPlan.dwf", _
            @"C:\Test\Out\ConvertedDrawings");
```

**Code Sample - VB.NET**

```
Dim item As PNDocConvQueueServiceLib.IPNDocConvQueueItem

' Create the conversion item
item = New PNDocConvQueueServiceLib.PNDocConvQueueItem()

' Set conversion settings
item.Set("ConverterPlugIn.PNBuiltinsOCRPDF.Enabled", "1")
item.Set("ConverterPlugIn.PNBuiltinsOCRPDF.Languages", "eng+fra")
item.Set("ConverterPlugIn.PNBuiltinsOCRPDF.FirstPageOnly", "0")

item.Set("Devmode settings;Resolution", "300")
item.Set("Save;Output File Format", "Adobe PDF Multipaged")
...
' convert the file
item.Convert("Cadd - Builtin", _
            "C:\Test\BuildingPlan.dwf", _
            "C:\Test\Out\ConvertedDrawings")
```

Conversion Settings -PNBuiltinCaddConverter

Name: ConverterPlugIn.PNBuiltinsOCRPDF.Enabled

Enable this setting to run OCR on your pages and create a searchable PDF file. Running OCR does slow down the conversion process. Does not apply to any other output format.

Values: String value; 1 to enable OCR, 0 to disable.

Conversion Settings -PNBuiltinCaddConverter

Name: **ConverterPlugIn.PNBuiltinsOCRPDF.Languages**

Values: A string representing the languages to recognize on the page. English is preselected by default. You must have at least one language selected for OCR. Select multiple languages by separating each language code with a plus (+) sign. For example, "eng+fra+deu" will recognize English, French and German text on your pages.

- Arabic(ara)
- **English(eng)**
- French(fra)
- German(deu)
- Hebrew(heb)
- Hindi(hin)
- Italian(ita)
- Spanish(spa)

Additional languages can be downloaded and added to Document Conversion Service if necessary.

Name: **ConverterPlugIn.PNBuiltinsOCRPDF.FirstPageOnly**

Values: Allows you to only run the OCR process on the first page of every file. String value; 1 to process only the first page, 0 to process all the pages in the document.

Adding Languages

Document Conversion Service comes with files to support recognizing Arabic, English, French, German, Hebrew, Hindi, Italian, and Spanish. You can download additional language files or complete sets of language files from [Traineddata Files for Tesseract](#).

To add them to Document Conversion Service, copy the desired *.traineddata files into the following folder:

```
%PROGRAMDATA%\PEERNET\Document Conversion Service\tessdata
```

Built-in PDF Converter Options

The Built-in PDF Converter replaces the [Adobe Reader converter](#). It eliminates the need to install Adobe Reader to convert PDF files. Unlike when converting using the Adobe Reader converter, files are opened and converted without printing, resulting in faster conversion.

**Caution**

This feature is not supported on Microsoft® Windows Server 2008 R2 and Microsoft® Windows 7.

This converter has no individual options.

Built-in Text Converter Options

This converter handles processing text-based files, including wide-format text files. It handles text-based log files, batch files, source code files, and many others.



Caution

This feature is not supported on Microsoft® Windows Server 2008 R2 and Microsoft® Windows 7.

Text files may or may not contain form feeds. A form feed is a special character that causes any text after it to start on a new page. When your text document has form feeds, the text encoding, font size, and page size used to position the page contents are auto-detected.

A document with no form feed characters is considered a single page. The number of characters and lines, along with the font and font size, is used to calculate a page size. The text is split into pages using the calculated page size. Text files that commonly have no form feeds are source code files, XML files, log files, and similar.

There are two sets of options - one for [text files with form feeds](#) and one for [text documents without form feeds](#).

Table values in bold text are the default value for that setting.



Sample Profile

```
<?xml version="1.0" encoding="utf-8"?>
<Profile Type="0"
  DisplayName="TIFF 300dpi Custom Text"
  Description="Converts Text files using a custom font.">
  <Settings>
    <add Name
      ="ConverterPlugIn.PNBuiltinTextConverterSettingsContentWithFormFeeds.FontName"
      Value="Consolas"/>
    <add Name
      ="ConverterPlugIn.PNBuiltinTextConverterSettingsContentWithFormFeeds.FontSizeInPoints"
      Value="11"/>

    <!-- Output file options -->
    <add Name="Devmode settings;Resolution" Value="300"/>
    <add Name="Save;Output File Format" Value="TIFF Multipaged"/>
    ...

  </Settings>
</Profile>
```

**Code Sample - C#**

```
PNDocConvQueueServiceLib.PNDocConvQueueItem item = null;

// Create the conversion item
item = new PNDocConvQueueServiceLib.PNDocConvQueueItem();

// Set conversion settings
item.Set("ConverterPlugIn.PNBuiltinTextConverterSettingsContentWithFormFeeds.FontName",
"Consolas");
item.Set
("ConverterPlugIn.PNBuiltinTextConverterSettingsContentWithFormFeeds.FontSizeInPoints",
"11");

item.Set("Devmode settings;Resolution", "300");
item.Set("Save;Output File Format", "TIFF Multipaged");
...
// convert the file
item.Convert("Text - Builtin", _
            @"C:\Test\Report.txt", _
            @"C:\Test\Out\ConvertedTextReports");
```

**Code Sample - VB.NET**

```
Dim item As PNDocConvQueueServiceLib.IPNDocConvQueueItem

' Create the conversion item
item = New PNDocConvQueueServiceLib.PNDocConvQueueItem()

' Set conversion settings
item.Set("ConverterPlugIn.PNBuiltinTextConverterSettingsContentWithFormFeeds.FontName",
"Consolas")
item.Set
("ConverterPlugIn.PNBuiltinTextConverterSettingsContentWithFormFeeds.FontSizeInPoints",
"10")

item.Set("Devmode settings;Resolution", "300")
item.Set("Save;Output File Format", "TIFF Multipaged")
...
' convert the file
item.Convert("Text - Builtin", _
            "C:\Test\Report.txt", _
            "C:\Test\Out\ConvertedTextReports")
```

This table lists the options for handling text files **with form feeds** (page eject). A form feed automatically causes any text after it to start on a new page. For options for text files **without form feeds**, see [this table](#).

Conversion Settings -PNBuiltinTextConverter - Text With Form Feeds Options

Name: **ConverterPlugIn.PNBuiltinTextConverterSettingsContentWithFormFeeds.FontName**

Choose which font to use for the text in your document. Choose a fixed-width (fixed pitch) font for best results when your text files contain tables and column of data when alignment is important. If the font specified does not exist, or the name is spelled incorrectly, the default of **Courier New** is used.

Values: String value of the font name. The default is **Courier New**.

Name: **ConverterPlugIn.PNBuiltinTextConverterSettingsContentWithFormFeeds.FontSizeInPoints**

The size of the font is in points. Each point is 1/72 of an inch. A font size of 10 points means each character is approximately 1/14 of an inch tall.

If you change the font and/or the font size, and you are word-wrapping the text, you may need to adjust the MaxCharactersPerLine to account for your document content and chosen font.

Values: String value of the size of the font in points. The default is **10**. The smallest size allowed is **6**, and largest, **72**.

Name: **ConverterPlugIn.PNBuiltinTextConverterSettingsContentWithFormFeeds.LineWrapMode**

Line wrapping is the automatic shifting of text to a new line when we have reached the maximum number of characters per line. The maximum number of characters per line depends on any font, font size, paper size, and maximum character settings you have set.

Values: **None** - Do not line wrap. When not line wrapping, the paper size any text that occurs past the maximum number of characters per line is truncated.
Plain - wraps the text at the calculated number of characters per line, with no attempt to backtrack to find the last word break.
Word - wraps the text at the calculated number of characters per line, backtracking to find the last word break when necessary.

Name: **ConverterPlugIn.PNBuiltinTextConverterSettingsContentWithFormFeeds.MaxCharactersPerLine**

Specifies the maximum number of characters per line when *LineWrapMode* is set to *Plain* or *Word*. The desired maximum number of characters per line depends on any font, font size, and paper size you have set. You may need to adjust this if you change the font and/or the font size.

Values: String value of the maximum number of characters per line. The default is **83**.

Conversion Settings -PNBuiltinTextConverter - Text With Form Feeds Options

Name: **ConverterPlugIn.PNBuiltinTextConverterSettingsContentWithFormFeeds.PageSizeDetectionMode**

Choose to auto-detect the page size needed for your text document, size the content to fit a specified page size, or always use a specified page size.

Values: **AutoDetect** - Scans the text document to determine the number of characters per line and lines per page. The number of characters and lines, and form feed characters, along with the *FontName* and *FontSizeInPoints*, is used to calculate the page size needed. Using this page size, we search the Document Conversion Service printer's forms list for a paper size that is the closest match without being smaller. If we cannot find a matching paper size, we use the settings for *PageSize_DefaultPageWidthInPoints* and *PageSize_DefaultPageHeightInPoints*.

SizeToFit - Scans the text document to determine the number of characters per line and lines per page. The number of characters and lines, along with the font and font size, is used to calculate the page size. Unlike AutoDetect, we use this calculated page size as the paper size. There is no attempt to match to a standard paper size.

UseSpecified - Always uses the page size set using *PageSize_DefaultPageWidthInPoints* and *PageSize_DefaultPageHeightInPoints*. Text that exceeds the width or height of the page can get cut off.

Name: **ConverterPlugIn.PNBuiltinTextConverterSettingsContentWithFormFeeds.PageSize_AutoDetect_MinPageWidthInPoints**

The minimum width of an auto-detected page size. The page will not be smaller than this width, but it can be wider. This is entered in points. There are 72 points per inch, so a default width of 576 is 8 inches.

Values: String value of the minimum page width in points. The default is **576**, or 8 inches.

Name: **ConverterPlugIn.PNBuiltinTextConverterSettingsContentWithFormFeeds.PageSize_AutoDetect_MinPageHeightInPoints**

The minimum height of an auto-detected page size. The page will not be shorter than this width, but it can be taller. There are 72 points per inch, so a height of 576 is 8 inches.

Values: String value of the minimum page height in points. The default is **576**, or 8 inches.

Conversion Settings -PNBuiltinTextConverter - Text With Form Feeds Options

Name:	ConverterPlugIn.PNBuiltinTextConverterSettingsContentWithFormFeeds.PageSize_DefaultPageWidthInPoints
	Sets the desired page width in points. This setting is used if <i>PageSizeDetectionMode</i> is set to <i>UseSpecified</i> , or if it is set to <i>AutoDetect</i> and it does not match an appropriate page size. There are 72 points per inch, so the default width of 612 is 8.5 inches.
Values:	String value of the desired page width in points. The default is 612 , or 8.5 inches.
Name:	ConverterPlugIn.PNBuiltinTextConverterSettingsContentWithFormFeeds.PageSize_DefaultPageHeightInPoints
	Sets the desired page height in points. This setting is used if <i>PageSizeDetectionMode</i> is set to <i>UseSpecified</i> , or if it is set to <i>AutoDetect</i> and it does not match an appropriate page size. There are 72 points per inch, so the default height of 792 is 11 inches.
Values:	String value of the desired page height in points. The default is 792 , or 11 inches.
Name:	ConverterPlugIn.PNBuiltinTextConverterSettingsContentWithFormFeeds.PageSize_LeftMarginInPoints
	Sets the width of the left margin. A margin is the space between the edge of the page and the main body of text. There are 72 points per inch, so the default width of 18 is 0.25 inches.
Values:	String value of the desired left margin width in points. The default is 54 , or 0.75 inches.
Name:	ConverterPlugIn.PNBuiltinTextConverterSettingsContentWithFormFeeds.PageSize_RightMarginInPoints
	Sets the width of the right margin. A margin is the space between the edge of the page and the main body of text. There are 72 points per inch, so the default width of 18 is 0.25 inches.
Values:	String value of the desired right margin width in points. The default is 54 , or 0.75 inches.

Conversion Settings -PNBuiltinTextConverter - Text With Form Feeds Options	
Name:	ConverterPlugIn.PNBuiltinTextConverterSettingsContentWithFormFeeds.PageSize_TopMarginInPoints
	Sets the height of the top margin. A margin is the space between the edge of the page and the main body of text. There are 72 points per inch, so the default height of 18 is 0.25 inches.
Values:	String value of the desired top margin height in points. The default is 72 , or 1 inch.
Name:	ConverterPlugIn.PNBuiltinTextConverterSettingsContentWithFormFeeds.PageSize_BottomMarginInPoints
	Sets the height of the bottom margin. A margin is the space between the edge of the page and the main body of text. There are 72 points per inch, so the default height of 18 is 0.25 inches.
Values:	String value of the desired bottom margin height in points. The default is 72 , or 1 inch.
Name:	ConverterPlugIn.PNBuiltinTextConverterSettingsContentWithFormFeeds.TabStopEveryNSpaces
	Replaces any tab stops in the text document with the given number of spaces (columns).
Values:	String value of the number of spaces to replace the tab stop with. The default is 8 spaces.

This table lists the options for handling text files **without form feeds** (page ejects). When a text document does not have form feeds, the settings **PageSize_DefaultPageWidthInPoints** and **PageSize_DefaultPageHeightInPoints** are used to determine the paper size and when a new page begins. For options for text files **with form feeds**, see [this table](#).

Conversion Settings -PNBuiltinTextConverter - Text With No Form Feeds Options	
Name:	ConverterPlugIn.PNBuiltinTextConverterSettingsContentNoFormFeeds.FontName
	Choose which font to use for the text in your document. Choose a fixed-width (fixed pitch) font for best results when your text files contain tables and column of data when alignment is important. If the font specified does not exist, or the name is spelled incorrectly, the default of Courier New is used.
Values:	String value of the font name. The default is Courier New .

Conversion Settings -PNBuiltinTextConverter - Text With No Form Feeds Options

Name: **ConverterPlugIn.PNBuiltinTextConverterSettingsContentNoFormFeeds.FontSizeInPoints**

The size of the font is in points. Each point is 1/72 of an inch. A font size of 10 points means each character is approximately 1/14 of an inch tall.

If you change the font and/or the font size, and you are word-wrapping the text, you may need to adjust the MaxCharactersPerLine to account for your document content and chosen font.

Values: String value of the size of the font in points. The default is **10**. The smallest size allowed is **6**, and largest, **72**.

Name: **ConverterPlugIn.PNBuiltinTextConverterSettingsContentNoFormFeeds.LineWrapMode**

Line wrapping is the automatic shifting of text to a new line when we have reached the maximum number of characters per line. The maximum number of characters per line depends on any font, font size, paper size, and maximum character settings you have set.

Values: None - Do not line wrap. When not line wrapping, any text that occurs past the maximum number of characters per line is truncated.
Plain - wraps the text at the calculated number of characters per line, with no attempt to backtrack to find the last word break.
Word - wraps the text at the calculated number of characters per line, backtracking to find the last word break when necessary.

Name: **ConverterPlugIn.PNBuiltinTextConverterSettingsContentNoFormFeeds.MaxCharactersPerLine**

Specifies the maximum number of characters per line when *LineWrapMode* is set to *Plain* or *Word*. The desired maximum number of characters per line depends on any font, font size, and paper size you have set. You may need to adjust this if you change the font and/or the font size.

Values: String value of the maximum number of characters per line. The default is **83**.

Conversion Settings -PNBuiltinTextConverter - Text With No Form Feeds Options

Name: **ConverterPlugIn.PNBuiltinTextConverterSettingsContentNoFormFeeds.PageSizeDetectionMode**

Choose to auto-detect the page size needed for your text document, size the content to fit a specified page size, or always use a specified page size.

Values: **AutoDetect** - Scans the text document to determine the number of characters per line and lines per page. The number of characters and lines, and form feed characters, along with the *FontName* and *FontSizeInPoints*, is used to calculate the page size needed. Using this page size, we search the Document Conversion Service printer's forms list for a paper size that is the closest match without being smaller. If we cannot find a matching paper size, we use the settings for *PageSize_DefaultPageWidthInPoints* and *PageSize_DefaultPageHeightInPoints*.

SizeToFit - Scans the text document to determine the number of characters per line and lines per page. The number of characters and lines, along with the font and font size, is used to calculate the page size. Unlike **AutoDetect**, we use this calculated page size as the paper size. There is no attempt to match to a standard paper size.

UseSpecified - Always uses the page size set using *PageSize_DefaultPageWidthInPoints* and *PageSize_DefaultPageHeightInPoints*. Text that exceeds the width or height of the page can get cut off.

Name: **ConverterPlugIn.PNBuiltinTextConverterSettingsContentNoFormFeeds.PageSize_AutoDetect_MinPageWidthInPoints**

The minimum width of an auto-detected page size. The page will not be smaller than this width, but it can be wider. This is entered in points. There are 72 points per inch, so a default width of 576 is 8 inches.

Values: String value of the minimum page width in points. The default is **576**, or 8 inches.

Name: **ConverterPlugIn.PNBuiltinTextConverterSettingsContentNoFormFeeds.PageSize_AutoDetect_MinPageHeightInPoints**

The minimum height of an auto-detected page size. The page will not be shorter than this width, but it can be taller. There are 72 points per inch, so a height of 576 is 8 inches.

Values: String value of the minimum page height in points. The default is **576**, or 8 inches.

Conversion Settings -PNBuiltinTextConverter - Text With No Form Feeds Options

Name: **ConverterPlugIn.PNBuiltinTextConverterSettingsContentNoFormFeeds.PageSize_DefaultPageWidthInPoints**

Sets the desired page width in points. This setting is used if *PageSizeDetectionMode* is set to *UseSpecified*, or if it is set to *AutoDetect* and it does not match an appropriate page size. There are 72 points per inch, so the default width of 612 is 8.5 inches.

Values: String value of the desired page width in points. The default is **612**, or 8.5 inches.

Name: **ConverterPlugIn.PNBuiltinTextConverterSettingsContentNoFormFeeds.PageSize_DefaultPageHeightInPoints**

Sets the desired page height in points. This setting is used if *PageSizeDetectionMode* is set to *UseSpecified*, or if it is set to *AutoDetect* and it does not match an appropriate page size. There are 72 points per inch, so the default height of 792 is 11 inches.

Values: String value of the desired page height in points. The default is **792**, or 11 inches.

Name: **ConverterPlugIn.PNBuiltinTextConverterSettingsContentNoFormFeeds.PageSize_LeftMarginInPoints**

Sets the width of the left margin. A margin is the space between the edge of the page and the main body of text. There are 72 points per inch, so the default width of 18 is 0.25 inches.

Values: String value of the desired left margin width in points. The default is **54**, or 0.75 inches.

Name: **ConverterPlugIn.PNBuiltinTextConverterSettingsContentNoFormFeeds.PageSize_RightMarginInPoints**

Sets the width of the right margin. A margin is the space between the edge of the page and the main body of text. There are 72 points per inch, so the default width of 18 is 0.25 inches.

Values: String value of the desired right margin width in points. The default is **54**, or 0.75 inches.

Conversion Settings -PNBuiltinTextConverter - Text With No Form Feeds Options

Name: **ConverterPlugIn.PNBuiltinTextConverterSettingsContentNoFormFeeds.Page Size_TopMarginInPoints**

Sets the height of the top margin. A margin is the space between the edge of the page and the main body of text. There are 72 points per inch, so the default height of 18 is 0.25 inches.

Values: String value of the desired top margin height in points. The default is **72**, or 1 inch.

Name: **ConverterPlugIn.PNBuiltinTextConverterSettingsContentNoFormFeeds.Page Size_BottomMarginInPoints**

Sets the height of the bottom margin. A margin is the space between the edge of the page and the main body of text. There are 72 points per inch, so the default height of 18 is 0.25 inches.

Values: String value of the desired bottom margin height in points. The default is **72**, or 1 inch.

Name: **ConverterPlugIn.PNBuiltinTextConverterSettingsContentNoFormFeeds.TabS topEveryNSpaces**

Replaces any tab stops in the text document with the given number of spaces (columns).

Values: String value of the number of spaces to replace the tab stop with. The default is **8** spaces.

Built-in Cadd Converter Options

The Built-in Cadd Converter converts the following types of **Computer-Aided Design** and **Drafting** files: DWF, DWFX, PLT and GBX. It does not require any third-party applications.



Caution

This feature is not supported on Microsoft® Windows Server 2008 R2 and Microsoft® Windows 7.

Table values in **bold** text are the default value for that setting.



Sample Profile

```
<?xml version="1.0" encoding="utf-8"?>
<Profile Type="0"
    DisplayName="TIFF 300dpi from CAD"
    Description="Converts Cadd Files with optional password.">
  <Settings>

    <add Name ="ConverterPlugIn.PNBuiltinCaddConverterSettings.OpenPassword"
Value="password"/>

    <!-- Output file options -->
    <add Name="Devmode settings;Resolution" Value="300"/>
    <add Name="Save;Output File Format" Value="TIFF Multipaged"/>
    ...

  </Settings>
</Profile>
```



Code Sample - C#

```
PNDocConvQueueServiceLib.PNDocConvQueueItem item = null;

// Create the conversion item
item = new PNDocConvQueueServiceLib.PNDocConvQueueItem();

// Set conversion settings
item.Set("ConverterPlugIn.PNBuiltinCaddConverterSettings.OpenPassword", "password");

item.Set("Devmode settings;Resolution", "300");
item.Set("Save;Output File Format", "TIFF Multipaged");
...
// convert the file
item.Convert("Cadd - Builtin", _
    @"C:\Test\BuildingPlan.dwf", _
    @"C:\Test\Out\ConvertedDrawings");
```

**Code Sample - VB.NET**

```
Dim item As PNDocConvQueueServiceLib.IPNDocConvQueueItem

' Create the conversion item
item = New PNDocConvQueueServiceLib.PNDocConvQueueItem()

' Set conversion settings
item.Set("ConverterPlugIn.PNBuiltinCaddConverterSettings.OpenPassword", "password")

item.Set("Devmode settings;Resolution", "300")
item.Set("Save;Output File Format", "TIFF Multipaged")
...
' convert the file
item.Convert("Cadd - Builtin", _
            "C:\Test\BuildingPlan.dwf", _
            "C:\Test\Out\ConvertedDrawings")
```

Conversion Settings -PNBuiltinCaddConverter

Name:	ConverterPlugIn.PNBuiltinCaddConverterSettings.OpenPassword
	Applies to password protected DWF and DWFX files. Pass in the password to use to open the files.
Values:	String value; the password to use to try an open the file.

Built-in Image Converter Options

The Built-in Image Converter is a replacement for our original [Image Converter](#). We've improved the speed and memory use needed when converting, modifying, and rotating images. This converter improves image conversion speed and provides better handling of all image types.



Caution

This feature is not supported on Microsoft® Windows Server 2008 R2 and Microsoft® Windows 7.

This converter has no individual options.

Word Converter Options

These options control the behavior of the Word converter used by Document Conversion Service. Table values in **bold** text are the default value for that setting.



Sample Profile

```
<?xml version="1.0" encoding="utf-8"?>
<Profile Type="0"
  DisplayName="TIFF 300dpi Word with Markup, replace dates "
  Description="Prints Word documents with comments and tracking visible.">
  <Settings>

    <!-- Print Word with markup and 2 pages per sheet -->
    <add Name="Microsoft.Word.Document.PrintOut.Item"
      Value="DocumentAndMarkup"/>
    <add Name="Microsoft.Word.PageSetup.TwoPagesOnOne"
      Value="True"/>

    <!-- Replace date fields with <AUTODATE> string -->
    <add Name="Microsoft.Word.ReplaceFieldDateWith"
      Value="&lt;AUTODATE&gt;"/>

    <!-- Output file options -->
    <add Name="Devmode settings;Resolution" Value="300"/>
    <add Name="Save;Output File Format" Value="TIFF Multipaged"/>
    ...
  </Settings>
</Profile>
```



Code Sample - C#

```
PNDocConvQueueServiceLib.PNDocConvQueueItem item = null;

// Create the conversion item
item = new PNDocConvQueueServiceLib.PNDocConvQueueItem();

// Set conversion settings
item.Set("Microsoft.Word.Document.PrintOut.Item", "DocumentAndMarkup");
item.Set("Microsoft.Word.PageSetup.TwoPagesOnOne", "True");
item.Set("Microsoft.Word.ReplaceFieldDateWith", "<AUTODATE>");
item.Set("Devmode settings;Resolution", "300");
item.Set("Save;Output File Format", "TIFF Multipaged");
...
// convert the file
item.Convert("Microsoft Word",
  @"C:\Test\Report.docx",
  @"C:\Test\Out\ConvertedReport");
```



Code Sample - VB.NET

```
Dim item As PNDocConvQueueServiceLib.IPNDocConvQueueItem

' Create the conversion item
item = New PNDocConvQueueServiceLib.PNDocConvQueueItem()

' Set conversion settings
item.Set("Microsoft.Word.Document.PrintOut.Item", "DocumentAndMarkup")
item.Set("Microsoft.Word.PageSetup.TwoPagesOnOne", "True")
item.Set("Microsoft.Word.ReplaceFieldDateWith", "<AUTODATE>")
item.Set("Devmode settings;Resolution", "300")
item.Set("Save;Output File Format", "TIFF Multipaged")
...
' convert the file
item.Convert("Microsoft Word", _
            "C:\Test\Report.docx", _
            "C:\Test\Out\ConvertedReport")
```

Conversion Settings - Word Printing Options

Name: **Microsoft.Word.Document.PrintOut.Item**

Choose what parts of the document to print.

Values: **Document** - prints only the document.
DocumentAndMarkup - prints the document and any markup such as tracked changes and comments.
DocumentMarkup - prints only the markup.
DocumentProperties - prints only the document properties.

Name: **Microsoft.Word.Document.PrintOut.PageType**

Choose if you want to print all pages, even pages or odd pages.

Values: **All**
Even
Odd

Name: **Microsoft.Word.ActiveWindow.View.MarkupMode**

Sets the display mode for tracked changes in the document. Applies when using the printing option *Word.Document.PrintOut.Item* set to *DocumentAndMarkup* or *DocumentMarkup*.

Values: **BalloonRevisions** - Displays revisions in balloons in the left or right margin.
InLineRevisions - Displays revisions within the text using strikethrough for deletions and underlining for insertions.
MixedRevisions - Shows only comments and formatting revisions in the document.

Conversion Settings - Word Printing Options

Name:	Microsoft.Word.ActiveWindow.View.RevisionsView (Office 2010 and earlier)
	This setting is deprecated starting with Office 2013. Use Microsoft.Word.ActiveWindow.View.RevisionsFilter.View and Microsoft.Word.ActiveWindow.View.RevisionsFilter.Markup instead. Specifies whether the original version of a document or a version with revisions and formatting changes applied are displayed.
Values:	ViewFinal - Displays the document with formatting and content changes applied. ViewOriginal - Displays the document before changes were made.
Name:	Microsoft.Word.ActiveWindow.View.RevisionsFilter.View (Office 2013 and later)
	Specifies whether the original version of a document or a version with revisions and formatting changes applied are displayed. Replaces Microsoft.Word.ActiveWindow.View.RevisionsView in Office 2013 and later versions.
Values:	ViewFinal - Displays the document with formatting and content changes applied. ViewOriginal - Displays the document before changes were made.
Name:	Microsoft.Word.ActiveWindow.View.RevisionsFilter.Markup (Office 2013 and later)
	Specifies the extent of reviewer markup displayed in the document. This setting is used starting with Office 2013.
Values:	NoMarkup - Displays the final document with no markup visible. SimpleMarkup - Displays the final document in simple markup: with revisions incorporated, but with no markup visible. AllMarkup - Displays the final document with all markup visible.
Name:	Microsoft.Word.ActiveWindow.View.ShowComments
	Pass <i>True</i> to display any comments in the document. Must be used with <i>Microsoft.Word.ActiveWindow.View.MarkupMode</i> to display the comments as balloons or inline, and <i>Microsoft.Word.Document.PrintOut.Item</i> set to print document markup.
Values:	String value " True " or " False ".

Conversion Settings - Word Printing Options

Name:	Microsoft.Word.ActiveWindow.View.ShowFormatChanges
	Pass <i>True</i> to display any formatting changes made to a document with Track Changes enabled. Must be used with <i>Microsoft.Word.ActiveWindow.View.MarkupMode</i> to display the comments as balloons or inline, and <i>Microsoft.Word.Document.PrintOut.Item</i> set to print document markup.
Values:	String value " True " or "False".
Name:	Microsoft.Word.ActiveWindow.View.ShowHiddenText
	Pass <i>True</i> to display any text that was formatted as hidden.
Values:	String value "True" or "False".
Name:	Microsoft.Word.ActiveWindow.View.ShowHighlight
	Pass <i>True</i> to have highlighted text displayed with the highlighted background.
Values:	String value "True" or "False".
Name:	Microsoft.Word.ActiveWindow.View.ShowInkAnnotations
	Pass <i>True</i> to to show handwritten ink annotations in the document. Must be used with <i>Microsoft.Word.Document.PrintOut.Item</i> set to print document markup.
Values:	String value " True " or "False".
Name:	Microsoft.Word.ActiveWindow.View.ShowInsertionsAndDeletions
	Pass <i>True</i> to display any insertions and deletions made to a document with Track Changes enabled. Must be used with <i>Microsoft.Word.ActiveWindow.View.MarkupMode</i> set to display the changes as balloons or inline, and <i>Microsoft.Word.Document.PrintOut.Item</i> set to print document markup.
Values:	String value " True " or "False".

Conversion Settings - Word Printing Options

Name: **Microsoft.Word.ActiveWindow.View.ShowMarkupAreaHighlight**

Pass *True* to have the markup area that shows revision and comment balloons displayed shaded. Applies only when *Microsoft.Word.ActiveWindow.View.MarkupMode* is set to display markup as balloons, and *Microsoft.Word.Document.PrintOut.Item* is set to print document markup.

Values: String value "**True**" or "**False**".

Name: **Microsoft.Word.Options.AllowA4LetterResizing**

Pass *True* to automatically adjust Letter-sized documents to fit A4 paper, or to adjust A4-sized documents to fit Letter paper. This only affects printing and happens when the paper size of the printer does not match the paper size that is set in Word.

Values: String value "**True**" or "**False**".

Conversion Settings - Word Field Replacement

Name: **Microsoft.Word.ReplaceFieldDateWith**

Replaces any DATE fields in the Word document with the provided string.

Values: The string value to place in the field.

Name: **Microsoft.Word.ReplaceFieldTimeWith**

Replaces any TIME fields in the Word document with the provided string.

Values: The string value to place in the field.

Name: **Microsoft.Word.ReplaceFieldFileNameWith**

Replaces any FILENAME fields in the Word document with the provided string.

Values: A string value to replace the auto file name field.

Conversion Settings - Word Document Protection

Name: **Microsoft.Word.UnprotectPassword**

The password to use to remove the protection on the the Word document and allow changes. This password is passed as clear text and is visible to anyone.

Values: A string value containing the password.

Name: **Microsoft.Word.OpenPassword**

The password to use to open a password-protected Word document. This password is passed as clear text and is visible to anyone.

Values: A string value containing the password.

Name: **Microsoft.Word.WritePassword**

The password to use to allow saving changes to the Word document. This password is passed as clear text and is visible to anyone.

Values: A string value containing the password.

Conversion Settings - Word Page Setup Printing Options

Name: **Microsoft.Word.PageSetup.BookFoldPrinting**

Pass *True* to print the document as a booklet.

Values: String value "True" or "False".

Name: **Microsoft.Word.PageSetup.BookFoldPrintingSheets**

The number pages to print in each booklet. This number must be a multiple of 4. If not, the default setting of "Auto" will be used.

When using "Auto", Word will automatically determine the number of sheets per booklet, splitting the sheets into separate booklets as necessary. Passing "All" will print all of your pages in a single booklet.

Values: String value "**Auto**", "All" or the number of pages to be printed in each booklet.

Conversion Settings - Word Page Setup Printing Options

Name:	Microsoft.Word.PageSetup.BookFoldRevPrinting
	Pass <i>True</i> to reverse the printing order for booklet printing, bidirectional or Asian language documents only.
Values:	String value "True" or "False".
Name:	Microsoft.Word.PageSetup.BottomMargin
	Set the size of the bottom margin in points.
Values:	String value of the desired margin height.
Name:	Microsoft.Word.PageSetup.DifferentFirstPageHeaderFooter
	Pass <i>True</i> to use a different header on the first page.
Values:	String value "True" or "False".
Name:	Microsoft.Word.PageSetup.FooterDistance
	Set the distance (in points) between the top of the footer to the bottom of the page.
Values:	String value of the desired footer height.
Name:	Microsoft.Word.PageSetup.Gutter
	Set the amount of extra margin space added for binding.
Values:	String value of the desired gutter width.
Name:	Microsoft.Word.PageSetup.GutterPos
	Sets which side of the document the gutter is placed.
Values:	Left Right Top

Conversion Settings - Word Page Setup Printing Options

Name: **Microsoft.Word.PageSetup.GutterStyle**

Sets how the gutters are placed; on the left for left-to-right languages or on the right side of the document for right-to-left languages.

Values: Bidi - use bidirectional gutters for right-to-left languages.
Latin - use Latin gutter for left-to-right text.

Name: **Microsoft.Word.PageSetup.HeaderDistance**

Set the distance (in points) between the bottom of the header to the top of the page.

Values: String value of the desired header height.

Name: **Microsoft.Word.PageSetup.LayoutMode**

Sets the layout of the text in the document. Genko, Grid and LineGrid use the setting Microsoft.Word.PageSetup.LinesPage.

Values: Default - No grid is used to lay out text.
Genko - Text is laid out on a grid with characters aligned on the gridlines.
Grid - Text is laid out on a grid but the characters are not aligned on the gridlines.
LineGrid - Text is laid out on a grid; only the number of lines is specified.

Name: **Microsoft.Word.PageSetup.LeftMargin**

Set the size of the left margin in points.

Values: String value of the desired margin height.

Name: **Microsoft.Word.PageSetup.LinesPage**

The number of lines per page of the document. Used with the Microsoft.Word.PageSetup.LayoutMode setting.

Values: String value of the desired number of lines per page.

Name: **Microsoft.Word.PageSetup.MirrorMargins**

Pass *True* to have the inside and outside margins of facing pages to be the same width.

Values: String value "True" or "False".

Conversion Settings - Word Page Setup Printing Options

Name:	Microsoft.Word.PageSetup.OddAndEvenPagesHeaderFooter Pass <i>True</i> to have different headers for odd-numbered and even-numbered pages.
Values:	String value "True" or "False".
Name:	Microsoft.Word.PageSetup.Orientation Sets the orientation of the page.
Values:	Landscape Portrait
Name:	Microsoft.Word.PageSetup.PageHeight Sets the height of the page in points.
Values:	String value of the desired height.
Name:	Microsoft.Word.PageSetup.PageWidth Sets the width of the page in points.
Values:	String value of the desired width.

Conversion Settings - Word Page Setup Printing Options

Name: **Microsoft.Word.PageSetup.PaperSize**

Sets the paper size.

Values:

- Paper10x14 - 10 in. x 14 in.
- Paper11x17 - 11 in. x 17 in.
- PaperA3 - A3 (297 mm x 420 mm)
- PaperA4 - A4 (210 mm x 297 mm)
- PaperA4Small - A4 Small (210 mm x 297 mm)
- PaperA5 - A5 (148 mm x 210 mm)
- PaperB4 - B4 (250 mm x 354 mm)
- PaperB5 - B5 (182 mm x 257 mm)
- PaperCsheet - C size sheet
- PaperEnvelope10 - Envelope #10 (4-1/8 in. x 9-1/2 in.)
- PaperEnvelope11 - Envelope #11 (4-1/2 in. x 10-3/8 in.)
- PaperEnvelope14 - Envelope #14 (5 in. x 11-1/2 in.)
- PaperEnvelope9 - Envelope #9 (3-7/8 in. x 8-7/8 in.)
- PaperEnvelopeB4 - Envelope B4 (250 mm x 353 mm)
- PaperEnvelopeB5 - Envelope B5 (176 mm x 250 mm)
- PaperEnvelopeB6 - Envelope B6 (176 mm x 125 mm)
- PaperEnvelopeC3 - Envelope C3 (324 mm x 458 mm)
- PaperEnvelopeC4 - Envelope C4 (229 mm x 324 mm)
- PaperEnvelopeC5 - Envelope C5 (162 mm x 229 mm)
- PaperEnvelopeC6 - Envelope C6 (114 mm x 162 mm)
- PaperEnvelopeC65 - Envelope C65 (114 mm x 229 mm)
- PaperEnvelopeDL - Envelope DL (110 mm x 220 mm)
- PaperEnvelopeItaly - Envelope (110 mm x 230 mm)
- PaperEnvelopeMonarch - Envelope Monarch (3-7/8 in. x 7-1/2 in.)
- PaperEnvelopePersonal - Envelope (3-5/8 in. x 6-1/2 in.)
- PaperExecutive - Executive (7-1/2 in. x 10-1/2 in.)
- PaperFanfoldLegalGerman - German Legal Fanfold (8-1/2 in. x 13 in.)
- PaperFanfoldStdGerman - German Standard Fanfold (8-1/2 in. x 12 in.)
- PaperFolio - Folio (8-1/2 in. x 13 in.)
- PaperLedger - Ledger (17 in. x 11 in.)
- PaperLegal - Legal (8-1/2 in. x 14 in.)
- PaperLetter - Letter (8-1/2 in. x 11 in.)
- PaperLetterSmall - Letter Small (8-1/2 in. x 11 in.)
- PaperNote - Note (8-1/2 in. x 11 in.)
- PaperQuarto - Quarto (215 mm x 275 mm)
- PaperStatement - Statement (5-1/2 in. x 8-1/2 in.)
- PaperTabloid - Tabloid (11 in. x 17 in.)

Name: **Microsoft.Word.PageSetup.RightMargin**

Set the size of the right margin in points.

Values: String value of the desired margin width.

Conversion Settings - Word Page Setup Printing Options

Name: **Microsoft.Word.PageSetup.SuppressEndnotes**

Pass *True* to suppress any endnotes.

Values: String value "True" or "False".

Name: **Microsoft.Word.PageSetup.TopMargin**

Set the size of the top margin in points.

Values: String value of the desired margin height.

Name: **Microsoft.Word.PageSetup.TwoPagesOnOne**

Pass *True* to split the paper right down the horizontal center (for portrait) and vertical center (for landscape) and print two "pages" per sheet of paper. This does not shrink two pages of the document onto each single output page but rather changes the text layout of the document to reflect each page size being one half of the currently selected paper size.

Values: String value "True" or "False".

Name: **Microsoft.Word.PageSetup.VerticalAlignment**

Sets the vertical alignment of the text on each page.

Values: Bottom
Center
Justify
Top

Excel Converter Options

These options control the behavior of the Excel converter used by Document Conversion Service. If the workbook, or any spreadsheet in the workbook is password protected and the password is not known, the options are ignored. The settings cannot be applied to a protected workbook or spreadsheet.

Table values in **bold** text are the default value for that setting. Not all settings have default values; these settings are optional and the appropriate setting in the spreadsheet being printed will be used.

Sample Profile

```
<?xml version="1.0" encoding="utf-8"?>
<Profile Type="0"
  DisplayName="TIFF 300dpi Excel Charts First, replace dates "
  Description="Prints Excel charts, sheets with grid lines.">
  <Settings>

    <!-- Print Charts then workbooks, show grid lines -->
    <add Name ="Microsoft.Excel.PrintOut" Value="PrintOutChartsThenWorkbook"/>
    <add Name ="Microsoft.Excel.PageSetup.PrintGridlines" Value="True"/>

    <!-- Replace header/footer date fields with <AUTODATE> string -->
    <add Name ="Microsoft.Excel.ReplaceFieldDateWith"
      Value="&lt;AUTODATE&gt;"/>
    <add Name ="Microsoft.Excel.PageSetup.LeftHeader"
      Value="Sheet: &A"/>

    <!-- Output file options -->
    <add Name="Devmode settings;Resolution" Value="300"/>
    <add Name="Save;Output File Format" Value="TIFF Multipaged"/>
    ...

  </Settings>
</Profile>
```

Code Sample - C#

```
PNDocConvQueueServiceLib.PNDocConvQueueItem item = null;

// Create the conversion item
item = new PNDocConvQueueServiceLib.PNDocConvQueueItem();

// Set conversion settings
item.Set("Microsoft.Excel.PrintOut", "PrintOutChartsThenWorkbook");
item.Set("Microsoft.Excel.PageSetup.PrintGridlines", "True");

// Replace header/footer date fields with <AUTODATE> string
item.Set("Microsoft.Excel.ReplaceFieldDateWith", "<AUTODATE>");
item.Set("Microsoft.Excel.PageSetup.LeftHeader", "Sheet: &A");

item.Set("Devmode settings;Resolution", "300");
item.Set("Save;Output File Format", "TIFF Multipaged");
...
// convert the file
item.Convert("Microsoft Excel",
  @"C:\Test\Report.xlsx",
  @"C:\Test\Out\ConvertedReport");
```

**Code Sample - VB.NET**

```
Dim item As PNDocConvQueueServiceLib.IPNDocConvQueueItem

' Create the conversion item
item = New PNDocConvQueueServiceLib.PNDocConvQueueItem()

' Set conversion settings
item.Set("Microsoft.Excel.PrintOut", "PrintOutChartsThenWorkbook")
item.Set("Microsoft.Excel.PageSetup.PrintGridlines", "True")

' Replace header/footer date fields with <AUTODATE> string
item.Set("Microsoft.Excel.ReplaceFieldDateWith", "<AUTODATE>")
item.Set("Microsoft.Excel.PageSetup.LeftHeader", "Sheet: &A");

item.Set("Devmode settings;Resolution", "300")
item.Set("Save;Output File Format", "TIFF Multipaged")
...
' convert the file
item.Convert("Microsoft Excel", _
            "C:\Test\Report.xlsx", _
            "C:\Test\Out\ConvertedReport")
```

Conversion Settings - Excel General Formatting & Printing Options**Name:** **Microsoft.Excel.PrintOut**

Choose what part of the Excel spreadsheet to print.
The settings

For *PrintOutChartsOnly*, *PrintOutChartsThenWorkbook* and *PrintOutWorkbookThenCharts*, the option *Microsoft.Excel.PrintOut.PrintEmbeddedChartsFirst* controls if embedded charts are printed before or after any chart tabs in the spreadsheet.

Values: **PrintOutWorkbookOnly** - prints the entire workbook just as Excel does.

PrintOutActiveSheetOnly - prints only the last active (selected) sheet in the workbook. This is the selected tab at the time the Excel file was last saved.

PrintOutSelectedSheetsOnly - prints only the selected sheets in the workbook. Multiple sheets can be selected using the Ctrl+Left Click with the mouse.

PrintOutSheetsWithPrintAreasOnly - prints only sheets that have a print area set.

PrintOutChartsOnly - prints any charts tabs and embedded charts in the workbook.
PrintOutChartsThenWorkbook - prints all chart tabs and embedded charts, then prints all sheets in the workbook.

PrintOutWorkbookThenCharts - prints all sheets in the workbook, then prints all chart tabs and embedded charts.

For the three options above, embedded charts can be before or after other charts, as specified by the *Microsoft.Excel.PrintOut.PrintEmbeddedChartsFirst* setting.

Conversion Settings - Excel General Formatting & Printing Options	
Name:	Microsoft.Excel.PrintHiddenWorksheets Choose whether to print hidden worksheets or not.
Values:	False - do not print hidden worksheets. True - print hidden worksheets.
Name:	Microsoft.Excel.PrintOut.PrintEmbeddedChartsFirst When printing embedded charts, determines if the embedded charts are printed before or after any chart tabs in the spreadsheet. Applies only when <i>Microsoft.Excel.PrintOut</i> is set to print charts.
Values:	False - print embedded charts after all other charts. True - print embedded charts first.
Name:	Microsoft.Excel.PrintSheetsRangeByIndex The sheet numbers and ranges to include when printing. Separate each number and range with a comma. For example, "1, 3-5" prints sheet 1 and sheets 3 through 5. Numbers in the range exceeding the sheet count of the source document are ignored. Sheet numbers in the range are for visible sheets unless <i>Microsoft.Excel.PrintHiddenWorksheets</i> is true, then hidden sheets are included. Applies to the Microsoft.Excel.PrintOut options <i>PrintOutWorkbookOnly</i> , <i>PrintOutChartsOnly</i> , <i>PrintOutChartsThenWorkbook</i> and <i>PrintOutWorkbookThenCharts</i> . The range applies to both sheets and charts in the workbook. This print filter can be combined with <i>Microsoft.Excel.PrintSheetsRangeByName</i> , <i>Microsoft.Excel.PrintFirstNSheets</i> , <i>Microsoft.Excel.PrintLastNSheets</i> , and <i>Microsoft.Excel.PrintIfSheetNameMatchesRegex</i> .
Values:	The string representing the numbered sheet range.

Conversion Settings - Excel General Formatting & Printing Options

Name: **Microsoft.Excel.PrintSheetsRangeByName**

The names of the sheets and charts to include when printing, separated with a colon symbol (:) to print multiple sheets. Names not in the worksheet collection are ignored.

Applies only to visible sheets unless *Microsoft.Excel.PrintHiddenWorksheets* is true.

Applies to the **Microsoft.Excel.PrintOut** options *PrintOutWorkbookOnly*, *PrintOutChartsOnly*, *PrintOutChartsThenWorkbook* and *PrintOutWorkbookThenCharts*. The name selection applies to both sheets and charts in the workbook.

This print filter can be combined with *Microsoft.Excel.PrintSheetsRangeByIndex*, *Microsoft.Excel.PrintFirstNSheets*, *Microsoft.Excel.PrintLastNSheets* and *Microsoft.Excel.PrintIfSheetNameMatchesRegex*.

Values: The string of sheet or chart names, such as "Sheet1:Sheet3:Chart1".

Name: **Microsoft.Excel.PrintFirstNSheets**

Includes the designated number of sheets or charts, starting at the beginning of the workbook. If the workbook has less sheets (tabs) in total than the requested number, all sheets are printed.

Applies only to visible sheets unless *Microsoft.Excel.PrintHiddenWorksheets* is true.

Applies to the **Microsoft.Excel.PrintOut** options *PrintOutWorkbookOnly*, *PrintOutChartsOnly*, *PrintOutChartsThenWorkbook* and *PrintOutWorkbookThenCharts*. Applies to both sheets and charts in the workbook.

This print filter can be combined with *Microsoft.Excel.PrintSheetsRangeByName*, *Microsoft.Excel.PrintSheetsRangeByIndex*, *Microsoft.Excel.PrintLastNSheets*, and *Microsoft.Excel.PrintIfSheetNameMatchesRegex*.

Values: The number of sheets to print.

Conversion Settings - Excel General Formatting & Printing Options

Name: **Microsoft.Excel.PrintLastNSheets**

Includes the last designated number of sheets or charts, starting in the middle and going to the end of the workbook. If the workbook has less sheets (tabs) in total than the requested number, all sheets are printed.

Applies only to visible sheets unless *Microsoft.Excel.PrintHiddenWorksheets* is true.

Applies to the **Microsoft.Excel.PrintOut** options *PrintOutWorkbookOnly*, *PrintOutChartsOnly*, *PrintOutChartsThenWorkbook* and *PrintOutWorkbookThenCharts*. Applies to both sheets and charts in the workbook.

This print filter can be combined with *Microsoft.Excel.PrintSheetsRangeByName*, *Microsoft.Excel.PrintSheetsRangeByIndex*, *Microsoft.Excel.PrintFirstNSheets* and *Microsoft.Excel.PrintIfSheetNameMatchesRegex*.

Values: The number of sheets to print.

Name: **Microsoft.Excel.PrintIfSheetNameMatchesRegex**

Includes the sheet or chart if its name matches the regular expression.

Applies only to visible sheets unless *Microsoft.Excel.PrintHiddenWorksheets* is true.

Applies to the **Microsoft.Excel.PrintOut** options *PrintOutWorkbookOnly*, *PrintOutChartsOnly*, *PrintOutChartsThenWorkbook* and *PrintOutWorkbookThenCharts*. Applies to both sheets and charts in the workbook.

This print filter can be combined with *Microsoft.Excel.PrintSheetsRangeByIndex*, *Microsoft.Excel.PrintSheetsRangeByName*, *Microsoft.Excel.PrintFirstNSheets* and *Microsoft.Excel.PrintLastNSheets*.

Values: The regular expression to match the sheet name against.

Name: **Microsoft.Excel.AutoFit.KeepEmbeddedChartScaling**

Applies only when *Microsoft.Excel.AutoFitRows* and *Microsoft.Excel.AutoFitColumns* are set and if one or more embedded charts are on the sheet. When *True*, the width and height of any rows and columns under embedded charts are not auto-adjusted so that the chart does not change shape. Default is *True*.

Values: False - autofit all rows or columns, even under embedded charts. This can cause any charts to be squished or stretched.
True - do not autofit rows and columns under embedded charts; charts will keep their original scaling on the sheet.

Conversion Settings - Excel General Formatting & Printing Options

Name:	Microsoft.Excel.Worksheet.IncludeCellFormulasAsComments
	<p>For any cell that contains a formula, the formula added to that cell as a comment. If the cell has a comment, the formula is inserted with a carriage return before any current comment text. This must be used with <i>Microsoft.Excel.PageSetup.PrintComments</i> set to <i>PrintSheetEnd</i> to include the cell formulas listed by cell reference at the end of each sheet.</p> <p>To append the formula to the cell contents instead of inserting at the beginning, set <i>Microsoft.Excel.Worksheet.PrependCellFormulaToCommentText</i> to <i>False</i>.</p>
Values:	<p>False - do not add/update existing comments with the cell formula.</p> <p>True - add/update existing comment with the cell formula.</p>
Name:	Microsoft.Excel.Worksheet.PrependCellFormulaToCommentText
	<p>When using <i>Microsoft.Excel.Worksheet.IncludeCellFormulasAsComments</i>, the formula is prepended to the beginning of any existing comment text by default. To append the formula to the end of any existing comment text, set this option to <i>False</i>.</p>
Values:	<p>False - append the cell formula to the end of any existing comment text.</p> <p>True - prepend the cell formula to the beginning of any existing comment text.</p>
Name:	Microsoft.Excel.Worksheet.PrintOut.IgnorePrintAreas
	<p>When set to <i>True</i>, any print areas set on the worksheet will be ignored and the entire worksheet printed. Use with <i>Microsoft.Excel.Worksheet.PrintOut.ResetAllPageBreaks</i> to print the worksheet differently from the printing options in the worksheet.</p>
Values:	<p>False - prints using any print area set on the worksheet.</p> <p>True - prints the entire worksheet.</p>
Name:	Microsoft.Excel.Worksheet.ShowAllData
	<p>Makes all rows of any filtered data visible. This setting only applies to filtered data in the worksheet. To show hidden columns or rows use <i>Microsoft.Excel.AutoFitRows</i> and <i>Microsoft.Excel.AutoFitColumns</i>.</p>
Values:	<p>False - Leave data filtered (hidden).</p> <p>True - Show all the data on the worksheet.</p>

Conversion Settings - Excel General Formatting & Printing Options

Name: **Microsoft.Excel.Worksheet.ResetAllPageBreaks**

Set as *True* to resets all page breaks on each worksheet. Use with *Microsoft.Excel.Worksheet.PrintOut.IgnorePrintAreas* to print the worksheet differently from the printing options in the worksheet.

Values: **False** - Leave page breaks alone.
True - Reset all page breaks.

Name: **Microsoft.Excel.AutoFitRows**

If set to *True* then the height of the rows in the spreadsheet will be adjusted automatically to fit the contents of the cells. This setting will allow you to show all hidden rows in the worksheet.

Values: String value "True" or "False".

Name: **Microsoft.Excel.AutoFitRows.Adjust**

This setting is only applied when *Microsoft.Excel.AutoFitRows* is set to "True" and allows you to add the height specified (in points) to all rows after they have been auto-fit. The maximum row height allowed in Excel is 409 points. It is not normally needed to add height to each row and adding height to each row can be a time-consuming operation; only use this option if absolutely needed.

Values: String value of the amount, in points, by which to adjust the row height.

Name: **Microsoft.Excel.AutoFitColumns**

If set to *True* then the width the columns in the spreadsheet will be adjusted to fit the contents of the cells. This setting will allow you to show all hidden columns in the worksheet.

Values: String value "True" or "False".

Name: **Microsoft.Excel.AutoFitColumns.Adjust**

This setting is only applied when *Microsoft.Excel.AutoFitColumns* is set to "True" and allows you to add the width specified (in points) to all columns after they have been auto-fit. The maximum column width allowed in Excel is 255 points. It is not normally needed to add width to each column and adding width to each column can be a time-consuming operation; only use this option if absolutely needed.

Values: String value of the amount, in points, by which to adjust the column width.

Conversion Settings - Excel General Formatting & Printing Options

Name:	Microsoft.Excel.AutoFit.KeepEmbeddedChartScaling
	Only applies when auto-fit rows and columns is enabled. When set to its default of "True", autofit is not applied to any rows and/or columns that are under any embedded charts on the sheet. All other rows and columns are auto-fit. This allows the embedded charts to maintain the scale they were originally set at when placed on the spreadsheet. If set to "False", the chart will change size depending on the new height and width of the underlying rows and columns.
Values:	String value "True" or "False".
Name:	Microsoft.Excel.UnfreezePanels
	If the spreadsheet has any non-scrolling, "frozen" panels, pass "True" to unfreeze them before printing.
Values:	String value "True" or "False".
Name:	Microsoft.Excel.ClearFormatsOnEmptyRowsOnTop
	Clears the formatting of any empty rows (cells with no data) at the top of the spreadsheet so that only rows with data in them are printed.
Values:	String value "True" or "False".
Name:	Microsoft.Excel.ClearFormatsOnEmptyRowsOnBottom
	Clears the formatting of any empty rows (cells with no data) at the bottom of the spreadsheet so that only rows with data in them are printed.
Values:	String value "True" or "False".
Name:	Microsoft.Excel.ClearFormatsOnEmptyColumnsOnLeft
	Clears the formatting of any empty columns (cells with no data) on the left hand side of the spreadsheet so that only columns with data in them are printed.
Values:	String value "True" or "False".
Name:	Microsoft.Excel.ClearFormatsOnEmptyColumnsOnRight
	Clears the formatting of any empty columns (cells with no data) on the right hand side of the spreadsheet so that only columns with data in them are printed.
Values:	String value "True" or "False".

Conversion Settings - Excel General Formatting & Printing Options

Name: **Microsoft.Excel.RemoveBackgroundColors**

Clears the background colors and fills for all cells. Leaves text color and borders unchanged.

Note: This does not apply to cells that have conditional formatting applied.

Values: String value "True" or "**False**".

Name: **Microsoft.Excel.SetAllTextAsBlack**

Sets all text to black.

Note: This does not apply to cells that have conditional formatting applied.

Values: String value "True" or "**False**".

Name: **Microsoft.Excel.ClearTableStyle**

Clears the table styling from any columns or rows in the spreadsheet. Leaves the cell data, formatting and formulas in place. This can be a time-consuming operation as the table formatting is copied to each cell; only use this option if absolutely needed. To do the same but also remove the formatting, use *Microsoft.Excel.ClearTableStyleAndFormatting*.

Values: String value "True" or "**False**".

Name: **Microsoft.Excel.ClearTableStyleAndFormatting**

Clears the table styling and any table formatting from any columns or rows in the spreadsheet. Leaves the cell data and formulas in place.

Values: String value "True" or "**False**".

Name: **Microsoft.Excel.ClearAllConditionalFormatting**

Clears all conditional formatting applied to any cells. This includes removing background colors and text styling, color scales, data bars and icon sets.

Note: This does not apply to any spreadsheet that is protected or shared.

Values: String value "True" or "**False**".

Conversion Settings - Excel General Formatting & Printing Options

Name: **Microsoft.Excel.TrackChanges.HighlightChangesOnScreen**

If *Track Changes* has been enabled for the workbook, any cell on any spreadsheet that has been changed will be highlighted.

Values: String value "True" or "False".

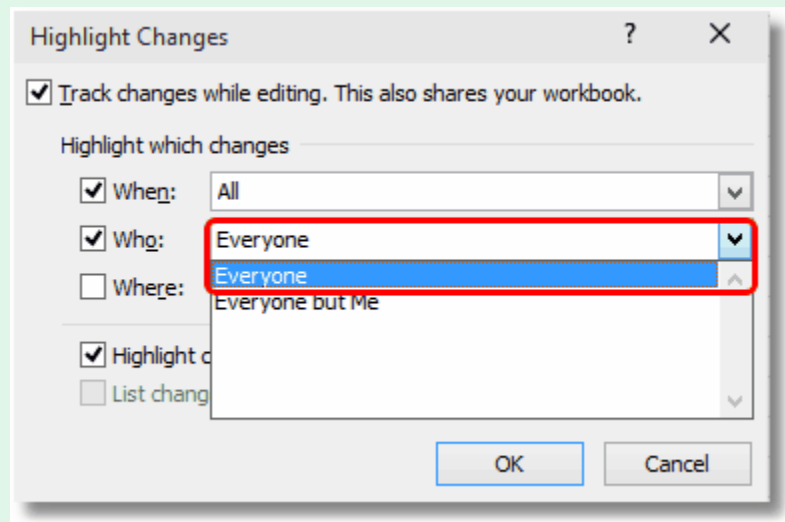
Name: **Microsoft.Excel.TrackChanges.ListChangesOnNewSheet**

If *Track Changes* has been enabled for the workbook, setting this to *True* will create a new temporary, protected spreadsheet that lists all of changes made to the workbook. If not using the English version of Excel, *Microsoft.Excel.TrackChanges.ExcelTrackChangesWhoParameter* will also need to be set.

Values: String value "True" or "False".

Name: **Microsoft.Excel.TrackChanges.ExcelTrackChangesWhoParameter**

When using an Office installation in a language other than English, this option must specify the word "Everyone" in that language to list the tracked changes for all users. The default for this setting is "Everyone". The 5 most common languages are listed below, or you can find the needed parameter on the Highlight Changes dialog in your version of Excel. The English version is shown below.



Values: English - **Everyone**
 French - Tous, Tout le monde
 Italian - Tutti
 German - Jeder
 Spanish - Todos

Conversion Settings - Excel Page Setup Printing Options

Name:	Microsoft.Excel.PageSetup.AlignMarginsHeaderFooter
	Have Excel align the header and the footer with the margins set in the page setup options.
Values:	String value "True" or "False".
Name:	Microsoft.Excel.PageSetup.BlackAndWhite
	Print the Excel document in black and white.
Values:	String value "True" or "False".
Name:	Microsoft.Excel.PageSetup.BottomMargin
	Set the size of the bottom margin in points.
Values:	String value of the desired margin height.
Name:	Microsoft.Excel.PageSetup.CenterFooter
	The text to display in the center footer area of the worksheet.
Values:	String value of the text to display.
Name:	Microsoft.Excel.PageSetup.CenterHeader
	The text to display in the center header area of the worksheet.
Values:	String value of the text to display.
Name:	Microsoft.Excel.PageSetup.CenterHorizontally
	Center the worksheet horizontally on the page when printed.
Values:	String value "True" or "False".
Name:	Microsoft.Excel.PageSetup.CenterVertically
	Center the worksheet vertically on the page when printed.
Values:	String value "True" or "False".

Conversion Settings - Excel Page Setup Printing Options	
Name:	Microsoft.Excel.PageSetup.DifferentFirstPageHeaderFooter
	If this is <i>True</i> a different header or footer is used for the first page of the worksheet (<i>applies to Office 2007 or higher</i>).
Values:	String value "True" or "False".
Name:	Microsoft.Excel.PageSetup.Draft
	Prints the worksheet without graphics when set to <i>True</i> .
Values:	String value "True" or "False".
Name:	Microsoft.Excel.PageSetup.FirstPageNumber
	Sets the first page number that will be used when this sheet is printed.
Values:	String value of the page number to start with.
Name:	Microsoft.Excel.PageSetup.FitToPagesTall
	Set the number of pages tall the worksheet will scale to when printed. Ignored when Microsoft.Excel.PageSetup.Zoom is set to <i>True</i> .
Values:	String value of the number of pages tall to use or "False" to use the scaling set in the Microsoft.Excel.PageSetup.FitToPagesWide setting.
Name:	Microsoft.Excel.PageSetup.FitToPagesWide
	Set the number of pages wide the worksheet will scale to when printed. Ignored when Microsoft.Excel.PageSetup.Zoom is set to <i>True</i> .
Values:	String value of the number of pages wide to use or "False" to use the scaling set in the Microsoft.Excel.PageSetup.FitToPagesTall setting.
Name:	Microsoft.Excel.PageSetup.FooterMargin
	Sets the distance, in points, from the bottom of the page to the footer.
Values:	String value of the desired margin height.

Conversion Settings - Excel Page Setup Printing Options	
Name:	Microsoft.Excel.PageSetup.HeaderMargin
	Sets the distance, in points, from the top of the page to the header.
Values:	String value of the desired margin height.
Name:	Microsoft.Excel.PageSetup.LeftFooter
	The text to display in the left footer area of the worksheet.
Values:	String value of the text to display.
Name:	Microsoft.Excel.PageSetup.LeftHeader
	The text to display in the left header area of the worksheet.
Values:	String value of the text to display.
Name:	Microsoft.Excel.PageSetup.LeftMargin
	Set the size of the left margin in points.
Values:	String value of the desired margin height.
Name:	Microsoft.Excel.PageSetup.OddAndEvenPagesHeaderFooter
	Set to <i>True</i> if different headers and footers have been set for odd-numbered and even-numbered pages.
Values:	String value "True" or "False".
Name:	Microsoft.Excel.PageSetup.Order
	Choose the page order when printing multiple spreadsheet pages per page.
Values:	DownThenOver - print the spreadsheet pages down then across the page. OverThenDown - print the spreadsheet pages across the page, then down.
Name:	Microsoft.Excel.PageSetup.Orientation
	Choose the orientation of the Excel spreadsheet.
Values:	Landscape Portrait

Conversion Settings - Excel Page Setup Printing Options

Name: Microsoft.Excel.PageSetup.PaperSize

Sets the size of the paper the worksheet will be printed on.

Values:

Paper10x14 - 10 in. x 14 in.
 Paper11x17 - 11 in. x 17 in.
 PaperA3 - A3 (297 mm x 420 mm)
 PaperA4 - A4 (210 mm x 297 mm)
 PaperA4Small - A4 Small (210 mm x 297 mm)
 PaperA5 - A5 (148 mm x 210 mm)
 PaperB4 - B4 (257 mm x 364 mm)
 PaperB5 - B5 (182 mm x 257 mm)
 PaperCsheet - C size sheet
 PaperDsheet - D size sheet
 PaperEnvelope10 - Envelope #10 (4-1/8 in. x 9-1/2 in.)
 PaperEnvelope11 - Envelope #11 (4-1/2 in. x 10-3/8 in.)
 PaperEnvelope12 - Envelope #12 (4-1/2 in. x 11 in.)
 PaperEnvelope14 - Envelope #14 (5 in. x 11-1/2 in.)
 PaperEnvelope9 - Envelope #9 (3-7/8 in. x 8-7/8 in.)
 PaperEnvelopeB4 - Envelope B4 (250 mm x 353 mm)
 PaperEnvelopeB5 - Envelope B5 (176 mm x 250 mm)
 PaperEnvelopeB6 - Envelope B6 (176 mm x 125 mm)
 PaperEnvelopeC3 - Envelope C3 (324 mm x 458 mm)
 PaperEnvelopeC4 - Envelope C4 (229 mm x 324 mm)
 PaperEnvelopeC5 - Envelope C5 (162 mm x 229 mm)
 PaperEnvelopeC6 - Envelope C6 (114 mm x 162 mm)
 PaperEnvelopeC65 - Envelope C65 (114 mm x 229 mm)
 PaperEnvelopeDL - Envelope DL (110 mm x 220 mm)
 PaperEnvelopeItaly - Envelope (110 mm x 230 mm)
 PaperEnvelopeMonarch - Envelope Monarch (3-7/8 in. x 7-1/2 in.)
 PaperEnvelopePersonal - Envelope (3-5/8 in. x 6-1/2 in.)
 PaperEsheet - E size sheet
 PaperExecutive - Executive (7-1/2 in. x 10-1/2 in.)
 PaperFanfoldLegalGerman - German Legal Fanfold (8-1/2 in. x 12 in.)
 PaperFanfoldStdGerman - German Legal Fanfold (8-1/2 in. x 13 in.)
 PaperFanfoldUS - U.S. Standard Fanfold (14-7/8 in. x 11 in.)
 PaperFolio - Folio (8-1/2 in. x 13 in.)
 PaperLedger - Ledger (17 in. x 11 in.)
 PaperLegal - Legal (8-1/2 in. x 14 in.)
 PaperLetter - Letter (8-1/2 in. x 11 in.)
 PaperLetterSmall - Letter Small (8-1/2 in. x 11 in.)
 PaperNote - Note (8-1/2 in. x 11 in.)
 PaperQuarto - Quarto (215 mm x 275 mm)
 PaperStatement - Statement (5-1/2 in. x 8-1/2 in.)
 PaperTabloid - Tabloid (11 in. x 17 in.)

Conversion Settings - Excel Page Setup Printing Options	
Name:	Microsoft.Excel.PageSetup.PrintArea
	Sets the range to be printed, as a string using Excel's A1-style references.
Values:	String containing the print area. Pass an empty string to print the entire worksheet.
Name:	Microsoft.Excel.PageSetup.PrintComments
	Determines where any comments in the worksheet are printed.
Values:	PrintSheetEnd - print the comments as notes at the end of the worksheet. PrintInPlace - comments are printed in-place in the worksheet as pop-up notes. PrintNoComments - comments are not printed.
Name:	Microsoft.Excel.PageSetup.PrintErrors
	Set the type of print error displayed.
Values:	PrintErrorsDisplayed - display all print errors. PrintErrorsBlank - print errors are blank. PrintErrorsDash - display print errors as dashes. PrintErrorsNA - display print errors as not available.
Name:	Microsoft.Excel.PageSetup.PrintGridlines
	If set to <i>True</i> then grid lines will be printed on each spreadsheet.
Values:	String value "True" or "False".
Name:	Microsoft.Excel.PageSetup.PrintHeadings
	If set to <i>True</i> then column and row headings will be printed on each spreadsheet.
Values:	String value "True" or "False".
Name:	Microsoft.Excel.PageSetup.PrintNotes
	Set to <i>True</i> to print cell notes as end notes with the worksheet.
Values:	String value "True" or "False".
Name:	Microsoft.Excel.PageSetup.PrintQuality
	Sets the print quality, or DPI, of the worksheet. This is different from the <i>DevMode settings;Resolution</i> setting in the Devmode settings section.

Conversion Settings - Excel Page Setup Printing Options	
Values:	1200, 720, 600, 400, 360, 300, 240, 200, 150, 120, 100, 75, 60, 50
Name:	Microsoft.Excel.PageSetup.PrintTitleColumns
	Sets the columns that contain the cells to be repeated on the left side of each page as a string using Excel's A1-style references.
Values:	String containing the columns to use as title columns. Pass an empty string to turn off title columns.
Name:	Microsoft.Excel.PageSetup.PrintTitleRows
	Sets the rows that contain the cells to be repeated on the top of each page as a string using Excel's A1-style references.
Values:	String containing the rows use as title rows. Pass an empty string to turn off title rows.
Name:	Microsoft.Excel.PageSetup.RightFooter
	The text to display in the right footer area of the worksheet.
Values:	String value of the text to display.
Name:	Microsoft.Excel.PageSetup.RightHeader
	The text to display in the right header area of the worksheet.
Values:	String value of the text to display.
Name:	Microsoft.Excel.PageSetup.RightMargin
	Set the size of the left margin in points.
Values:	String value of the desired margin width.
Name:	Microsoft.Excel.PageSetup.ScaleWithDocHeaderFooter
	If set to <i>True</i> then the header and footer will be scaled with the document when the size of the document changes.
Values:	String value "True" or "False".

Conversion Settings - Excel Page Setup Printing Options

Name: **Microsoft.Excel.PageSetup.TopMargin**

Set the size of the top margin in points.

Values: String value of the desired margin height.

Name: **Microsoft.Excel.PageSetup.Zoom**

Sets a percentage (between 10 and 400 percent) by which the worksheet will be scaled when printed.

Values: String value representing the zoom percentage, or "False" to use the Microsoft.Excel.PageSetup.FitToPagesTall and Microsoft.Excel.PageSetup.FitToPagesWide properties instead.

Conversion Settings - Excel Field Replacement

Name: **Microsoft.Excel.ReplaceFieldDateWith**

Replaces any DATE fields in the header and footer in the Excel document with the provided string.

Values: The string value to place in the field.

Name: **Microsoft.Excel.ReplaceFieldTimeWith**

Replaces any TIME fields in the header and footer in the Excel document with the provided string.

Values: The string value to place in the field.

Name: **Microsoft.Excel.ReplaceFieldFileNameWith**

Replaces any FILENAME fields in the header and footer in the Excel document with the provided string.

Values: A string value to replace the auto file name field.

Conversion Settings - Excel Field Replacement

Name:	Microsoft.Excel.ReplaceFormulasWithAutoDateAndTimeAsString
	Replaces any cells containing a formula with the functions TODAY() and NOW() with the provided string. This will replace the entire cell formula.
Values:	A string value to display as the cell contents.

Conversion Settings - Document Protection

Name:	Microsoft.Excel.UnprotectPassword
	The password is used to unprotect the Excel document and allow changes. This password is passed as clear text and is visible to anyone.
Values:	A string value containing the password.
Name:	Microsoft.Excel.OpenPassword
	The password is used to open a password-protected Excel document. This password is passed as clear text and is visible to anyone.
Values:	A string value containing the password.
Name:	Microsoft.Excel.WritePassword
	The password is used to allow saving changes to the Excel document. This password is passed as clear text and is visible to anyone.
Values:	A string value containing the password.
Name:	Microsoft.Excel.RemoveDocumentProtection
	Does not apply to Excel 2013 and later versions.
	Temporarily remove any workbook or spreadsheet protection that may be set on the document. This allows Excel printing and formatting options to be applied.
Values:	String value "True" or "False". Default is True for Excel 2010 and previous versions. Ignored for Office 2013 and later.

Conversion Settings - Document Protection

Name: **Microsoft.Excel.SkipFileValidation**

Office File Validation is a security feature added starting with Microsoft Office 2010. This feature checks Office files created with older versions to ensure they were safe to open before actually opening them. Files can be marked as invalid if they are corrupt or contain malicious code. Unfortunately, this can also mean that files created previous versions of Office can mistakenly be tagged as invalid when they are not. You can use this setting to disable this feature.

We do not recommend enabling this feature; you do so at your own risk. Use with caution and only disable if you know and trust the source of the Excel files.

Values: **False** - Files are always validated upon opening.
True - Skip file validation upon opening. ***This setting is not recommended.***

Header and Footer Formatting Codes

The following formatting codes are used to customize the header and footer contents of the spreadsheet with page numbers, the date, the name of the sheet, or the name and path of the file taken from the Excel file being converted.

Applies to these settings:

- Microsoft.Excel.PageSetup.LeftHeader
- Microsoft.Excel.PageSetup.CenterHeader
- Microsoft.Excel.PageSetup.RightHeader
- Microsoft.Excel.PageSetup.LeftFooter
- Microsoft.Excel.PageSetup.CenterFooter
- Microsoft.Excel.PageSetup.RightFooter

These formatting codes are applied to the header and footer contents **after** any auto date, time or filename replacement is applied from the settings *Microsoft.Excel.ReplaceFieldDateWith*, *Microsoft.Excel.ReplaceFieldTimeWith*, and *Microsoft.Excel.ReplaceFieldFileNameWith*.

This means that if you use an autodate, autotime or file name formatting code in a custom header, you will get the autodate, autotime or file name in the header or footer, and not the replacement string.

&P	Current page number
&N	Number of pages

&D	Auto date
&T	Auto time
&Z&F	Path to file
&F	File name
&A	Sheet name

PowerPoint Converter Options

These options control the behavior of the PowerPoint converter used by Document Conversion Service. Table values in **bold** text are the default value for that setting. Not all settings have default values; these settings are optional and the appropriate setting in the presentation being printed will be used.

Sample Profile

```
<?xml version="1.0" encoding="utf-8"?>
<Profile Type="0"
    DisplayName="TIFF 300dpi Powerpoint Markup, replace dates "
    Description="Prints Powerpoint slides with comments and tracking visible.">
    <Settings>

        <add Name ="Microsoft.PowerPoint.PageSetup.FirstSlideNumber" Value="2"/>
        <add Name ="Microsoft.PowerPoint.PageSetup.NotesOrientation"
            Value="OrientationVertical"/>
        <add Name ="Microsoft.PowerPoint.PrintOptions.FitToPage" Value="True"/>

        <!-- Output file options -->
        <add Name="Devmode settings;Resolution" Value="300"/>
        <add Name="Save;Output File Format" Value="TIFF Multipaged"/>
        ...

    </Settings>
</Profile>
```

Code Sample - C#

```
PNDocConvQueueServiceLib.PNDocConvQueueItem item = null;

// Create the conversion item
item = new PNDocConvQueueServiceLib.PNDocConvQueueItem();

// Set conversion settings
item.Set("Microsoft.PowerPoint.PageSetup.FirstSlideNumber", "2");
item.Set("Microsoft.PowerPoint.PageSetup.NotesOrientation",
    "OrientationVertical");
item.Set("Microsoft.PowerPoint.PrintOptions.FitToPage",
    "True");
item.Set("Devmode settings;Resolution", "300");
item.Set("Save;Output File Format", "TIFF Multipaged");
...
// convert the file
item.Convert("Microsoft PowerPoint", _
    @"C:\Test\Report.pptx", _
    @"C:\Test\Out\ConvertedPresentation");
```


**Code Sample - VB.NET**

```

Dim item As PNDocConvQueueServiceLib.IPNDocConvQueueItem

' Create the conversion item
item = New PNDocConvQueueServiceLib.PNDocConvQueueItem()

' Set conversion settings
item.Set("Microsoft.PowerPoint.PageSetup.FirstSlideNumber", "2")
item.Set("Microsoft.PowerPoint.PageSetup.NotesOrientation",
"OrientationVertical")
item.Set("Microsoft.PowerPoint.PrintOptions.FitToPage",
"True")
item.Set("Devmode settings;Resolution", "300")
item.Set("Save;Output File Format", "TIFF Multipaged")
...
' convert the file
item.Convert("Microsoft PowerPoint", _
"C:\Test\Report.pptx", _
"C:\Test\Out\ConvertedPresentation")

```

Conversion Settings - PowerPoint Page Setup

Name: **Microsoft.PowerPoint.PageSetup.FirstSlideNumber**

Sets the slide number for the first slide in the presentation.

Values: String value containing the starting number, such as "2".

Name: **Microsoft.PowerPoint.PageSetup.NotesOrientation**

Sets the printed orientation of notes pages, handouts, and outlines for the specified presentation. If the value passed down does not match the strings below, the orientation will default to OrientationHorizontal.

Values: OrientationHorizontal
OrientationVertical
OrientationMixed

Name: **Microsoft.PowerPoint.PageSetup.SlideOrientation**

Sets the printed orientation of slides in the presentation. If the value passed down does not match the strings below, the orientation will default to OrientationHorizontal.

Values: OrientationHorizontal
OrientationVertical
OrientationMixed

Conversion Settings - PowerPoint Page Setup	
Name:	Microsoft.PowerPoint.PageSetup.SlideHeight
	Sets the height of the slide in points.
Values:	String value of the desired slide height.
Name:	Microsoft.PowerPoint.PageSetup.SlideSize
	Sets the slide size for the specified presentation
Values:	SlideSizeOnScreen - On Screen SlideSizeLetterPaper - Letter Paper SlideSizeA4Paper - A4 Paper SlideSize35MM - 35MM SlideSizeOverhead - Overhead SlideSizeBanner - Banner SlideSizeLedgerPaper - Ledger Paper SlideSizeA3Paper - A3 Paper SlideSizeB4ISOPaper - B4 ISO Paper SlideSizeB5ISOPaper - B5 ISO Paper SlideSizeB4JISPaper - B4 JIS Paper SlideSizeB5JISPaper - B5 JIS Paper SlideSizeHagakiCard - Hagaki Card
Name:	Microsoft.PowerPoint.PageSetup.SlideWidth
	Sets the width of the slide in points.
Values:	String value of the desired slide width.

Conversion Settings - PowerPoint Print Options	
Name:	Microsoft.PowerPoint.PrintOptions.FitToPage
	If set to "True" then the slides will be scaled to fill the page they're printed on.
Values:	String value "True" or "False".
Name:	Microsoft.PowerPoint.PrintOptions.FrameSlides
	If set to "True" then a thin frame is placed around the border of the printed slides.
Values:	String value "True" or "False".

Conversion Settings - PowerPoint Print Options

Name: **Microsoft.PowerPoint.PrintOptions.HandoutOrder**

Sets the page layout order for printed handouts that show multiple slides on one page.

Values: PrintHandoutVerticalFirst
PrintHandoutHorizontalFirst

Name: **Microsoft.PowerPoint.PrintOptions.HighQuality**

If set to "True" then the slides will be printed in high quality.

Values: String value "True" or "False".

Name: **Microsoft.PowerPoint.PrintOptions.OutputType**

Sets which component (slides, handouts, notes pages, or an outline) of the presentation is to be printed, and in the case of handouts, how many slides per page.

Values: PrintOutputSlides - print slides only.
PrintOutputNotesPages - prints slides with notes.
PrintOutputOutline - outline only.
PrintOutputBuildSlides - build slides only (Office 2003 and 2007 only).
PrintOutputOneSlideHandouts - handouts with a single slide per page.
PrintOutputTwoSlideHandouts - handouts with two slides per page.
PrintOutputThreeSlideHandouts - handouts with three slides per page.
PrintOutputFourSlideHandouts - handouts with four slides per page.
PrintOutputSixSlideHandouts - handouts with six slides per page.
PrintOutputNineSlideHandouts - handouts with nine slides per page.

Name: **Microsoft.PowerPoint.PrintOptions.PrintColorType**

Prints the presentation in one of black and white, in pure black and white (also referred to as high contrast), or in color.

Values: PrintColor
PrintBlackAndWhite
PrintPureBlackAndWhite

Conversion Settings - PowerPoint Print Options

Name:	Microsoft.PowerPoint.PrintOptions.PrintComments
	If set to "True" then any comments will be printed along with the slides in the presentation.
Values:	String value "True" or "False".
Name:	Microsoft.PowerPoint.PrintOptions.PrintFontsAsGraphics
	If set to "True" then any text created with TrueType fonts will be printed as graphics.
Values:	String value "True" or "False".
Name:	Microsoft.PowerPoint.PrintOptions.PrintHiddenSlides
	If set to "True" then any hidden slides in the presentation will also be printed.
Values:	String value "True" or "False".
Name:	Microsoft.PowerPoint.PrintOptions.SlideShowName
	Sets the name of the custom slide show to print.
Values:	A string value containing the name of the custom slide show in the presentation.

Conversion Settings - Document Protection

Name:	Microsoft.PowerPoint.OpenPassword
	The password is used to open a password-protected PowerPoint presentation. This password is passed as clear text and is visible to anyone.
Values:	A string value containing the password.

Adobe Reader Options

These options control the behavior of the Adobe Reader converter used by Document Conversion Service. Table values in **bold** text are the default value for that setting.



Sample Profile

```
<?xml version="1.0" encoding="utf-8"?>
<Profile Type="0"
  DisplayName="TIFF 300dpi Adobe with Markup"
  Description="Prints Adobe files with any stamps and markup visible.">
  <Settings>

    <add Name ="Adobe.PDF.PrintOptions.CommentsAndForms"
      Value="DocumentsAndMarkups"/>

    <!-- Output file options -->
    <add Name="Devmode settings;Resolution" Value="300"/>
    <add Name="Save;Output File Format" Value="TIFF Multipaged"/>
    ...

  </Settings>
</Profile>
```



Code Sample - C#

```
PNDocConvQueueServiceLib.PNDocConvQueueItem item = null;

// Create the conversion item
item = new PNDocConvQueueServiceLib.PNDocConvQueueItem();

// Set conversion settings
item.Set("Adobe.PDF.PrintOptions.CommentsAndForms", "DocumentsAndMarkups");

item.Set("Devmode settings;Resolution", "300");
item.Set("Save;Output File Format", "TIFF Multipaged");
...
// convert the file
item.Convert("Adobe Acrobat Reader", _
  @"C:\Test\Report.pdf", _
  @"C:\Test\Out\ConvertedPDF");
```



Code Sample - VB.NET

```
Dim item As PNDocConvQueueServiceLib.IPNDocConvQueueItem

' Create the conversion item
item = New PNDocConvQueueServiceLib.PNDocConvQueueItem()

' Set conversion settings
item.Set("Adobe.PDF.PrintOptions.CommentsAndForms", "DocumentsAndMarkups")
item.Set("Devmode settings;Resolution", "300")
item.Set("Save;Output File Format", "TIFF Multipaged")
...
' convert the file
item.Convert("Adobe Acrobat Reader", _
  "C:\Test\Report.pdf", _
  "C:\Test\Out\ConvertedPDF")
```

Conversion Settings - Adobe Reader Print Options

Name: **Adobe.PDF.PrintOptions.CommentsAndForms**

Choose what is visible on the page when the PDF file is printed. Markup consists of any comments and annotations, including stamps, that have been placed on the PDF.

Values: **DocumentsAndMarkups** - prints the document with any markup and stamps visible.
DocumentsAndStamps - prints the document with only stamp annotations visible. Markup is not shown
Documents - prints only the document. Markup and stamps are not printed.

Name: **Adobe.PDF.PrintOptions.ChoosePaperSourceByPDFPageSize**

When "True", Adobe will use the page size of each page in the PDF to determine the paper size of the output page (paper source); in this case the page size of the output images will match the original PDF document. If you are controlling the paper size using the [Devmode settings:Paper Size](#) setting, this option should be set to *false*. This will tell Adobe to scale the pages to the new paper size. This option is enabled (set to "True") by default.

Values: String value "True" or "False".

Name: **Adobe.PDF.PrintOptions.PageAutoRotate**

When "True", the PDF page will be rotated to fit the output page orientation when needed. Use when *Adobe.PDF.PrintOptions.ChoosePaperSourceByPDFPageSize* is set to "False". This option is disabled (set to "False") by default.

Values: String value "True" or "False".

Name: **Adobe.PDF.PrintOptions.PageScaling**

Choose how the PDF page will be scaled to the output page. Use when *Adobe.PDF.PrintOptions.ChoosePaperSourceByPDFPageSize* is set to "False". This option is set to "ShrinkToFit" by default.

Note: This option applies only when using Adobe Reader with the Adobe Reader converter; if using Adobe Acrobat, this option is not recognized.

Values: **ActualSize** - prints the PDF page at its original page size. If the output page is smaller than the original PDF page size, the page may be cropped.
ShrinkToFit - PDF pages that are larger than the output page size will be scaled to fit on the page; smaller pages are not scaled and are centered on the larger page. This is the default value.

Conversion Settings - Adobe Reader Print Options

Name: **Adobe.PDF.PrintOptions.PrintAsImage**

Choose how the PDF page will be printed. This option is enabled (set to "True") by default as it produces the best quality output.

Values: String value "True" or "False".

Name: **Adobe.PDF.PrintOptions.PrintCommentPopups**

Set to true to also print comment popups when printing with Adobe.PDF.PrintOptions.CommentsAndForms set to DocumentsAndMarkups. The comments must be open to be printed, otherwise only the comment icon is printed. Valid for Adobe Reader version 10 and higher.

Values: String value "True" or "False".

Name: **Adobe.PDF.PrintOptions.AllowDuplexPrintJobs**

Allows PDF files set with duplex printing options to successfully convert.

An empty blank page created by the Adobe printing engine will be added to the end of any documents with an odd number of pages. Setting to "False" will cause the file to fail to convert.

Values: String value "True" or "False".

Name: **Adobe.PDF.PrintOptions.IgnoreDuplexPrintingOptions**

Ignores any duplex (double-sided, FlipOnLongEdge, FlipOnShortEdge) printing options set in the PDF file. The file is converted single-sided. Overrides the **Adobe.PDF.PrintOptions.AllowDuplexPrintJobs** setting, if set. Does not apply to password-protected PDF files.

Values: String value "True" or "False".

Conversion Settings - Adobe Reader JavaScript Options

Name: **Adobe.PDF.Javascript.Enable**

Enable or disable any JavaScript in the PDF document. This option is disabled (set to "False") by default as JavaScript in PDF files can be a security risk. If your PDF files contain JavaScript that you need to have run to display the file properly, you can enable JavaScript processing by setting this options to "True".

Values: String value "True" or "False".

Conversion Settings - Adobe Reader General Options

Name: **Adobe.PDF.IgnoreSecurity**

Available starting in DCS 3.0.016.

This setting ignores, if possible, any security and passwords set on the PDF file, allowing the PDF file to be converted. PDF files with both user and owner passwords will still fail to convert. This option is enabled (set to "True") by default.

Values: String value "True" or "False".

Name: **Adobe.PDF.CreateTempCopyOnRetry**

Available starting in DCS 3.0.016.

When *True*, this setting will attempt to copy this PDF to a new temporary PDF for processing. For badly formed PDF files this can sometimes repair issues that prevent the file from opening and/or converting. This option is enabled (set to "True") by default.

Values: String value "True" or "False".

Internet Explorer Options

These options control the behavior of the Internet Explorer converter used by Document Conversion Service. Table values in **bold** text are the default value for that setting.

The default Internet Explorer options are to print no headers or footer information, use margins of 0.75", to print all background color and images and to shrink the page to fit. See [Adding Headers, Footers and Fonts to HTML Conversion](#) for instruction on customizing the Internet Explorer converter settings.

There are also application level Internet Explorer settings to control image scaling and browser emulation; see [Application Level Configuration Settings](#) to change these options.



Sample Profile

```
<?xml version="1.0" encoding="utf-8"?>
<Profile Type="0"
  DisplayName="Internet Explorer TIFF 300dpi"
  Description="Created TIFF from IE with footer only.">
  <Settings>

    <!-- Add footer with URL in center -->
    <add Name="Microsoft.InternetExplorer.PageSetup.Footer"
      Value="&amp;b&amp;u&amp;b"/>

    <!-- Add 0.50 inch margins -->
    <add Name="Microsoft.InternetExplorer.PageSetup.MarginBottom"
      Value="0.50"/>
    <add Name="Microsoft.InternetExplorer.PageSetup.MarginLeft"
      Value="0.50"/>
    <add Name="Microsoft.InternetExplorer.PageSetup.MarginRight"
      Value="0.50"/>
    <add Name="Microsoft.InternetExplorer.PageSetup.MarginTop"
      Value="0.50"/>

    <!-- Output file options -->
    <add Name="Devmode settings;Resolution" Value="300"/>
    <add Name="Save;Output File Format" Value="TIFF Multipaged"/>
    ...

  </Settings>
</Profile>
```



Code Sample - C#

```
PNDocConvQueueServiceLib.PNDocConvQueueItem item = null;

// Create the conversion item
item = new PNDocConvQueueServiceLib.PNDocConvQueueItem();

// Set conversion settings
item.Set("Microsoft.InternetExplorer.PageSetup.Footer",
        "&b&u&b");
item.Set("Microsoft.InternetExplorer.PageSetup.MarginBottom", "0.50");
item.Set("Microsoft.InternetExplorer.PageSetup.MarginLeft", "0.50");
item.Set("Microsoft.InternetExplorer.PageSetup.MarginRight", "0.50");
item.Set("Microsoft.InternetExplorer.PageSetup.MarginTop", "0.50");
...
// convert the file
item.Convert("Internet Explorer",
            @"C:\Test\ArchiveReport.mht",
            @"C:\Test\Out\ConvertedReport");
```



Code Sample - VB.NET

```
Dim item As PNDocConvQueueServiceLib.IPNDocConvQueueItem

' Create the conversion item
item = New PNDocConvQueueServiceLib.PNDocConvQueueItem()

' Set conversion settings
item.Set("Microsoft.InternetExplorer.PageSetup.Footer", "&b&u&b")
item.Set("Microsoft.InternetExplorer.PageSetup.MarginBottom", "0.50")
item.Set("Microsoft.InternetExplorer.PageSetup.MarginLeft", "0.50")
item.Set("Microsoft.InternetExplorer.PageSetup.MarginRight", "0.50")
item.Set("Microsoft.InternetExplorer.PageSetup.MarginTop", "0.50")
...
' convert the file
item.Convert("Internet Explorer", _
            "C:\Test\ArchiveReport.mht", _
            "C:\Test\Out\ConvertedReport")
```

Conversion Settings - Page Setup

Name:	Microsoft.InternetExplorer.PageSetup.Header
	The format of the header to print on each page. By default, no page header is printed.
Values:	If you do want a header when converting HTML files, follow the instructions here .

Conversion Settings - Page Setup	
Name:	Microsoft.InternetExplorer.PageSetup.Footer
	The format of the footer to print on each page. By default, no page footer is printed.
Values:	If you do want a footer when converting HTML files, follow the instructions here .
Name:	Microsoft.InternetExplorer.PageSetup.Font
	The font to use if printing headers and footers. The font is specified as follows, with text in bold specifying the font name, its point size and the color. The last two options, <i>font-style: italic</i> ; and <i>font-weight: bold</i> are optional and are only to be included if bold, italic, or bold and italic text is wanted.
Values:	String value containing the font definition. font-family: <name>; font-size: <size>pt; color: rgb(0,0,0); font-style: italic; font-weight: bold;
Name:	Microsoft.InternetExplorer.PageSetup.MarginBottom
	The bottom margin in inches. The default is 0.75.
Values:	String value of the desired margin height.
Name:	Microsoft.InternetExplorer.PageSetup.MarginLeft
	The left-hand side margin in inches. The default is 0.75.
Values:	String value of the desired margin width.
Name:	Microsoft.InternetExplorer.PageSetup.MarginRight
	The right-hand side margin in inches. The default is 0.75.
Values:	String value of the desired margin width.
Name:	Microsoft.InternetExplorer.PageSetup.MarginTop
	The top margin in inches. The default is 0.75.
Values:	String value of the desired margin height.

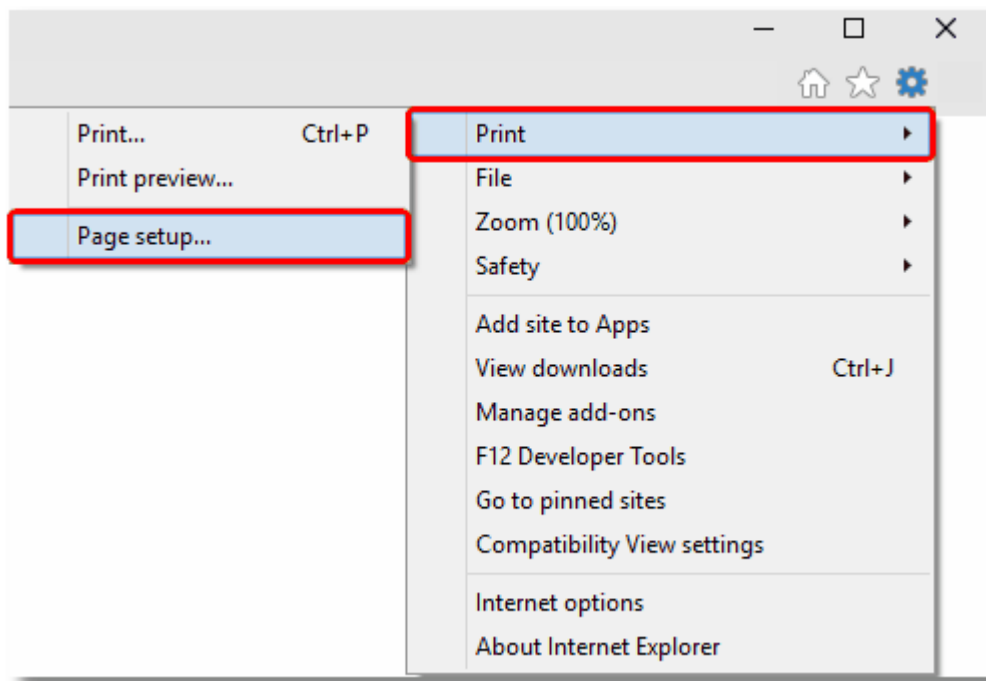
Conversion Settings - Page Setup

Name:	Microsoft.InternetExplorer.PageSetup.PrintBackground Determines if background colors and images are printed. By default, they are always printed.
Values:	String value "True" or "False".
Name:	Microsoft.InternetExplorer.PageSetup.ShrinkToFit Determines if the page is scaled to fit on the the printed page. By default it is always printed with Shrink-to-Fit enabled. By default, the minimum scale factor is 30, meaning the page will shrink to at most 30% of its original size to try and fit the contents on the page. If you need the page to be larger, this scaling factor can be customized in the <i>Internet Explorer</i> section in the <i>ApplicationFactory</i> section of the Document Conversion Service application configuration file. See also Application Level Configuration Settings . <pre><AppFactory Name="Internet Explorer" Type="PEERNET.PNDocConv.Applications.PNInternetExplorerApplicatio Assembly="PNInternetExplorerApplicationFactory"> <Settings> ... <add Name="ConverterPlugIn.PNIExplorer.ShrinkToFitScaleMin" Value="30" /> </Settings> </AppFactory></pre>
Values:	String value "True" or "False".

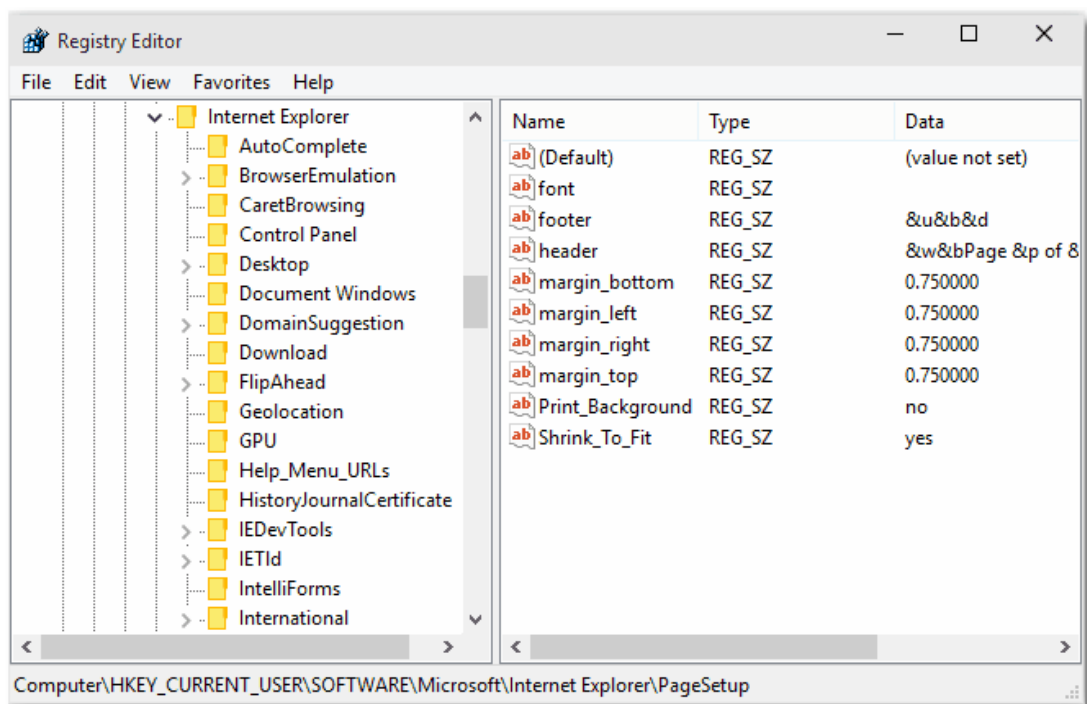
Adding Headers, Footers and Fonts to HTML Conversion

The simplest method to add header and footer information and font information is to use the *Page Setup* dialog in Internet Explorer to configure the margins, headers, footers and other page setup options and then copy these settings from the registry keys Internet Explorer uses to store this information.

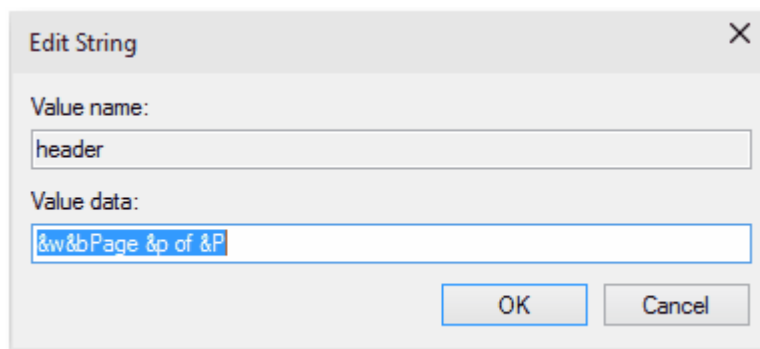
1. Open Internet Explorer to any web page or html file.
2. In the upper right corner, click the Tools icon (it looks like a blue gear), then select Print - Page Setup.
 - a. Alternatively you can press the F10 key to show the application menu and then select File - Page Setup.



3. In the Page Setup dialog, define your margins, any header and footer information, and optionally choose the font you want to use. Click OK, then exit Internet Explorer.
4. Open the registry using *RegEdit* (type regedit.exe into the Start menu search field or from the Start menu go to Programs - Accessories - Run and type regedit.exe).
5. In the registry editor, go to the HKEY_CURRENT_USER folder, then Software - Microsoft - Internet Explorer - PageSetup.



6. In the right-hand pane, double click any of the values to open the *Edit String* dialog box. From here you can copy and paste the header and footer formatted strings. When using these strings in the conversion profiles, any & characters need to be replaced with & for the string to be parsed correctly.



Application Level Configuration Settings

Document Conversion Service uses Internet Explorer to convert HTM, HTML and MHT files. When dealing with MHT and HTML files with large images, and older style HTML files formatted for earlier browser versions the options for image scaling and browser emulation may need to be configured to produce the desired output file.

These options are set in the Internet Explorer section of the application configuration file. Changing these options will require a restart of Document Conversion Service for the new settings to take effect.

Setting the Minimum Scale For Internet Explorer

HTML files and MHT files such as email messages from Outlook can sometimes have very wide images. By default, these files are always printed with Shrink-to-Fit enabled and a minimum scale factor of 30. This means that the page will shrink to at most 30% of its original size to fit the image contents on the page.

If you need the images to be scaled larger, the setting *ConverterPlugin.PNIExplorer.ShrinkToFitScaleMin* can be adjusted from between 30 to 100 to get the size of image you want.

This option is set at the application level and cannot be changed per file. Changes to this setting require a restart of Document Conversion Service to take effect.

Setting the Browser Emulation for Internet Explorer

In certain cases, older HTML files created for previous versions of Internet Explorer will not convert correctly when printed using the latest version of Internet Explorer. This is because Internet Explorer runs with *Edge compatibility* by default and it is this new compatibility and rendering that has a problem with the older style HTML.

If you have these type of files, the setting *ConverterPlugin.PNIExplorer.BrowserEmulation* can be used to force Internet Explorer to emulate older versions of the browser so that the files are rendered properly based on the older browsers rendering engine.

This option is set at the application level and cannot be changed per file. Changes to this setting require a restart of Document Conversion Service to take effect.



Configuration Section for Internet Explorer

```
<AppFactories>
  <Factories>

    <AppFactory Name="Internet Explorer"
      Type="PEERNET.PNDocConv.Applications.PNInternetExplorerApplicationFactory"
      Assembly="PNInternetExplorerApplicationFactory">
      <Settings>
        <add Name="Enabled" Value="auto"/>
        <add Name="MaxInstances" Value="auto"/>
        <add Name="RecycleThreshold" Value="0"/>
        <add Name="DocumentOpenTimeout" Value="360000"/>

        <!-- Value range 30 - 100 -->
        <add Name="ConverterPlugIn.PNIExplorer.ShrinkToFitScaleMin" Value="30"/>

        <!-- Values: Empty string, IE7, IE8, IE8FORCE, IE9, IE9FORCE, IE10, IE10FORCE, IE11 -->
        <add Name="ConverterPlugIn.PNIExplorer.BrowserEmulation" Value="" />

      </Settings>
    </AppFactory>

    ...

  </Factories>
</AppFactories>

<Settings>
  <!-- Global factory settings -->
  <add Name="MaxInstances" Value="auto"/>
  <add Name="RecycleThreshold" Value="0"/>
</Settings>
```


Ghostscript Converter Options

These options control the behavior of the Ghostscript converter used by Document Conversion Service. Table values in **bold** text are the default value for that setting.



Sample Profile

```
<?xml version="1.0" encoding="utf-8"?>
<Profile Type="0"
    DisplayName="Ghostscript Custom Font TIFF 300dpi"
    Description="Created TIFF from Ghostscript using custom fonts.">
  <Settings>

    <!-- Antialias options -->
    <add Name="ConverterPlugIn.PNGhostscriptConverter.TextAntiAlias" Value="4"/>
    <add Name="ConverterPlugIn.PNGhostscriptConverter.GraphicsAntiAlias" Value="4"/>

    <!-- Search these folders for matching fonts -->
    <add Name="ConverterPlugIn.PNGhostscriptConverter.FontPath"
        Value="C:\psfonts;c:\Windows\Fonts;C:\MyFonts"/>

    <!-- Output file options -->
    <add Name="Devmode settings;Resolution" Value="300"/>
    <add Name="Save;Output File Format" Value="TIFF Multipaged"/>
    ...

  </Settings>
</Profile>
```



Code Sample - C#

```
PNDocConvQueueServiceLib.PNDocConvQueueItem item = null;

// Create the conversion item
item = new PNDocConvQueueServiceLib.PNDocConvQueueItem();

// Set conversion settings
item.Set("ConverterPlugIn.PNGhostscriptConverter.TextAntiAlias", "4");
item.Set("ConverterPlugIn.PNGhostscriptConverter.Graphics", "4");

item.Set("ConverterPlugIn.PNGhostscriptConverter.FontPath",
    @"C:\psfonts;c:\Windows\Fonts;C:\MyFonts");
...
// convert the file
item.Convert("Ghostscript",
    @"C:\Test\ArchiveReport.ps",
    @"C:\Test\Out\ConvertedReport");
```



Code Sample - VB.NET

```
Dim item As PNDocConvQueueServiceLib.IPNDocConvQueueItem

' Create the conversion item
item = New PNDocConvQueueServiceLib.PNDocConvQueueItem()

' Set conversion settings
item.Set("ConverterPlugIn.PNGhostscriptConverter.TextAntiAlias", "4")
item.Set("ConverterPlugIn.PNGhostscriptConverter.Graphics", "4")

item.Set("ConverterPlugIn.PNGhostscriptConverter.FontPath", _
        "C:\psfonts;c:\Windows\Fonts;C:\MyFonts")

...
' convert the file
item.Convert("Ghostscript", _
            "C:\Test\ArchiveReport.ps", _
            "C:\Test\Out\ConvertedReport")
```

Name:	ConverterPlugIn.PNGhostscriptConverter.TextAntiAlias
	The size of the subsample box used when antialiasing text in the file. Antialiasing is used to improve the quality of the text on the page when converted to an image. A subsample box of 4 will produce the best result. The lower subsample values will increase the speed of conversion but can affect the image quality.
Values:	The size of the subsample box can be 4, 2 or 1. The default is 4.
Name:	ConverterPlugIn.PNGhostscriptConverter.GraphicsAntiAlias
	The size of the subsample box used when antialiasing graphics in the file. Antialiasing is used to improve the quality of any graphics on the page when converted to an image of a different resolution. A subsample box of 4 will produce the best result. The lower subsample values will increase the speed of conversion but can affect the image quality.
Values:	The size of the subsample box can be 4, 2 or 1. The default is 4.
Name:	ConverterPlugIn.PNGhostscriptConverter.FontPath
	By default, the special Windows <i>Fonts</i> folder and the folder c:\psfonts are used by Ghostscript to find the fonts used in the Postscript or PDF documents. You can override this setting by providing your own semicolon-separated list of folders in which to search.
Values:	String value containing a semi-colon separated list of folders.

Image Converter Options

These options control the behavior of the image converter used by Document Conversion Service. Table values in **bold** text are the default value for that setting.



Sample Profile

```
<?xml version="1.0" encoding="utf-8"?>
<Profile Type="0"
    DisplayName="Image Converter 300dpi"
    Description="Create images using the PEERNET image converter.">
  <Settings>

    <!-- Image Conversion options, use LEAD first -->
    <add Name="ConverterPlugIn.PNImageConverter.ImageToolkitOrder" Value="LEAD;WIC"/>
    <add Name="ConverterPlugIn.PNImageConverter.LEADScalingMode" Value="BICUBIC"/>
    <add Name="ConverterPlugIn.PNImageConverter.WICScalingMode" Value="BICUBIC"/>

    <!-- Background color for transparent images, white is default -->
    <add Name="ConverterPlugIn.PNImageConverter.AlphaBackgroundColorRGB" Value="255,255,255"

    <!-- Output file options -->
    <add Name="Devmode settings;Resolution" Value="300"/>
    <add Name="Save;Output File Format" Value="TIFF Multipaged"/>
    ...

  </Settings>
</Profile>
```



Code Sample - C#

```
PNDocConvQueueServiceLib.PNDocConvQueueItem item = null;

// Create the conversion item
item = new PNDocConvQueueServiceLib.PNDocConvQueueItem();

// Set conversion settings
item.Set("ConverterPlugIn.PNImageConverter.ImageToolkitOrder", "LEAD;WIC");
item.Set("ConverterPlugIn.PNImageConverter.LEADScalingMode", "BICUBIC");
item.Set("ConverterPlugIn.PNImageConverter.WICScalingMode", "BICUBIC");

// Background color for transparent images, white is default
item.Set("ConverterPlugIn.PNImageConverter.AlphaBackgroundColorRGB", "255,255,255");

// Output file options
item.Set("Devmode settings;Resolution", "300");
item.Set("Save;Output File Format", "TIFF Multipaged");

...
// convert the file
item.Convert("PEERNET Image Converted",
    @"C:\Test\screenshot.png",
    @"C:\Test\Out\ConvertedImage");
```



Code Sample - VB.NET

```
Dim item As PNDocConvQueueServiceLib.IPNDocConvQueueItem

' Create the conversion item
item = New PNDocConvQueueServiceLib.PNDocConvQueueItem()

' Set conversion settings
item.Set("ConverterPlugIn.PNImageConverter.ImageToolkitOrder", "LEAD;WIC")
item.Set("ConverterPlugIn.PNImageConverter.LEADScalingMode", "BICUBIC")
item.Set("ConverterPlugIn.PNImageConverter.WICScalingMode", "BICUBIC")

' Background color for transparent images, white is default
item.Set("ConverterPlugIn.PNImageConverter.AlphaBackgroundColorRGB", "255,255,255")

item.Set("Devmode settings;Resolution", "300")
item.Set("Save;Output File Format", "TIFF Multipaged")

...
' convert the file
item.Convert("PEERNET Image Converted", _
            "C:\Test\screenshot.png", _
            "C:\Test\Out\ConvertedImage")
```

Conversion Settings - Toolkits and Scaling Modes

Name:	ConverterPlugIn.PNImageConverter.ImageToolkitOrder This string lists, in the order in which they will be used, the image tool kits that PEERNET Image Converter will use to try and convert an image. The default value, "LEAD;WIC", will use LEAD first and then try WIC (W indows I maging C omponent) if the image could not be converted. The two tool kits support opening and reading different file formats; see Supported Image File Formats below for a complete list. You do not need to install anything extra to use these either of these tool kits. The LEAD tool kit is bundled with Document Conversion Service and the Windows Image Component is part of the Windows operating system.
Values:	LEAD;WIC - use LEAD first, then try WIC if the image could not be converted. WIC;LEAD - use WIC first, then try LEAD if the image could not be converted. LEAD - only use LEAD. WIC - only use WIC.

Conversion Settings - Toolkits and Scaling Modes

Name: **ConverterPlugIn.PNImageConverter.LEADScalingMode**

This is the sampling or filtering mode to use when scaling an image. An image needs to be scaled when the resolution of the source image and destination image are not the same.

Values: **NORMAL** - Nearest neighbor, this is the fastest mode and often can produce the smallest image.
LINEAR - A linear interpolation algorithm, slower than **NORMAL** but better image quality.
BICUBIC - Bicubic interpolation resizing, slower than **LINEAR**, but better image quality.

Name: **ConverterPlugIn.PNImageConverter.WICScalingMode**

This is the sampling or filtering mode to use when scaling an image. An image needs to be scaled when the resolution of the source image and destination image are not the same.

Values: **NORMAL** - Uses nearest neighbor scaling. This is nearest neighbor scaling, which is the fastest mode and often can produce the smallest image. The tradeoff is a lower image quality.
LINEAR - A bilinear interpolation algorithm where the weighted average of a 2x2 grid is used to compute the pixel values of the new image. Better quality than **NORMAL**.
BICUBIC - The new pixel values are computed using a weighted average of a 4x4 grid.
FANT - This scaling mode produces the best quality images but is slower and more CPU intensive than the others.

Name: **ConverterPlugIn.PNImageConverter.KeepSourceImageResolution**

Optionally keep the output image's resolution the same as source image. Note that fax mode and other image option actions ([Image Options](#)) will still override the end result. Overrides the *Devmode settings;Resolution* settings from [Devmode settings](#).

Values: **True** - Create the new image with the same resolution as the original image.
False - Creates the new image with the resolution specified in the *Devmode settings;Resolution* setting.

Conversion Settings - Toolkits and Scaling Modes

Name: **ConverterPlugIn.PNImageConverter.ResampleImageToMaxWidthOrHeightInPixels**

Dynamically sample the output image to a specific maximum width or height, whichever criteria is met first. The desired dimension is specified in *pixels*. Note that fax mode and other image option actions ([Image Options](#)) will still override the end result.

Values: The desired maximum width or height in pixels.

Name: **ConverterPlugIn.PNImageConverter.AlphaBackgroundColorRGB**

For images that support transparency, or alphablending, optionally set the desired background color when converting the image. The default background color is White.

Values: The desired background color set as RGB triplet separated by commas.

255,255,255 - White

0,0,0 - Black

Supported Image File Formats

The table below lists the image formats supported by each tool kit.

Cserve Portable Network Graphics images (*.png)	•	•
Graphics Interchange Format image files (*.gif)	•	•
Icon Format (*.ico)		•
JPEG images (*.jpg)	•	•
TIFF images (*.tif)	•	•
Windows Bitmap images (*.bmp)	•	•
Windows Media Photo (*.wdp, *.hdp, *.jxr)		•
ZSoft PCX images (*.pcx)	•	
ZSoft DCX images (*.dcx)	•	

OutsideIn AX Options

These options control the behavior of the OutsideIn AX converter used by Document Conversion Service. Table values in **bold** text are the default value for that setting.



Sample Profile

```
<?xml version="1.0" encoding="utf-8"?>
<Profile Type="0"
    DisplayName="TIFF 300dpi OutsideIn AX"
    Description="OutsideIn AX with borders and margins">
    <Settings>
        <add Name ="Oracle.OutsideInAX.BMPPrintBorder" Value="0"/>
        <add Name ="Oracle.OutsideInAX.IntlFlags" Value="1"/>
        <add Name ="Oracle.OutsideInAX.PrintMarginTop" Value="0.50"/>
        <add Name ="Oracle.OutsideInAX.PrintMarginBottom" Value="0.50"/>

        <!-- Output file options -->
        <add Name="Devmode settings;Resolution" Value="300"/>
        <add Name="Save;Output File Format" Value="TIFF Multipaged"/>
        ...
    </Settings>
</Profile>
```



Code Sample - C#

```
PNDocConvQueueServiceLib.PNDocConvQueueItem item = null;

// Create the conversion item
item = new PNDocConvQueueServiceLib.PNDocConvQueueItem();

// Set conversion settings
item.Set("Oracle.OutsideInAX.BMPPrintBorder", "0");
item.Set("Oracle.OutsideInAX.IntlFlags", "1");
item.Set("Oracle.OutsideInAX.PrintMarginTop", "0.50");
item.Set("Oracle.OutsideInAX.PrintMarginBottom", "0.50");

item.Set("Devmode settings;Resolution", "300");
item.Set("Save;Output File Format", "TIFF Multipaged");
...
// convert the file
item.Convert("Microsoft Outside-In AX",
    "C:\Test\Report.wpd",
    "C:\Test\Out\ConvertedReport");
```



Code Sample - VB.NET

```
Dim item As PNDocConvQueueServiceLib.IPNDocConvQueueItem

' Create the conversion item
item = New PNDocConvQueueServiceLib.PNDocConvQueueItem()

' Set conversion settings
item.Set("Oracle.OutsideInAX.BMPPrintBorder", "0")
item.Set("Oracle.OutsideInAX.IntlFlags", "1")
item.Set("Oracle.OutsideInAX.PrintMarginTop", "0.50")
item.Set("Oracle.OutsideInAX.PrintMarginBottom", "0.50")

item.Set("Devmode settings;Resolution", "300")
item.Set("Save;Output File Format", "TIFF Multipaged")
...
' convert the file
item.Convert("Microsoft Word", _
            "C:\Test\Report.wpd", _
            "C:\Test\Out\ConvertedReport")
```

Conversion Settings - OutsideIn AX Printing

Name: **Oracle.OutsideInAX.BMPPrintBorder**

Print a one pixel wide border around the image.

Values: 0 - do not print the border
1 - print the border

Name: **Oracle.OutsideInAX.VECPrintBorder**

Print a one pixel wide border around the image.

Values: 0 - do not print the border
1 - print the border

Name: **Oracle.OutsideInAX.IntlFlags**

Specifies what unit of measurement is used for the print margins below. Units are either inches or metric units.

Values: 0 - Metric
1 - Imperial (Inches)

Name: **Oracle.OutsideInAX.PrintMarginTop**

The top print margin height.

Values: A string value representing the printer margin as a floating point number, such as 0.50 for half an inch.

Conversion Settings - OutsideIn AX Printing**Name:** **Oracle.OutsideInAX.PrintMarginBottom**

The bottom print margin height.

Values: A string value representing the printer margin as a floating point number, such as 0.50 for half an inch.**Name:** **Oracle.OutsideInAX.PrintMarginLeft**

The left print margin width.

Values: A string value representing the printer margin as a floating point number, such as 0.50 for half an inch.**Name:** **Oracle.OutsideInAX.PrintMarginRight**

The right print margin width.

Values: A string value representing the printer margin as a floating point number, such as 0.50 for half an inch.

Save

These options control the orientation, resolution, color mode and paper size of the output file. You can also choose to split multipage files based on the number of pages per file or a file size threshold. Table values in **bold** text are the default value for that setting.



Sample Profile

```
<?xml version="1.0" encoding="utf-8"?>
<Profile Type="0"
  DisplayName="TIFF 300dpi Serialized with Overwrite"
  Description="Create TIFF serialized, overwrite existing files.">
  <Settings>

    <!-- Output file options -->
    <add Name="Devmode settings;Resolution" Value="300"/>
    <add Name="Save;Output File Format" Value="TIFF Serialized"/>
    <add Name="Save;Prompt" Value="0"/>
    <add Name="Save;Overwrite" Value="1"/>
    <add Name="Save;Color reduction" Value="BW"/>
    ...

  </Settings>
</Profile>
```



Code Sample - C#

```
PNDocConvQueueServiceLib.PNDocConvQueueItem item = null;

// Create the conversion item
item = new PNDocConvQueueServiceLib.PNDocConvQueueItem();

// Set conversion settings
item.Set("Devmode settings;Resolution", "300");
item.Set("Save;Output File Format", "TIFF Serialized");
item.Set("Save;Color reduction", "BW");
item.Set("Save;
...
// convert the file
item.Convert("Microsoft Word",
  @"C:\Test\Report.docx",
  @"C:\Test\Out\ConvertedReport");
```

**Code Sample - VB.NET**

```
Dim item As PNDocConvQueueServiceLib.IPNDocConvQueueItem

' Create the conversion item
item = New PNDocConvQueueServiceLib.PNDocConvQueueItem()

' Set conversion settings
item.Set("Devmode settings;Resolution", "300")
item.Set("Save;Output File Format", "TIFF Serialized")
item.Set("Save;Prompt", "0")
item.Set("Save;Overwrite", "1")
item.Set("Save;Color reduction", "BW")
...
' convert the file
item.Convert("Microsoft Word", _
            "C:\Test\Report.docx", _
            "C:\Test\Out\ConvertedReport")
```

Conversion Settings - Save**Name:** **Save;Use JobID**

Use the driver JobID when creating the file name. The driver stores an internal number that is automatically incremented for each print job.

Values: **0** - Do not include JobID in file name.
1 - Include JobID in file name.

Name: **Save;Append**

Append the new images to an existing file name or sequence.

Values: **0** - Do not append, output is a new file.
1 - Output is appended to existing file or sequence.

Name: **Save;Output directory**

Values: The output directory path in which to save the image.

Name: **Save;Output filename**

Values: Base file name excluding path and extension to use to name the file. Default is the document name submitted to print job.

Conversion Settings - Save

Name: **Save;Output File Format**

The type of file to create.

Values: **JPEG** - JPEG (*.jpg)
 TIFF Multipaged - TIFF Multipaged (*.tif)
 TIFF Serialized - TIFF Serialized (*.tif)
 Adobe PDF Multipaged - Adobe PDF Multipaged (*.pdf)
 Adobe PDF Serialized - Adobe PDF Serialized (*.pdf)
 CompuServe GIF - CompuServe GIF (*.gif)
 CompuServe PNG - CompuServe PNG (*.png)
 Windows BMP - Windows BMP (*.bmp)
 TARGA - Targa (*.tga)
 Adobe Photoshop 3.0 - Adobe Photoshop 3.0 (*.psd)
 ZSoft PCX - ZSoft PCX (*.pcx)
 ZSoft DCX - ZSoft DCX (*.dcx)

Name: **Save;remove file extension**

Removes the filename extension from the original filename before creating the new filename. If set to 0, a file *Document.doc* created as TIFF would become *Document.doc.tif*; when set to remove the extension, the resulting filename would be *Document.tif*.

Values: **0** - Leave original filename extension in new filename
 1 - Remove original filename extension before creating new filename.

Name: **Save;Color reduction**

Use the color reduction options below to reduce the number of colors in the output files.

Values: none - No color reduction
Optimal - Reduce to lowest color count needed per page
 BW - Reduce to black and white using selected dithering method
 grey - Reduce to greyscale
 256Colors - Create all pages as 8-bit color (256 colors)
 16Colors - Create all pages as 4-bit color (16 colors)
 optimalMax256Colors - Reduces to lowest color count needed for each page, any pages over 256 colors are reduced to 256 colors.
 optimalMax16Colors - Reduces to lowest color count needed for each page, any pages over 16 colors are reduced to 16 colors.

Conversion Settings - Save

Name: **Save;Dithering method**

Dithering enhances the appearance of color images that have been reduced to black and white.

Values: None - No dithering
 Floyd - Floyd-Steinberg dithering
 Burkes - Burkes dithering
 Bayer - Bayer dithering
Halftone - Halftone dithering

Name: **Save;SplitFileEveryNPagesEnabled**

Enables file splitting based on the page count set by **SplitFileEveryNPages**. When file splitting is enabled, the [serialized naming profile](#) is always used to name each file in the sequence. Can be combined with *SplitFileWhenFileSizeExceedsThresholdEnabled* to split by page count and file size.

File splitting only applies to the following multipaged file formats:

- TIFF Multipaged - TIFF Multipaged (*.tif)
- Adobe PDF Multipaged - Adobe PDF Multipaged (*.pdf)
- ZSoft DCX - ZSoft DCX (*.dcx)

Values: **0** - Do not split the file, create a single multipaged file.
1 - Split the file when the page count reaches limit set by SplitFileEveryNPages.

Name: **Save;SplitFileEveryNPages**

The page count at which to start creating a new file.

Values: 0-4294967295, default is **1000**.

Conversion Settings - Save

Name: **Save;SplitFileWhenFileSizeExceedsThresholdEnabled**

Enables file splitting based on a file size threshold set by **SplitFileSizeThresholdInBytes**. The file is split when the file size gets larger than the threshold. When file splitting is enabled, the [serialized naming profile](#) is always used to name each file in the sequence. Can be combined with *SplitFileEveryNPagesEnabled* to split by file size and page count.

File splitting only applies to the following multipaged file formats:

- TIFF Multipaged - TIFF Multipaged (*.tif)
- Adobe PDF Multipaged - Adobe PDF Multipaged (*.pdf)
- ZSoft DCX - ZSoft DCX (*.dcx)

Values: **0** - Do not split the file, create a single multipaged file.
1 - Split the file when the file size exceeds the limit set by *SplitFileSizeThresholdInBytes*.

Name: **Save;SplitFileSizeThresholdInBytes**

The file size, in bytes, at which to start creating a new file.

Values: 0-18446744073709551615, default is **1073741824**, or 1GB.

Devmode settings

These options control the orientation, resolution, color mode and paper size of the output file. Table values in **bold** text are the default value for that setting.

Sample Profile

```
<?xml version="1.0" encoding="utf-8"?>
<Profile Type="0"
    DisplayName="TIFF 300dpi Color "
    Description="Create color TIFF">
    <Settings>

        <!-- Output file options -->
        <add Name="Devmode settings;Resolution" Value="300"/>
        <add Name="Devmode settings;Color" Value="1"/>
        <add Name="Save;Output File Format" Value="TIFF Multipaged"/>
        ...
    </Settings>
</Profile>
```

Code Sample - C#

```
PNDocConvQueueServiceLib.PNDocConvQueueItem item = null;

// Create the conversion item
item = new PNDocConvQueueServiceLib.PNDocConvQueueItem();

// Set conversion settings
item.Set("Devmode settings;Resolution", "300");
item.Set("Devmode settings;Color", "1");
item.Set("Save;Output File Format", "TIFF Multipaged");
...
// convert the file
item.Convert("Microsoft Word",
    @"C:\Test\Report.docx",
    @"C:\Test\Out\ConvertedReport");
```

Code Sample - VB.NET

```
Dim item As PNDocConvQueueServiceLib.IPNDocConvQueueItem

' Create the conversion item
item = New PNDocConvQueueServiceLib.PNDocConvQueueItem()

' Set conversion settings
item.Set("Devmode settings;Resolution", "300")
item.Set("Devmode settings;Color", "1")
item.Set("Save;Output File Format", "TIFF Multipaged")
...
' convert the file
item.Convert("Microsoft Word", _
    "C:\Test\Report.docx", _
    "C:\Test\Out\ConvertedReport")
```

Conversion Settings - Devmode	
Name:	Devmode settings;Orientation
	Orientation of the page when the file is converted.
Values:	Portrait Landscape
Name:	Devmode settings;Resolution
	Number of dots per inch.
Values:	1200, 720, 600, 400, 360, 300 , 254, 240, 200, 150, 120, 100, 75, 60, 50
Name:	Devmode settings;Color
	Print files in color or black and white
Values:	1 Color mode 0 Black and white, or monochrome mode.
Name:	Devmode settings;Paper Size
	Standard paper sizes available. Other custom paper sizes you may have added are also available by name.
Values:	Letter Letter Small Tabloid Legal Statement Executive A3 A4 A4 Small A5 B4 B5 Folio Quarto 10x14 11x17 Note Envelope #9 Envelope #10 Envelope #11 Envelope #12 Envelope #14

Conversion Settings - Devmode**Name:** **Devmode settings;Paper Size**

Standard paper sizes available. Other custom paper sizes you may have added are also available by name.

C Size Sheet
 D Size Sheet
 E Size Sheet
 F Size Sheet
 Envelope DL
 Envelope C5
 Envelope C3
 Envelope C4
 Envelope C6
 Envelope C65
 Envelope B4
 Envelope B5
 Envelope B6
 Envelope Italy
 Envelope Monarch
 Envelope Personal
 US Std Fanfold
 German Std Fanfold
 German Legal Fanfold
 ISO B4
 Japanese Postcard
 9x11
 10x11
 15x11
 Envelope Invite
 Letter Extra
 Legal Extra
 Tabloid Extra
 A4 Extra
 Letter Transverse
 A4 Transverse
 Letter Extra Transverse
 A Plus
 B Plus
 Letter Plus
 A4 Plus
 A5 Transverse
 B5 Transverse
 A3 Extra
 A5 Extra
 B5 Extra
 A3 Transverse
 A3 Extra Transverse
 A1 594 x 841 mm
 A0 841 x 1189 mm
 B3 (ISO) 353 x 500 mm
 B2 (ISO) 500 x 707 mm

Conversion Settings - Devmode

Name: Devmode settings;Paper Size

Standard paper sizes available. Other custom paper sizes you may have added are also available by name.

B1 (ISO) 707 x 1000 mm
B3 (JIS) 364 x 515 mm
B2 (JIS) 515 x 728 mm
B1 (JIS) 728 x 1030 mm
B0 (JIS) 1030 x 1456 mm


Advanced File Naming

There are four different naming profiles that control how the output file is named. Which naming profile is used depends on if you are creating serialized or multipaged output, and if you have the `Save;UseJobID` setting set to true. It is the combination of these settings that determines which profile is used to build the output filename.

The only exception to this is when file splitting by page count (`Save;SplitFileEveryNPagesEnabled`) or file size (`Save;SplitFileWhenFileSizeExceedsThresholdEnabled`) is enabled. When file splitting is enabled, the serialized naming profile is always used to name each file in the sequence. The file splitting options are only used when creating multipaged file types.

Serialized or Multi-page	Inclu de Jobl D	Naming Profile
Serialized	No	Serialized
	Yes	Serialized w/ JobID
Multi-paged	No	Multi-page
	Yes	Multi-page w/ JobID

In most scenarios you will never need to change these values. Care must be taken when you do. The table below lists the settings to use to customize the output file naming. Table values in **bold** text are the default value for that setting.



Sample Profile

```

<?xml version="1.0" encoding="utf-8"?>
<Profile Type="0"
  DisplayName="Color TIFF 300dpi Named by Page Number"
  Description ="Create color TIFF named by page number">
  <Settings>

    <!-- Output file options -->
    <add Name="Devmode settings;Resolution" Value="300"/>
    <add Name="Devmode settings;Color" Value="1"/>
    <add Name="Save;Output File Format" Value="TIFF Serialized"/>
    <add Name="Advanced File Naming;Format string S" Value="%s"/>
    <add Name="Advanced File Naming;Variables S"
      Value="$(PrintedPageNumber)"/>

    ...

  </Settings>
</Profile>

```



Code Sample - C#

```
PNDocConvQueueServiceLib.PNDocConvQueueItem item = null;

// Create the conversion item
item = new PNDocConvQueueServiceLib.PNDocConvQueueItem();

// Set conversion settings
item.Set("Devmode settings;Resolution", "300");
item.Set("Devmode settings;Color", "1");
item.Set("Save;Output File Format", "TIFF Serialized");

item.Set("Advanced File Naming;Format string S", "%s");
item.Set("Advanced File Naming;Variables S",
        "$(PrintedPageNumber)");

...
// convert the file
item.Convert("Microsoft Word",
            @"C:\Test\Report.docx",
            @"C:\Test\Out\ConvertedReport");
```



Code Sample - VB.NET

```
Dim item As PNDocConvQueueServiceLib.IPNDocConvQueueItem

' Create the conversion item
item = New PNDocConvQueueServiceLib.PNDocConvQueueItem()

' Set conversion settings
item.Set("Devmode settings;Resolution", "300")
item.Set("Devmode settings;Color", "1")
item.Set("Save;Output File Format", "TIFF Serialized")

item.Set("Advanced File Naming;Format string S", "%s")
item.Set("Advanced File Naming;Variables S",
        "$(PrintedPageNumber)")

...
' convert the file
item.Convert("Microsoft Word", _
            "C:\Test\Report.docx", _
            "C:\Test\Out\ConvertedReport")
```

Conversion Settings - Advanced File Naming

Name:	Advanced File Naming;Format string S
	Format string for the serialized naming profile. Also used to name the sequence of files when file splitting is enabled.
Values:	A string containing the format string used to create the output file name. The format string can contain placeholders %s and %d that correspond to the variables passed in <i>Advanced File Naming;Variables S</i> below.

Conversion Settings - Advanced File Naming	
Name:	Advanced File Naming;Use default extension S
	Use the default file extension for the output type when naming the output file.
Values:	0 - Do not use default file extension 1 - Use default file extension
Name:	Advanced File Naming;Variables S
	Comma-delimited list of variables that correspond to the placeholders in the format string supplied in <i>Advanced File Naming;Format string S</i> above.
Values:	See list of variables below.
Name:	Advanced File Naming;Format string SJ
	Format string for serialized with JobID naming profile. In this profile a JobID, a number that is automatically incremented, is used as part of the filename.
Values:	A string containing the format string used to create the output file name. The format string can contain placeholders %s and %d that correspond to the variables passed in <i>Advanced File Naming;Variables SJ</i> below.
Name:	Advanced File Naming;Use default extension SJ
	Use the default file extension for the output type when naming the output file.
Values:	0 - Do not use default file extension 1 - Use default file extension
Name:	Advanced File Naming;Variables SJ
	Comma-delimited list of variables that correspond to the placeholders in the format string supplied in <i>Advanced File Naming;Format string SJ</i> above.
Values:	See list of variables below.
Name:	Advanced File Naming;Format string M
	Format string for the multipaged naming profile.
Values:	A string containing the format string used to create the output file name. The format string can contain placeholders %s and %d that correspond to the variables passed in <i>Advanced File Naming;Variables M</i> below.

Conversion Settings - Advanced File Naming	
Name:	Advanced File Naming;Use default extension M
	Use the default file extension for the output type when naming the output file.
Values:	0 - Do not use default file extension 1 - Use default file extension
Name:	Advanced File Naming;Variables M
	Comma-delimited list of variables that correspond to the placeholders in the format string supplied in <i>Advanced File Naming;Format string M</i> above.
Values:	See list of variables below.
Name:	Advanced File Naming;Format string MJ
	Format string for the multipaged with JobID naming profile. In this profile a JobID, a number that is automatically incremented, is used as part of the filename.
Values:	A string containing the format string used to create the output file name. The format string can contain placeholders %s and %d that correspond to the variables passed in <i>Advanced File Naming;Variables MJ</i> below.
Name:	Advanced File Naming;Use default extension MJ
	Use the default file extension for the output type when naming the output file.
Values:	0 - Do not use default file extension 1 - Use default file extension
Name:	Advanced File Naming;Variables MJ
	Comma-delimited list of variables that correspond to the placeholders in the format string supplied in <i>Advanced File Naming;Format string MJ</i> above.
Values:	See list of variables below.

Variables for Custom Naming

Variable	Type and Format String Place Holder	Description
\$(Day)	Numeric, %d	The day in numeric format that the print job was submitted to the printer, from 1-31.
\$(DocumentPageNumber)	Numeric, %d	The page number of the document being printed.
\$(FileExtension)	String, %s	The file extension for the type of file being created.
\$(FileNumber)	Numeric, %d	The file number of the sequence of files. For multipaged output, this is always 1. For serialized output this is the number of the file in the sequence.
\$(Hour)	Numeric, %d	The hour in numeric format that the print job was submitted to the printer, 1-12 or 0-23 depending on your system preferences.
\$(JobID)	Numeric, %d	The unique JobID used by the printer. This is set to zero when the driver is first installed and is automatically incremented by the driver at the start of every print job. The JobID is often used to ensure that all files created have unique names.
\$(JobStatus)	Numeric, %d	The status of the print job, 1 for success, 0 for failure.
\$(MachineName)	String, %s	The name of the computer the print job is running on.
\$(Minute)	Numeric, %d	The minute in numeric format that the print job was submitted to the printer, from 0-59.
\$(Month)	Numeric, %d	The month in numeric format that the print job was submitted to the printer, from 1-12.
\$(OutputFileName)	String, %s	The contents of the \$(OutputFileName) field. If this field is empty the name the printing application used when submitting the print job is used.
\$(PrintedPageNumber)	String, %s	The page number of the page being printed; this is not always the same as \$(DocumentPageNumber).

Variable	Type and Format String Place Holder	Description
\$(Second)	Numeric, %d	The second in numeric format that the print job was submitted to the printer, from 0-59.
\$(UserName)	String, %s	The name of the user who submitted the print job.
\$(Year)	Numeric, %d	The year in numeric format that the print job was submitted to the printer.

Default Naming Profile Strings

Profile	Format String	Variables and Resulting File Names for TIFF Creation
Serialized	%s_%3d	\$(OutputFileName) \$(FileNameNumber) C:\Test\Invoice_001.tif C:\Test\Invoice_002.tif C:\Test\Invoice_003.tif ...
Serialized w/ JobID	%3d_%s_%3d	\$(JobID) \$(OutputFileName) \$(FileNameNumber) C:\Test\010_Invoice_001.tif C:\Test\010_Invoice_002.tif C:\Test\010_Invoice_003.tif ...
Multi-page	%s	\$(OutputFileName) C:\Test\Invoice.tif
Multi-page w/ JobID	%3d_%s	\$(JobID) \$(OutputFileName) C:\Test\011_Invoice.tif

Image Options

These options control the fax mode and creation of the output file. Table values in **bold** text are the default value for that setting.



Sample Profile

```
<?xml version="1.0" encoding="utf-8"?>
<Profile Type="0"
  DisplayName="Create Fax TIFF Serialized"
  Description="Create profile F fax 204 x 196">
  <Settings>

    <!-- Output file options -->
    <add Name="Devmode settings;Color" Value="1"/>
    <add Name="Image Options;Fax" Value="1"/>
    <add Name="Image Options;Fax Profile" Value="0"/>
    <add Name="Image Options;Fax Resolution" Value="3"/>
    <add Name="Save;Output File Format" Value="TIFF Serialized"/>
    ...

  </Settings>
</Profile>
```



Code Sample - C#

```
PNDocConvQueueServiceLib.PNDocConvQueueItem item = null;

// Create the conversion item
item = new PNDocConvQueueServiceLib.PNDocConvQueueItem();

// Set conversion settings
item.Set("Devmode settings;Color", "1");
item.Set("Image Options;Fax", "1");
item.Set("Image Options;Fax Profile", "0");
item.Set("Image Options;Fax Resolution", "3");
item.Set("Save;Output File Format", "TIFF Serialized");

...
// convert the file
item.Convert("Microsoft Word",
  @"C:\Test\Report.docx",
  @"C:\Test\Out\ConvertedReport");
```



Code Sample - VB.NET

```
Dim item As PNDocConvQueueServiceLib.IPNDocConvQueueItem

' Create the conversion item
item = New PNDocConvQueueServiceLib.PNDocConvQueueItem()

' Set conversion settings
item.Set("Devmode settings;Color", "1")
item.Set("Image Options;Fax", "1")
item.Set("Image Options;Fax Profile", "0")
item.Set("Image Options;Fax Resolution", "3")
item.Set("Save;Output File Format", "TIFF Serialized")
...
' convert the file
item.Convert("Microsoft Word", _
            "C:\Test\Report.docx", _
            "C:\Test\Out\ConvertedReport")
```

Conversion Settings - Image Options

Name: Image Options;Fax

Values: 0 - Do not create fax format file.
1 - Create an image where its width is limited to fax resolution as determined by Fax Profile and Fax Resolution settings

Name: Image Options;Fax Profile

Values: 0 - Profile F, standard monochrome
1 - Profile S, simplified monochrome
2 - Profile C, color fax

Name: Image Options;Fax Resolution

Values: 0 - 200 x 100 resolution (Profile S, F)
1 - 200 x 200 resolution (Profile S, F, C)
2 - 204 x 98 resolution (Profile S, F)
3 - 204 x 196 resolution (Profile S, F)
4 - 300 x 300 resolution (Profile F, C)
5 - 400 x 400 resolution (Profile F, C)
6 - 408 x 391 resolution (Profile F)
7 - 204 x 391 resolution (Profile F)
8 - 300 x 600 resolution (Profile F)
9 - 400 x 800 resolution (Profile F)
10 - 600 x 600 resolution (Profile F, C)
11 - 600 x 1200 resolution (Profile F)
12 - 1200 x 1200 resolution (Profile F, C)
13 - 100 x 100 resolution (Profile F, C)

Conversion Settings - Image Options	
Name:	Image Options;Fax Use Printer Resolution
Values:	0 - Do not use printer resolution 1 - Use printer resolution
Name:	Image Options;Fax Paper Width
Values:	0 - Letter 1 - Legal 2 - A4 (ISO) 3 - B4 (ISO) 4 - A3 (ISO) 5 - Auto
Name:	Image Options;Fax Paper Height
Values:	0 - Variable height 1 - Fixed height
Name:	Image Options;Fax Page Scaling
Values:	0 - Fit to Page 1 - Actual Size
Name:	Image Options;Fax Page Scaling Auto Rotate
Values:	0 - Do not auto-rotate the page 1 - Auto-rotate the page if needed
Name:	Image Options;Fax Page Scaling Lock Aspect Ratio
Values:	0 - Do not maintain fax page aspect ratio when scaling 1 - Maintain fax page aspect ratio when scaling
Name:	Image Options;Fax Page Scaling Shrink Larger
Values:	0 - Do not shrink fax to fit on page 1 - Shrink fax to fit on page

Conversion Settings - Image Options	
Name:	Image Options;Fax Page Scaling H Align
Values:	Left - Align image left Middle - Align image in the center Right - Align image right
Name:	Image Options;Fax Page Scaling V Align
Values:	Top - Align image top Middle - Align image in the center Bottom - Align image bottom
Name:	Image Options;Fax Page Use 256 Greyscale Palette
Values:	0 - Use the smaller 64 grayscale palette 1 - Use 256 grayscale palette
Name:	Image Options;Fill order
Values:	LSB2MSB - Least significant bit to most significant bit MSB2LSB - Most significant bit to least significant bit
Name:	Image Options;EOLs Byte Aligned
Values:	0 - EOLs not byte aligned (no fillbits) 1 - EOLs byte aligned (use fillbits)
Name:	Image Options;Photometric
Values:	MinIsWhite MinIsBlack
Name:	Image Options;Include DateTime
Values:	0 - DateTime field not included in file 1 - DateTime field included in file
Name:	Image Options;Motorola Format
Values:	0 - Use Intel byte order 1 - Use Motorola byte order

Conversion Settings - Image Options**Name:** Image Options;Rotate portrait

Specified in degrees of rotation (counter-clockwise).

Values: 0
90
180
270**Name:** Image Options;Rotate landscape

Specified in degrees of rotation (counter-clockwise).

Values: 0
90
180
270**Name:** Image Options;Include Software Name and Release**Values:** 0 - Software field not included in file
1 - Software field field included in file

TIFF File Format

Table values in **bold** text are the default value for that setting.



Sample Profile

```
<?xml version="1.0" encoding="utf-8"?>
<Profile Type="0"
  DisplayName="Create Compressed TIFF Serialized"
  Description="Create Compressed TIFF Serialized">
  <Settings>

    <!-- Output file options -->
    <add Name="Devmode settings;Color" Value="1"/>
    <add Name="Save;Output File Format" Value="TIFF Serialized"/>
    <add Name="Save;Color reduction" Value="Optimal"/>
    <add Name="TIFF File Format;BW compression" Value="Group3-2D"/>
    <add Name="TIFF File Format;Color compression" Value="LZW"/>
    ...

  </Settings>
</Profile>
```



Code Sample - C#

```
PNDocConvQueueServiceLib.PNDocConvQueueItem item = null;

// Create the conversion item
item = new PNDocConvQueueServiceLib.PNDocConvQueueItem();

// Set conversion settings
item.Set("Devmode settings;Color", "1");
item.Set("Save;Output File Format", "TIFF Serialized");
item.Set("Save;Color reduction", "Optimal");
item.Set("TIFF File Format;BW compression", "Group3-2D");
item.Set("TIFF File Format;Color compression", "LZW");
...
// convert the file
item.Convert("Microsoft Word",
  @"C:\Test\Report.docx",
  @"C:\Test\Out\ConvertedReport");
```

**Code Sample - VB.NET**

```
Dim item As PNDocConvQueueServiceLib.IPNDocConvQueueItem

' Create the conversion item
item = New PNDocConvQueueServiceLib.PNDocConvQueueItem()

' Set conversion settings
item.Set("Devmode settings;Color", "1")
item.Set("Save;Output File Format", "TIFF Serialized")
item.Set("Save;Color reduction", "Optimal")
item.Set("TIFF File Format;BW compression", "Group3-2D")
item.Set("TIFF File Format;Color compression", "LZW")
...
' convert the file
item.Convert("Microsoft Word", _
            "C:\Test\Report.docx", _
            "C:\Test\Out\ConvertedReport")
```

Conversion Settings - TIFF File Format

Name: TIFF File Format;BW compression

Values: None - No black and white compression
Group4 - CCITT Group4 Fax compression
 Group3-2D - CCITT Group3 2D Fax compression
 Group3-1D - CCITT Group3 1D Fax compression
 MH - CCITT Modified Huffman compression
 LZW - LZW compression
 Packbits - Packbits (RLE) compression

Name: TIFF File Format;Color compression

Values: Uncompressed RGB - No color compression
 Uncompressed CMYK - No color compression, CMYK color
 Packbits RGB -Packbits (RLE) compression
 Packbits CMYK -Packbits (RLE) compression, CMYK color
 High quality JPEG - High quality JPEG compression
 Medium quality JPEG - Medium quality JPEG compression
 Low quality JPEG - Low quality JPEG compression
LZW RGB - LZW compression
 LZW CMYK - LZW compression, CMYK color

Name: TIFF File Format;Indexed compression

Values: Uncompressed - No color compression
 Packbits - Packbits (RLE) compression
 High quality JPEG - High quality JPEG compression
 Medium quality JPEG - Medium quality JPEG compression
 Low quality JPEG - Low quality JPEG compression
LZW - LZW compression

Conversion Settings - TIFF File Format

Name: TIFF File Format;Greyscale compression

Values: Uncompressed - No color compression
Packbits - Packbits (RLE) compression
High quality JPEG - High quality JPEG compression
Medium quality JPEG - Medium quality JPEG compression
Low quality JPEG - Low quality JPEG compression
LZW - LZW compression

PDF File Format

These options control the compression methods used during the creation of PDF output files. Table values in **bold** text are the default value for that setting.

Sample Profile

```
<?xml version="1.0" encoding="utf-8"?>
<Profile Type="0"
    DisplayName="Create PDF/A "
    Description="Create PDF/A-1b">
  <Settings>

    <!-- Output file options -->
    <add Name="Devmode settings;Color" Value="1"/>
    <add Name="Save;Output File Format" Value="PDF Multipaged"/>
    <add Name="Save;Prompt" Value="0"/>
    <add Name="Save;Overwrite" Value="1"/>
    <add Name="PDF File Format;PDF Standard" Value="PDF/A-1b"/>
    <add Name="PDF File Format;Use compression" Value="1"/>
    ...

  </Settings>
</Profile>
```

Code Sample - C#

```
PNDocConvQueueServiceLib.PNDocConvQueueItem item = null;

// Create the conversion item
item = new PNDocConvQueueServiceLib.PNDocConvQueueItem();

// Set conversion settings
item.Set("Devmode settings;Color", "1");
item.Set("Save;Output File Format", "PDF Multipaged");
item.Set("Save;Prompt", "0");
item.Set("Save;Overwrite", "1");
item.Set("PDF File Format;PDF Standard", "PDF/A-1b");
item.Set("PDF File Format;Use compression", "1");

...
// convert the file
item.Convert("Microsoft Word",
    @"C:\Test\Report.docx",
    @"C:\Test\Out\ConvertedReport");
```



Code Sample - VB.NET

```
Dim item As PNDocConvQueueServiceLib.IPNDocConvQueueItem

' Create the conversion item
item = New PNDocConvQueueServiceLib.PNDocConvQueueItem()

' Set conversion settings
item.Set("Devmode settings;Color", "1")
item.Set("Save;Output File Format", "PDF Multipaged")
item.Set("Save;Prompt", "0")
item.Set("Save;Overwrite", "1")
item.Set("PDF File Format;PDF Standard", "PDF/A-1b")
item.Set("PDF File Format;Use compression", "1")
...
' convert the file
item.Convert("Microsoft Word", _
            "C:\Test\Report.docx", _
            "C:\Test\Out\ConvertedReport")
```

Conversion Settings - PDF File Format

Name:	PDF File Format;Embed Pages as Images
Values:	0 - Creates vector pages, where possible, in the PDF file; does not OCR 1 - Embeds each page of the PDF as an image, creating a raster PDF
Name:	PDF File Format;Include Outline This setting applies only when creating vector PDF files, and only if the source file contains outline information. Outline information is shown as bookmarks in a PDF document.
Values:	0 - Does not include outline information in vector PDF files 1 - Includes outline (heading) information, where possible, in vector PDF files
Name:	PDF File Format;Use compression
Values:	0 - Do not compress the file 1 - Enable compression for the file
Name:	PDF File Format;Use ASCII
Values:	0 - No ASCII format compression 1 - Enable ASCII format compression
Name:	PDF File Format;PDF Standard
Values:	None - Create PDF files that are not PDF/A-1b compliant PDF/A-1b - Create PDF/A-1b compliant PDF files when creating raster PDF

Conversion Settings - PDF File Format	
Name:	PDF File Format;Content encoding
Values:	None - No compression ZIP - ZIP compression RLE - Packbits (run length) compression LZW - LZW compression
Name:	PDF File Format;Color compression
Values:	None - No color compression ZIP - ZIP compression RLE - Packbits (run length) compression JPEG High - High quality JPEG compression JPEG Medium - Medium quality JPEG compression JPEG Low - Low quality JPEG compression LZW - LZW compression
Name:	PDF File Format;Greyscale compression
Values:	None - No color compression ZIP - ZIP compression RLE - Packbits (run length) compression JPEG High - High quality JPEG compression JPEG Medium - Medium quality JPEG compression JPEG Low - Low quality JPEG compression LZW - LZW compression
Name:	PDF File Format;Indexed compression
Values:	None - No color compression ZIP - ZIP compression RLE - Packbits (run length) compression JPEG High - High quality JPEG compression JPEG Medium - Medium quality JPEG compression JPEG Low - Low quality JPEG compression LZW - LZW compression
Name:	PDF File Format;BW compression
Values:	None - No black and white compression Group4 - CCITT Group4 Fax compression Group3-2D - CCITT Group3 2D Fax compression Group3-1D -CCITT Group3 1D Fax compression

PDF Security

These options control the security options available in creation of PDF output files. Table values in **bold** text are the default value for that setting.

Sample Profile

```
<?xml version="1.0" encoding="utf-8"?>
<Profile Type="0"
    DisplayName="Create secure PDF "
    Description="Create secure PDF, no password">
    <Settings>

        <!-- Output file options -->
        <add Name ="Devmode settings;Color" Value="1"/>
        <add Name ="Save;Output File Format" Value="PDF Multipaged"/>
        <add Name ="Save;Prompt" Value="0"/>
        <add Name ="Save;Overwrite" Value="1"/>
        <add Name ="PDF File Format;PDF Standard" Value="None"/>
        <add Name ="PDF File Format;Use compression" Value="1"/>
        <add Name ="PDF Security;Use Security" Value="1"/>
        <add Name ="PDF Security;Encrypt Level" Value="1"/>
        ...

    </Settings>
</Profile>
```

Code Sample - C#

```
PNDocConvQueueServiceLib.PNDocConvQueueItem item = null;

// Create the conversion item
item = new PNDocConvQueueServiceLib.PNDocConvQueueItem();

// Set conversion settings
item.Set("Devmode settings;Color", "1");
item.Set("Save;Output File Format", "PDF Multipaged");
item.Set("Save;Prompt", "0");
item.Set("Save;Overwrite", "1");
item.Set("PDF File Format;PDF Standard", "None");
item.Set("PDF File Format;Use compression", "1");
item.Set("PDF Security;Use Security", "1");
item.Set("PDF Security;Encrypt Level", "1");

...
// convert the file
item.Convert("Microsoft Word",
    @"C:\Test\Report.docx",
    @"C:\Test\Out\ConvertedReport");
```

**Code Sample - VB.NET**

```

Dim item As PNDocConvQueueServiceLib.IPNDocConvQueueItem

' Create the conversion item
item = New PNDocConvQueueServiceLib.PNDocConvQueueItem()

' Set conversion settings
item.Set("Devmode settings;Color", "1")
item.Set("Save;Output File Format", "PDF Multipaged")
item.Set("Save;Prompt", "0")
item.Set("Save;Overwrite", "1")
item.Set("PDF File Format;PDF Standard", "None")
item.Set("PDF File Format;Use compression", "1")
item.Set("PDF Security;Use Security", "1")
item.Set("PDF Security;Encrypt Level", "1")
...
' convert the file
item.Convert("Microsoft Word", _
            "C:\Test\Report.docx", _
            "C:\Test\Out\ConvertedReport")

```

Conversion Settings - PDF Security

Name: PDF Security;Use Security

Values: 0 - No PDF security
1 - Enable PDF security

Name: PDF Security;Encrypt Level

Values: Values:
0 - Sets 40-bit encryption level
1 - Sets 128-bit encryption level

Name: PDF Security;Can Copy

Values: 0 - Do not allow users to copy text and graphics
1 - Allow users to copy text and graphics

Name: PDF Security;Can Print

Values: 0 - Do not allow users to print the document
1 - Allow users to print the document

Name: PDF Security;Can Change Doc

Values: 0 - Do not allow users to change the document
1 - Allow users to change the document

Conversion Settings - PDF Security	
Name:	PDF Security;Can ChangeOther
Values:	0 - Do not allow users to add or change comments and form fields 1 - Allow users to add or change comments and form fields
Name:	PDF Security;User Pswd On
Values:	0 - No user password required to open document 1 - User password required to open document
Name:	PDF Security;User Pswd
Values:	The user password.
Name:	PDF Security;Owner Pswd On
Values:	0 - No owner password required to change document 1 - Owner password required to change document
Name:	PDF Security;Owner Pswd
Values:	Owner password

JPEG File Format

These options control the compression levels of JPEG files. Table values in **bold** text are the default value for that setting.



Sample Profile

```
<?xml version="1.0" encoding="utf-8"?>
<Profile Type="0"
    DisplayName="JPEG 300dpi"
    Description="Create JPEG, compress color as Medium, grey as High.">
  <Settings>

    <!-- Output file options -->
    <add Name="Devmode settings;Resolution" Value="300"/>
    <add Name="Save;Output File Format" Value="JPEG"/>
    <add Name="Save;Prompt" Value="0"/>
    <add Name="Save;Overwrite" Value="1"/>
    <add Name="Save;Color reduction" Value="Optimal"/>
    <add Name="JPEG File Format;Color compression" Value="Medium Quality"/>
    <add Name="JPEG File Format;Greyscale compression" Value="High Quality"/>
    ...

  </Settings>
</Profile>
```



Code Sample - C#

```
PNDocConvQueueServiceLib.PNDocConvQueueItem item = null;

// Create the conversion item
item = new PNDocConvQueueServiceLib.PNDocConvQueueItem();

// Set conversion settings
item.Set("Devmode settings;Resolution", "300");
item.Set("Save;Output File Format", "JPEG");
item.Set("Save;Prompt", "0");
item.Set("Save;Overwrite", "1");
item.Set("Save;Color reduction", "Optimal");
item.Set("JPEG File Format;Color compression", "Medium Quality");
item.Set("JPEG File Format;Greyscale compression", "High Quality");
...
// convert the file
item.Convert("Microsoft Word",
    @"C:\Test\Report.docx",
    @"C:\Test\Out\ConvertedReport");
```



Code Sample - VB.NET

```
Dim item As PNDocConvQueueServiceLib.IPNDocConvQueueItem

' Create the conversion item
item = New PNDocConvQueueServiceLib.PNDocConvQueueItem()

' Set conversion settings
item.Set("Devmode settings;Resolution", "300")
item.Set("Save;Output File Format", "JPEG")
item.Set("Save;Prompt", "0")
item.Set("Save;Overwrite", "1")
item.Set("Save;Color reduction", "Optimal")
item.Set("JPEG File Format;Color compression", "Medium Quality")
item.Set("JPEG File Format;Greyscale compression", "High Quality")
...
' convert the file
item.Convert("Microsoft Word", _
            "C:\Test\Report.docx", _
            "C:\Test\Out\ConvertedReport")
```

Conversion Settings - JPEG File Format

Name:	JPEG File Format;Color compression
Values:	High Quality - High quality JPEG compression Medium Quality - Medium quality JPEG compression Low Quality - Low quality JPEG compression
Name:	JPEG File Format;Greyscale compression
Values:	High Quality - High quality JPEG compression Medium Quality - Medium quality JPEG compression Low Quality - Low quality JPEG compression

Processing

These options allow you to do extra processing to the image, such as trimming whitespace, cropping and resampling. Table values in **bold** text are the default value for that setting.

Sample Profile

```
<?xml version="1.0" encoding="utf-8"?>
<Profile Type="0"
    DisplayName="TIFF 300dpi Serialized with Trim"
    Description="Create TIFF serialized, trim whitespace">
    <Settings>

        <!-- Output file options -->
        <add Name="Devmode settings;Resolution" Value="300"/>
        <add Name="Save;Output File Format" Value="TIFF Serialized"/>
        <add Name="Save;Prompt" Value="0"/>
        <add Name="Processing;Trim Threshold" Value="0"/>
        <add Name="Processing;Trim left" Value="1"/>
        <add Name="Processing;Trim top" Value="1"/>
        <add Name="Processing;Trim bottom" Value="1"/>
        <add Name="Processing;Trim right" Value="1"/>
        ...

    </Settings>
</Profile>
```

Code Sample - C#

```
PNDocConvQueueServiceLib.PNDocConvQueueItem item = null;

// Create the conversion item
item = new PNDocConvQueueServiceLib.PNDocConvQueueItem();

// Set conversion settings
item.Set("Devmode settings;Resolution", "300");
item.Set("Save;Output File Format", "TIFF Serialized");
item.Set("Save;Prompt", "0");
item.Set("Processing;Trim Threshold", "0");
item.Set("Processing;Trim left", "1");
item.Set("Processing;Trim top", "1");
item.Set("Processing;Trim bottom", "1");
item.Set("Processing;Trim right", "1");
...
// convert the file
item.Convert("Microsoft Word",
    @"C:\Test\Report.docx",
    @"C:\Test\Out\ConvertedReport");
```



Code Sample - VB.NET

```
Dim item As PNDocConvQueueServiceLib.IPNDocConvQueueItem

' Create the conversion item
item = New PNDocConvQueueServiceLib.PNDocConvQueueItem()

' Set conversion settings
item.Set("Devmode settings;Resolution", "300")
item.Set("Save;Output File Format", "TIFF Serialized")
item.Set("Save;Prompt", "0")
item.Set("Processing;Trim Threshold", "0")
item.Set("Processing;Trim left", "1")
item.Set("Processing;Trim top", "1")
item.Set("Processing;Trim bottom", "1")
item.Set("Processing;Trim right", "1")
...
' convert the file
item.Convert("Microsoft Word", _
            "C:\Test\Report.docx", _
            "C:\Test\Out\ConvertedReport")
```

Conversion Settings - Processing

Name:	Processing;Units
	Specifies what unit of measurement is used for settings such as custom paper width or hardware margin. Units can be entered in inches (8.50in) or centimeters (21.59cm), provided the unit designation of inches (in) or centimeters (cm) is given. Also accepted are units entered in as hundredths of an inch (.01 Inches) or tenths of a millimeter(.1 Millimeters)
Values:	.01 Inches .1 Millimeters
Name:	Processing;Trim left
	Trim all areas from the left side of the page, based on the <i>Trim Threshold</i> below.
Values:	0 - Do not trim left side of page 1 - Trim left side of page
Name:	Processing;Trim top
	Trim all areas from the top edge of the page, based on the <i>Trim Threshold</i> below.
Values:	0 - Do not trim top of page 1 - Trim top of page

Conversion Settings - Processing	
Name:	Processing;Trim right
	Trim all areas from the right side of the page, based on the <i>Trim Threshold</i> below.
Values:	0 - Do not trim right side of page 1 - Trim right side of page
Name:	Processing;Trim bottom
	Trim all areas from the bottom edge of the page, based on the <i>Trim Threshold</i> below.
Values:	0 - Do not trim bottom of page 1 - Trim bottom of page
Name:	Processing;Trim Threshold
	All areas on the chosen sides of the image that fall at or below the chosen intensity level, or trim threshold. The intensity level is used to decide what pixels get thrown away. Colors are converted to a grayscale palette, and then compared to the chosen intensity level. Trimming on any side stops as soon as a pixel is encountered that is greater the chosen level. 0 is white, and 100 is black.
Values:	0-100
Name:	Processing;Crop
	Enable or disable the cropping options.
Values:	0 - Disable cropping 1 - Enable cropping
Name:	Processing;Crop Option
	Cropping can be specified in either of two ways: as page margins, or as a central area or region on the page.
Values:	0 - Crop region 1 - Crop margins

Conversion Settings - Processing	
Name:	Processing;Crop left Applies when Crop Option is set to <i>crop region</i> .
Values:	0 - 8000000 - Range in hundredths of an inch 0 - 20000000 - Range in tenths of a millimeter 0.000in - 80000.000in - Range in inches 0.000cm - 200000.000cm - Range in centimeters
Name:	Processing;Crop top Applies when Crop Option is set to 0 for <i>crop region</i> .
Values:	Same as <i>Processing;Crop left</i> above
Name:	Processing;Crop width Applies when Crop Option is set to 0 for <i>crop region</i> .
Values:	Same as <i>Processing;Crop left</i> above.
Name:	Processing;Crop height Applies when Crop Option is set to 0 for <i>crop region</i> .
Values:	Same as <i>Processing;Crop left</i> above
Name:	Processing;Crop margin left Applies when Crop Option is set to 1 for <i>crop margins</i> .
Values:	Same as <i>Processing;Crop left</i> above
Name:	Processing;Crop margin top Applies when Crop Option is set to 1 for <i>crop margins</i>
Values:	Same as <i>Processing;Crop left</i> above

Conversion Settings - Processing	
Name:	Processing;Crop margin right Applies when Crop Option is set to 1 for <i>crop margins</i>
Values:	Same as <i>Processing;Crop left above</i>
Name:	Processing;Crop margin bottom Applies when Crop Option is set to 1 for <i>crop margins</i>
Values:	Same as <i>Processing;Crop left above</i>
Name:	Processing;Copy Enable or disable the copy options. The Copy feature allow you to copy each page of the document to a larger or smaller page.
Values:	0 - Disable copy options 1 - Enable copy options
Name:	Processing;Copy to width The width of the new image
Values:	0 - 8000000 - Range in hundredths of an inch 0 - 20000000 - Range in tenths of a millimeter 0.000in - 80000.000in - Range in inches 0.000cm - 200000.000cm - Range in centimeters
Name:	Processing;Copy to height The height of the new image.
Values:	Same as <i>Processing;Copy to width above</i> .
Name:	Processing;Copy to IAM Left The desired left area margin settings for the new image.
Values:	Same as <i>Processing;Copy to width above</i>

Conversion Settings - Processing	
Name:	Processing;Copy to IAM Top
	The desired top area margin settings for the new image.
Values:	Same as <i>Processing;Copy to width</i> above
Name:	Processing;Copy to IAM Right
	The desired right area margin settings for the new image.
Values:	Same as <i>Processing;Copy to width</i> above
Name:	Processing;Copy to IAM Bottom
	The desired bottom area margin settings for the new image.
Values:	Same as <i>Processing;Copy to width</i> above
Name:	Processing;Copy H align
	How to horizontally align the copied image area.
Values:	Left - Align the copied image to the left on the page Middle - Align the copied image horizontally center on the page Right - Align the copied image to the right of the page
Name:	Processing;Copy V align
	How to vertically align the copied image area.
Values:	Top - Align the copied image to the top of the page Middle - Align the copied image vertically centered on the page Bottom - Align the copied image to the bottom of the page
Name:	Processing;Copy Page Scaling
	How to place the original page in the new image.
Values:	0 - Fit to Page 1 - Actual Size

Conversion Settings - Processing	
Name:	Processing;Copy Page Scaling Shrink Larger
	Scales the image down to fit the new image size if the original image is larger.
Values:	0 - Do not shrink page to fit 1 - Shrink page to fit
Name:	Processing;Copy Page Scaling Lock Aspect Ratio
	Use this option on to prevent distortion when scaling larger or smaller image to different image sizes.
Values:	0 - Do not maintain page aspect ratio when scaling 1 - Maintain page aspect ratio when scaling
Name:	Processing;Resample
	Scale the output file to a particular width and height in pixels, as a percentage of the original size, or by setting a new image resolution (DPI).
Values:	0 - Disable resampling options 1 - Enable resampling options
Name:	Processing;Resample Units
Values:	0 - Pixels 1 - Percentage 2 - DPI
Name:	Processing;Resample Lock Aspect Ratio
Values:	0 - Do not maintain page aspect ratio when resampling 1 - Maintain page aspect ratio when resampling
Name:	Processing;Resample Pixels Width
	Desired width in pixels.
Values:	0-4294967295 pixels, default width is 200 .

Conversion Settings - Processing	
Name:	Processing;Resample Pixels Height Desired height in pixels.
Values:	0-4294967295 pixels, default height is 200 .
Name:	Processing;Resample Width Percentage Change the width as a percentage of the original size.
Values:	1 to 500, default is 100 .
Name:	Processing;Resample Height Percentage Change the height as a percentage of the original size.
Values:	1 to 500, default is 100
Name:	Processing;Resample X DPI Change the X resolution of the image.
Values:	50-3600, default is 200
Name:	Processing;Resample Y DPI Change the Y resolution of the image.
Values:	50-3600, default is 200
Name:	Processing;Brightness Adjust Allows you to lighten or darken the images or text on your incoming pages.
Values:	--100 to -1 - darkens the image 0 - no change 1 to 100 - lightens the image
Name:	Processing;Rotate portrait Rotates portrait orientated images the desired degrees counter-clockwise.
Values:	0 , 90, 180, or 270

Conversion Settings - Processing**Name:** **Processing;Rotate landscape**

Rotates landscape orientated images the desired degrees counter-clockwise.

Values: **0, 90, 180, or 270**

Advanced Features

These options allow control of some of the advanced features, such as custom paper size and text extraction. Table values in **bold** text are the default value for that setting.



Sample Profile

```
<?xml version="1.0" encoding="utf-8"?>
<Profile Type="0"
  DisplayName="TIFF 300dpi Serialized Extract Text"
  Description="TIFF 300dpi Serialized Extract Text">
  <Settings>

    <!-- Output file options -->
    <add Name="Devmode settings;Resolution" Value="300"/>
    <add Name="Save;Output File Format" Value="TIFF Serialized"/>
    <add Name="Save;Prompt" Value="0"/>
    <add Name="Advanced Features;Extract Text" Value="1"/>
    <add Name="Advanced Features;Extract Text Layout" Value="Physical"/>
    ...

  </Settings>
</Profile>
```



Code Sample - C#

```
PNDocConvQueueServiceLib.PNDocConvQueueItem item = null;

// Create the conversion item
item = new PNDocConvQueueServiceLib.PNDocConvQueueItem();

// Set conversion settings
item.Set("Devmode settings;Resolution", "300");
item.Set("Save;Output File Format", "TIFF Serialized");
item.Set("Save;Prompt", "0");
item.Set("Advanced Features;Extract Text", "1");
item.Set("Advanced Features;Extract Text Layout", "Physical");
...
// convert the file
item.Convert("Microsoft Word",
  @"C:\Test\Report.docx",
  @"C:\Test\Out\ConvertedReport");
```

**Code Sample - VB.NET**

```
Dim item As PNDocConvQueueServiceLib.IPNDocConvQueueItem

' Create the conversion item
item = New PNDocConvQueueServiceLib.PNDocConvQueueItem()

' Set conversion settings
item.Set("Devmode settings;Resolution", "300")
item.Set("Save;Output File Format", "TIFF Serialized")
item.Set("Save;Prompt", "0")
item.Set("Advanced Features;Extract Text", "1")
item.Set("Advanced Features;Extract Text Layout", "Physical")
...
' convert the file
item.Convert("Microsoft Word", _
            "C:\Test\Report.docx", _
            "C:\Test\Out\ConvertedReport")
```

Conversion Settings - Advanced Features**Name: Advanced Features;Units**

Specifies what unit of measurement is used for settings such as custom paper width or hardware margin. Units can be entered in inches (8.50in) or centimeters (21.59cm), provided the unit designation of inches (in) or centimeters (cm) is given. Also accepted are units entered in as hundredths of an inch (.01 Inches) or tenths of a millimeter(.1 Millimeters).

Values: .01 Inches
.1 Millimeters

Name: Advanced Features;Custom Paper Enable

Enable or disable custom paper size.

Values: 0 - disable custom paper size
1 - enable custom paper size

Name: Advanced Features;Custom Paper Width

Specify the width of the custom paper size. *Custom Paper Enable* must be 1 for this to be used.

Values: 25 - 8000000 (default **850**) - Range in hundredths of an inch
64 - 20000000 - Range in tenths of a millimeter
0.250in - 80000.000in - Range in inches
0.640cm-200000.000cm - Range in centimeters

Conversion Settings - Advanced Features

Name:	Advanced Features;Custom Paper Height
	Specify the height of the custom paper size. <i>Custom Paper Enable</i> must be 1 for this to be used.
Values:	25 - 8000000 (default 1100) - Range in hundredths of an inch 64 - 20000000 - Range in tenths of a millimeter 0.250in - 80000.000in - Range in inches 0.640cm-200000.000cm - Range in centimeters
Name:	Advanced Features;Hardware Margin Left
Values:	0 - 100 (default = 0) - Range in hundredths of an inch 0 - 254 - Range in tenths of a millimeter 0.000in-1.000in - Range in inches 0.000cm-2.540cm - Range in centimeters
Name:	Advanced Features;Hardware Margin Top
Values:	0 - 100 (default = 0) - Range in hundredths of an inch 0 - 254 - Range in tenths of a millimeter 0.000in-1.000in - Range in inches 0.000cm-2.540cm - Range in centimeters
Name:	Advanced Features;Printer Area Margin Left
Values:	0 - 8000000 (default = 0) - Range in hundredths of an inch 0 - 20000000 - Range in tenths of a millimeter 0.000in - 80000.000in - Range in inches 0.000cm-200000.000cm - Range in centimeters
Name:	Advanced Features;Printer Area Margin Top
Values:	0 - 8000000 (default = 0) - Range in hundredths of an inch 0 - 20000000 - Range in tenths of a millimeter 0.000in - 80000.000in - Range in inches 0.000cm-200000.000cm - Range in centimeters
Name:	Advanced Features;Printer Area Margin Right
Values:	0 - 8000000 (default = 0) - Range in hundredths of an inch 0 - 20000000 - Range in tenths of a millimeter 0.000in - 80000.000in - Range in inches 0.000cm-200000.000cm - Range in centimeters

Conversion Settings - Advanced Features

Name: **Advanced Features;Printer Area Margin Bottom**

Values: 0 - 8000000 (default = **0**) - Range in hundredths of an inch
 0 - 20000000 - Range in tenths of a millimeter
 0.000in - 80000.000in - Range in inches
 0.000cm-200000.000cm - Range in centimeters

Name: **Advanced Features;Extract Text**

Enable this to also create a separate text file containing all of the textual elements of your source document.

Values: **0** - do not extract text
 1 - extract text into a separate text file

Name: **Advanced Features;Extract Text Filepath**

Path to file receiving extracted text.

Values: Full path to file to store text.

Name: **Advanced Features;Extract Text Layout**

Choose the layout of the text file.

Values: **Physical**
 Matches the format of the text in the original file.
Raw
 Saves the text in the order in which it was sent to the driver. This may not be the same order in the original file.
None
 No formatting is attempted. All text is written to the file as it is received

Name: **Advanced Features;Extract Text Encoding**

Choose the encoding of the text file.

Values: ANSI
 UTF-8
UTF-16

Conversion Settings - Advanced Features**Name:** **Advanced Features;Extract Text EOL****Values:** **Windows**
 Lines end with the CRLF line feed
 Mac
 Lines end with the LF line feed
 Unix
 Lines end with the CR line feed**Name:** **Advanced Features;Extract Text Emit Page Breaks****Values:** 0
 1**Name:** **Advanced Features;Control Strings Enabled****Values:** 0
 1

Endorsement Options

These options control the behavior of the endorsements that can be stamped on the output created by Document Conversion Service.

Endorsements are the placing of additional header and footer information at the top and bottom of each page. See also [Watermark Stamping](#) to add watermarks to the page content.

Header and footers can contain text such titles and page numbers. The default height of both the header and the footer is 12 points; this can be adjusted individually as needed.

Both the header and footer can be made up of three separate sections - a left section, a center section and a right section. The width of each section can be set individually to allow for text wrapping within each section. The default width for each section is the width of the page. Text in the top left and bottom left section is always left justified, text in the top center and bottom center section is always centered and text in top right and bottom right sections is always right justified.

The data displayed in each part of the header or footer can be formatted using the [Endorsement Formatting Codes](#) to add page number and total page count information to your header and footer text, as well as to display the text in different fonts, font sizes, colors and other text attributes such as bold, italic and underline. The default font used is Arial at 12 points.



Sample Profile

```
<?xml version="1.0" encoding="utf-8"?>
<Profile Type="0"
  DisplayName="Endorsed TIFF 300dpi"
  Description="Created TIFF with header and footers.">
  <Settings>

    <!-- Add header and footers for each page -->
    <add Name="Endorsements;Enable" Value="1"/>
    <add Name="Endorsements;HeaderHeightInPoints" Value="20"/>

    <!-- Change the text color and formatting. -->
    <add Name="Endorsements;HeaderLeftFormat"
      Value="&KFF0000&BInternal Use&B"/>

    <!-- Multiline. -->
    <add Name="Endorsements;HeaderRightFormat"
      Value="Confidential&x0A;DO NOT COPY"/>

    <!-- Change text style and size. -->
    <add Name="Endorsements;FooterHeightInPoints" Value="20"/>
    <add Name="Endorsements;FooterCenterFormat"
      Value="&'Courier'&P of &N"/>

    <!-- Output file options -->
    <add Name="Devmode settings;Resolution" Value="300"/>
    <add Name="Save;Output File Format" Value="TIFF Multipaged"/>
    ...

  </Settings>
</Profile>
```



Code Sample - C#

```
PNDocConvQueueServiceLib.PNDocConvQueueItem item = null;

// Create the conversion item
item = new PNDocConvQueueServiceLib.PNDocConvQueueItem();

// Set conversion settings
item.Set("Endorsements;HeaderHeightInPoints", "20");
item.Set("Endorsements;HeaderLeftFormat",
"&KFF0000&BInternal Use&B");
item.Set("Endorsements;HeaderRightFormat",
"Confidential\r\nDO NOT COPY");

item.Set("Endorsements;FooterHeightInPoints", "20");
item.Set("Endorsements;FooterCenterFormat",
"&'Courier'&P of &N");
...
// convert the file
item.Convert("Microsoft Word",
@"C:\Test\Report.docx",
@"C:\Test\Out\ConvertedReport");
```



Code Sample - VB.NET

```
Dim item As PNDocConvQueueServiceLib.IPNDocConvQueueItem

' Create the conversion item
item = New PNDocConvQueueServiceLib.PNDocConvQueueItem()

' Set conversion settings
item.Set("Endorsements;Enable", "1")

item.Set("Endorsements;HeaderHeightInPoints", "20")
item.Set("Endorsements;HeaderLeftFormat", _
"&KFF0000&BInternal Use&B")
item.Set("Endorsements;HeaderRightFormat", _
"Confidential\r\nDO NOT COPY")

item.Set("Endorsements;FooterHeightInPoints", "20")
item.Set("Endorsements;FooterCenterFormat", _
"&'Courier'&P of &N")
...
' convert the file
item.Convert("Microsoft Word", _
"C:\Test\Report.docx", _
"C:\Test\Out\ConvertedReport")
```

Conversion Settings - Endorsements Header and Footer Options

Name:	Endorsements;Enable
Values:	0 - Do not add endorsements 1 - Add specified endorsements to each page

Conversion Settings - Endorsements Header and Footer Options	
Name:	Endorsements;HeaderHeightInPoints
Values:	The height of the header area in points. The default is 12 points.
Name:	Endorsements;HeaderLeftWidthInPoints
Values:	The width of the left section of the header area in points. The default is the width of the page.
Name:	Endorsements;HeaderCenterWidthInPoints
Values:	The width of the center section of the header area in points. The default is the width of the page.
Name:	Endorsements;HeaderRightWidthInPoints
Values:	The width of the right section of the header area in points. The default is the width of the page.
Name:	Endorsements;HeaderLeftFormat
Values:	The text, with Endorsement Formatting Codes as needed, to put in the left section of the header.
Name:	Endorsements;HeaderCenterFormat
Values:	The text, with Endorsement Formatting Codes as needed, to put in the center section of the header.
Name:	Endorsements;HeaderRightFormat
Values:	The text, with Endorsement Formatting Codes as needed, to put in the right section of the header.
Name:	Endorsements;FooterHeightInPoints
Values:	The height of the footer area in points. The default is 12 points.

Conversion Settings - Endorsements Header and Footer Options	
Name:	Endorsements;FooterLeftWidthInPoints
Values:	The width of the left section of the footer area in points. The default is the width of the page.
Name:	Endorsements;FooterCenterWidthInPoints
Values:	The width of the center section of the footer area in points. The default is the width of the page.
Name:	Endorsements;FooterRightWidthInPoints
Values:	The width of the right section of the footer area in points. The default is the width of the page.
Name:	Endorsements;FooterLeftFormat
Values:	The text, with Endorsement Formatting Codes as needed, to put in the left section of the footer.
Name:	Endorsements;FooterCenterFormat
Values:	The text, with Endorsement Formatting Codes as needed, to put in the center section of the footer.
Name:	Endorsements;FooterRightFormat
Values:	The text, with Endorsement Formatting Codes as needed, to put in the right section of the footer.

Endorsement Formatting Codes

The following formatting codes are used to format the text strings placed in the headers and footers. If you are using the XML profiles to configure the endorsements you will need to use the XML character entities `&` and `"` to represent the ampersand (&) and quotation marks (") to allow the XML data to be interpreted correctly.

Header and Footer Formatting Codes		
XML Code	String Code	Description
<code>&P</code>	<code>&P</code>	<p>This code is replaced by the current page number.</p> <p>XML Example:</p> <pre><add Name="Endorsements;HeaderLeftFormat" Value="Page &P" /></pre> <p>String Example:</p> <pre>item.Set("Endorsements;HeaderLeftFormat", "Page &P")</pre>
<code>&N</code>	<code>&N</code>	<p>This code is replaced by the total number of pages in the output file.</p> <p>XML Example:</p> <pre><add Name="Endorsements;HeaderLeftFormat" Value="Page &P of &N" /></pre> <p>String Example:</p> <pre>item.Set("Endorsements;HeaderLeftFormat", "Page &P of &N")</pre>
<code>&B</code>	<code>&B</code>	<p>Turns bold formatting on and off. All text after the first occurrence of the formatting code will be bold until the same formatting code is encountered again.</p> <p>XML Example:</p> <pre><add Name="Endorsements;HeaderLeftFormat" Value="&BInternal Use&B - Confidential" /></pre> <p>String Example:</p> <pre>item.Set("Endorsements;HeaderLeftFormat", "&BInternal Use Only&B - Confidential")</pre>
<code>&I</code>	<code>&I</code>	<p>Turns italic formatting on and off. All text after the first occurrence of the formatting code will be italicized until the same formatting code is encountered again.</p> <p>XML Example:</p> <pre><add Name="Endorsements;HeaderLeftFormat" Value="&I Do Not Copy&I - Confidential" /></pre> <p>String Example:</p> <pre>item.Set("Endorsements;HeaderLeftFormat",</pre>

Header and Footer Formatting Codes		
XML Code	String Code	Description
		"&IDo Not Copy&I - Confidential")
&U	&U	<p>Turns font underlining on and off. All text after the first occurrence of the formatting code will be underlined until the same formatting code is encountered again.</p> <p>XML Example:</p> <pre><add Name="Endorsements;HeaderLeftFormat" Value="&U;UDo Not Copy&U - Confidential"/></pre> <p>String Example:</p> <pre>item.Set("Endorsements;HeaderLeftFormat", "&UDo Not Copy&U - Confidential")</pre>
&S	&S	<p>Turns font strike through formatting on and off. All text after the first occurrence of the formatting code will be struck through (a line down the middle of the text) until the same formatting code is encountered again.</p> <p>XML Example:</p> <pre><add Name="Endorsements;HeaderLeftFormat" Value="&S;SInternal Use&S - Confidential"/></pre> <p>String Example:</p> <pre>item.Set("Endorsements;HeaderLeftFormat", "&SInternal Use Only&S - Confidential")</pre>
&X	&X	<p>Turns font superscript formatting on and off. All text after the first occurrence of the formatting code will be printed in superscript (appears smaller than the normal line of type and is set slightly above it) until the same formatting code is encountered again.</p> <p>XML Example:</p> <pre><add Name="Endorsements;HeaderLeftFormat" Value="This is &Xsuperscript text&X - Confidential"/></pre> <p>String Example:</p> <pre>item.Set("Endorsements;HeaderLeftFormat", "This is &Xsuperscript text&X - Confidential")</pre>
&Y	&Y	<p>Turns font subscript formatting on and off. All text after the first occurrence of the formatting code will be printed in subscript (appears smaller than the normal line of type and is set slightly below it) until the same formatting code is encountered again.</p>

Header and Footer Formatting Codes		
XML Code	String Code	Description
		<p>XML Example:</p> <pre><add Name="Endorsements;HeaderLeftFormat" Value="This is &Ysubscript text&Y - Confidential"/></pre> <p>String Example:</p> <pre>item.Set("Endorsements;HeaderLeftFormat", "This is &Ysubscript text&Y - Confidential")</pre>
&'fontname'	&'fontname'	<p>Sets the font to be used for the following text. All text after the occurrence of the formatting code will be printed in the specified font until another font formatting code is encountered again. The default font is Arial.</p> <p>XML Example:</p> <pre><add Name="Endorsements;HeaderLeftFormat" Value="This is Arial and &'Verdana'this is Verdana."/></pre> <p>String Example:</p> <pre>item.Set("Endorsements;HeaderLeftFormat", "This is Arial and &'Verdana'this is Verdana.")</pre>
&n	&n	<p>Sets the font size, in points, to be used for the following text, where n is replaced with the desired point size. All text after the occurrence of the formatting code will be printed in the specified font size until another font size formatting code is encountered again. The default font size is 12 points.</p> <p>XML Example:</p> <pre><add Name="Endorsements;HeaderLeftFormat" Value="&14This is Arial 14 point."/></pre> <p>String Example:</p> <pre>item.Set("Endorsements;HeaderLeftFormat", "&14This is Arial 14 point.")</pre>
&K000000	&K000000	<p>Changes the color of the text. All text after the occurrence of the formatting code will be printed in the color specified until another color formatting code is encountered again. The default color is Black. The color is specified as six character RGB code.</p> <p>XML Example:</p> <pre><add Name="Endorsements;HeaderLeftFormat" Value="This is &KFF0000Red, this is &K00FF00Green."/></pre>

Header and Footer Formatting Codes		
XML Code	String Code	Description
		<p>String Example:</p> <pre>item.Set("Endorsements;HeaderLeftFormat", "This is &KFF0000Red, this is &K00FF00Green.")</pre>
&&	&&	<p>Allows the insertion of an ampersand character into the text.</p> <p>XML Example:</p> <pre><add Name="Endorsements;HeaderLeftFormat" Value="Printed by Company &amp;&amp;Company"/></pre> <p>String Example:</p> <pre>item.Set("Endorsements;HeaderLeftFormat", "Printed by Company && Company")</pre>

	\r\n	<p>Allows the insertion of a newline character into the text.</p> <p>XML Example:</p> <pre><add Name="Endorsements;HeaderLeftFormat" Value="Line 1&#x0A;Line 2"/></pre> <p>String Example:</p> <pre>item.Set("Endorsements;HeaderLeftFormat", "Line 1\r\nLine 2.")</pre>

Watermark Stamping

These options allow the placement of a centered, diagonal watermark on each page. The watermark text runs from bottom left to the top right of the page with the outline of each letter being printed. Table values in **bold** text are the default value for that setting.



Sample Profile

```
<?xml version="1.0" encoding="utf-8"?>
<Profile Type="0"
  DisplayName="TIFF 300dpi Serialized Extract Text"
  Description="TIFF 300dpi Serialized Extract Text">
  <Settings>

    <!-- Output file options -->
    <add Name="Devmode settings;Resolution" Value="300"/>
    <add Name="Save;Output File Format" Value="TIFF Serialized"/>
    <add Name="Save;Prompt" Value="0"/>
    <add Name="WatermarkStamp;Enabled" Value="1"/>
    <add Name="WatermarkStamp;CenteredDiagonalText" Value="DRAFT"/>
    <add Name="WatermarkStamp;CenteredDiagonalFontSizeInPoints" Value="36"/>
    ...

  </Settings>
</Profile>
```



Code Sample - C#

```
PNDocConvQueueServiceLib.PNDocConvQueueItem item = null;

// Create the conversion item
item = new PNDocConvQueueServiceLib.PNDocConvQueueItem();

// Set conversion settings
item.Set("Devmode settings;Resolution", "300");
item.Set("Save;Output File Format", "TIFF Serialized");
item.Set("Save;Prompt", "0");
item.Set("WatermarkStamp;Enabled", "1");
item.Set("WatermarkStamp;CenteredDiagonalText", "DRAFT");
item.Set("WatermarkStamp;CenteredDiagonalFontSizeInPoints", "36");
...
// convert the file
item.Convert("Microsoft Word",
  @"C:\Test\Report.docx",
  @"C:\Test\Out\ConvertedReport");
```



Code Sample - VB.NET

```
Dim item As PNDocConvQueueServiceLib.IPNDocConvQueueItem

' Create the conversion item
item = New PNDocConvQueueServiceLib.PNDocConvQueueItem()

' Set conversion settings
item.Set("Devmode settings;Resolution", "300")
item.Set("Save;Output File Format", "TIFF Serialized")
item.Set("Save;Prompt", "0")
item.Set("WatermarkStamp;Enabled", "1")
item.Set("WatermarkStamp;CenteredDiagonalText", "DRAFT")
item.Set("WatermarkStamp;CenteredDiagonalFontSizeInPoints", "36")
...
' convert the file
item.Convert("Microsoft Word", _
            "C:\Test\Report.docx", _
            "C:\Test\Out\ConvertedReport")
```

Conversion Settings - Advanced Features

Name:	WatermarkStamp;Enabled
	Enable or disable the watermark stamping feature.
Values:	0 - disable watermark stamping 1 - enable watermark stamping
Name:	WatermarkStamp;CenteredDiagonalText
Values:	The text to display as the watermark stamp.
Name:	WatermarkStamp;CenteredDiagonalFontSizeInPoints
Values:	The font size of the watermark text in points. Default is 36.

Advanced Configuration

The topics covered in this section discuss configuration options that can be applied to help you get the most from Document Conversion Service. Reading this section will allow you to tailor the resources used by the conversion service to give you optimal performance.



Changing the Application Configuration

When making changes to the application configuration file, Document Conversion Service will need to be restarted to pick up the changes.

The topics discussed will allow you to

- set the number of parallel conversions based in the number of CPUs and cores on the computer
- only load the converters for the document types you need to convert
- adjust the application pool to meet the demands of the number of documents you expect to process

Configuring Parallel Processing

The Document Conversion Service is designed to process many documents in parallel, up to the limits of your license model. The following settings are used to control the number documents and printers in parallel:

Setting Name	Value
NumberOfDocumentsInParallel	Number of documents that can be converted at the same time.
NumberOfPrinters	Controls the size of the Document Conversion Service printer pool.

These values are set to the keyword "auto" when first installed, which means that Document Conversion Service will automatically determine an appropriate value for these numbers based on the number of CPU's and cores on your computer. We recommend you leave this set to "auto" to get the best experience from Document Conversion Service. Setting this to a value that is too high for the capabilities of the computer can cause the computer to work very slowly.

The formula used for determining how many documents your system can handle is to multiply the number of cores per CPU by the number of CPU's and multiply that by 1.5. As an example, a single CPU system with 4 cores would be able to process 6 documents in parallel at a time:

```
(number of cores per CPU × number of CPU's) × 1.5 =  
(4 × 1) × 1.5 = 6 documents in parallel
```

Once the maximum value for the number of documents has been determined, this number is also compared against your purchased license (or the fact that you are running a trial version) and capped at the number of document in parallel allowed by your license model. You can, of course, always set this number lower as needed to balance this with other applications and services running on your computer.

Setting the Number of Documents in Parallel

The number of documents to process in parallel is stored as a collection of key-value pairs written in XML in the *General* section of the Document Conversion Service application configuration file. See [General Application Settings](#) for a complete list of all settings that can be changed in the General section.



General Configuration Section

```
<!-- General configuration options-->  
<General>  
  <Settings>  
    <!-- Maximum number of printers and threads is determined by your license model. -->  
    <add Name="NumberOfDocumentsInParallel" Value="auto"/>  
    <add Name="NumberOfPrinters" Value="auto"/>  
    ...  
  </Settings>  
</General>
```

Opening the Configuration File

Go to **DCS Dashboard** - DCS-Settings - Edit DCS Configuration to edit the configuration file using a visual GUI.

You can also go to Start - All Programs - PEERNET Document Conversion Service 3.0 - Edit DCS Configuration File to edit the configuration file using the DCS Editor.

The configuration file can also be opened in any XML editor and can be found here:

Configuration file location:

C:\Program Files\Document Conversion Service 3.0\Core\PNJobItemProcessor.exe.config

Setting the Number of Documents in Parallel

1. Once open in the DCS Editor, find and locate the **<General>** section.
2. In the **<Settings>** section, modify the NumberOfDocumentsInParallel value to the desired number to change how many documents are converted in parallel. Leave this value as "auto" to have Document Conversion Service optimize the number of documents in parallel based on your computer's capabilities.
3. The NumberOfPrinters controls the size of the Document Conversion Service printer pool. For optimal performance the size of the printer pool needs to match the NumberOfDocumentsInParallel setting. This value can also be left to "auto".
4. Save the edited file. If Document Conversion Service is running you will need to restart the conversion service to apply your new changes.



General Configuration Section - modified

```
<!-- General configuration options-->
<General>
  <Settings>
    <!-- Maximum number of printers and threads is determined by your license model. -->
    <add Name="NumberOfPrinters" Value="3"/>
    <add Name="NumberOfDocumentsInParallel" Value="3"/>
    ...
  </Settings>
</General>
```

Restoring the Configuration File

A backup copy of the original configuration file is stored in the following location for easy recovery.

Configuration file location:

C:\Program Files\Document Conversion Service 3.0\Core\Backup\PNJobItemProcessor.exe.config

Document Conversion Service Startup and Shutdown

The settings below control the startup and shutdown behavior of Document Conversion Service.

In most cases the values provided will be sufficient and will not need to be changed.

Setting Name	Value
RunSelfHealForCoreServices	Detects proper installation of required components and will attempt to self-heal if any components are found missing. This check is always performed by default. We do not recommend disabling this check.
RunSelfHealForOtherServices	Optional detection and self-heal of secondary components; detects proper installation and will attempt to self-heal if any components are found missing. This check is performed by default.
ThreadInitBeforeSignalRunningState	How long to wait for the converter factory threads to initialize and be ready to process documents.
MaxWaitForProcessingTimeoutInMinutes	The maximum amount of time, in minutes, to wait for a document to signal that it is being converted. The minimum timeout is 5 minutes, the default is 30 minutes.
SessionWaitForAllJobsCompletedTimeout	The maximum amount of time to wait for all documents to finish printing when shutting Document Conversion Service down. This setting is also documented in Document Conversion Service Printer Pool .
WaitForSrv10ToClose	The Document Conversion Service uses the PNSrv10 component and cannot close until that component has exited first. The default amount of time to wait is 60 seconds, this component normally exits in just over 30 seconds.
RestartServiceInHours	When set to the default value of 0, the Document Conversion Service is never restarted. If desired, the service can be set to be automatically restarted anywhere from every hour up to every seven days (168 hours).

Changing the Service Behavior

In most cases you will never need to change any of the default values set above upon install. If you do, make sure you keep a backup of your original settings.

Opening the Configuration File

Go to **DCS Dashboard** - DCS-Settings - Edit DCS Configuration to edit the configuration file using a visual GUI.

You can also go to Start - All Programs - PEERNET Document Conversion Service 3.0 - Edit DCS Configuration File to edit the configuration file using the DCS Editor.

The configuration file can also be opened in any XML editor and can be found here:

Configuration file location:

C:\Program Files\Document Conversion Service 3.0\Core\PNJobItemProcessor.exe.config

Changing the Service Behavior Values

These values are set in the general application settings section.

1. If you need to set the Document Conversion Service service to be restarted automatically, you can change the RestartServiceInHours setting.
2. The SessionWaitForAllJobsCompletedTimeout value is used when the Document Conversion Service is shutting down. This is the maximum amount of time to wait for all printing documents in the pool of printers to complete.
3. Save the edited file. If Document Conversion Service is running you will need to restart the conversion service to apply your new changes,



General Configuration Section - Service Startup & Shutdown

```
<General>
  <Settings>
    ...

    <add Name="SessionWaitForAllJobsCompletedTimeout" Value="300000"/>
    <add Name="ThreadInitBeforeSignalRunningState" Value="20000"/>

    <add Name="MaxWaitForProcessingTimeoutInMinutes" Value="30"/>

    <add Name="RestartServiceInHours" Value="0"/>

    <add Name="WaitForSrv10ToClose" Value="60000"/>

    <add Name="RunSelfHealForCoreServices" Value="true"/>
    <add Name="RunSelfHealForOtherServices" Value="true"/>
  </Settings>
</General>
```

Restoring the Configuration File

A backup copy of the original configuration file is stored in the following location for easy recovery.

Configuration file location:

C:\Program Files\Document Conversion Service 3.0\Core\Backup\PNJobItemProcessor.exe.config

Document Conversion Service Printer Pool

To perform optimally the Document Conversion Service printers in the printing pool need certain timeouts, such as how long to wait for a printer to become available, or how long to wait for a job to appear in the printer queue. In most cases the values provided will be sufficient and will not need to be changed.

Other settings, such as how many times to try to convert the document, or to limit how many pages can be converted can also be set here. These settings can be overridden by the individual settings for the converters in their `<PluginFactory>` section if needed.

Setting Name	Value
PrintSessionWaitTimeout*	How long the converter will wait to get access to a printing session. This value is entered in microseconds (1 second = 1000 microseconds).
PrintSessionFirstJobTimeout*	This setting is applied to the printing session used by the converter and determines how long the printing session will wait for a job to start spooling in the printer queue before releasing the printing session back into the printer pool. This value is entered in microseconds (1 second = 1000 microseconds).
PrintSessionAvailableTimeout*	This setting is applied to the printing session used by the converter and determines how long to wait between jobs entering the queue before releasing the printing session back into the printer pool. This value is entered in microseconds (1 second = 1000 microseconds).
PrintSessionWaitOnSpoolingTimeout*	How long the converter will wait for each job to start spooling in the printer queue. This value is entered in microseconds (1 second = 1000 microseconds).
PrintSessionWaitOnCompleteTimeout*	This is NOT the total amount of time for the document to convert, it is the amount of idle time used to determine when to cancel a document being created. If the converter does not see any progress (pages being converted) in this amount of time the document is canceled.
SessionWaitForAllJobsCompletedTimeout	The maximum amount of time to wait for all documents to finish printing when shutting Document Conversion Service down. This value is entered in microseconds (1 second = 1000 microseconds).
MaxRetryAttempts*	Controls the number of times to retry converting a document if it was not successful on printing. Minimum value is 0, meaning we will not retry, and the maximum number of

Setting Name	Value
	retries is 5. The default is 2.
MaxSpooledPagesAllowed*	<p>Sets the maximum number of pages that are allowed to be printed/spooled. The default value when this is not provided is 0, meaning there is no limit. The install sets this value to 20,000 pages to prevent extremely large documents for stalling the converting process. If a document exceeds this count, it enters an error state and no file is created. To limit how many pages to convert see the PageRange setting in General Converter Options.</p> <p>This option can also be overridden on a per document basis using profiles as described in Creating and Customizing Profiles.</p>
ZeroByteFiles*	<p>Determines if files with a size of zero (0 bytes) are skipped or failed when processed. When set to Fail, an error is produced. When set to Skip, the file is skipped and a message is produced instead of an error. Default behaviour is to fail the file.</p>

* These settings can be overridden by the individual settings for the converters in their `<PluginFactory>` section if needed.

Changing the Printer Pool Behavior

Opening the Configuration File

Go to **DCS Dashboard** - DCS-Settings - Edit DCS Configuration to edit the configuration file using a visual GUI.

You can also go to Start - All Programs - PEERNET Document Conversion Service 3.0 - Edit DCS Configuration File to edit the configuration file using the DCS Editor.

The configuration file can also be opened in any XML editor and can be found here:

Configuration file location:

C:\Program Files\Document Conversion Service 3.0\Core\PNJobItemProcessor.exe.config

Changing the Timeout Values

All timeout values are specified in milliseconds except for **MaxWaitForProcessingTimeoutInMinutes** and **RestartServiceInHours**.

1. If you are converting very large documents you may need to adjust the `PrintSessionWaitOnCompleteTimeout` value to a value larger than the default of 180000ms (3 minutes).

2. The SessionWaitForAllJobsCompletedTimeout value is used when the Document Conversion Service is shutting down. This is the maximum amount of time to wait for all printing documents in the pool of printers to complete.
3. Save the edited file. If Document Conversion Service is running you will need to restart the conversion service to apply your new changes,



General Configuration Section - Printer Pool Settings

```
<General>
  <Settings>

    <!-- The following values can be overridden in the individual -->
    <!-- converter settings below for converter customization -->
    <add Name="PrintSessionWaitTimeout" Value="5000"/>
    <add Name="PrintSessionFirstJobTimeout" Value="60000"/>
    <add Name="PrintSessionAvailableTimeout" Value="250"/>
    <add Name="PrintSessionWaitOnSpoolingTimeout" Value="10000"/>
    <add Name="PrintSessionWaitOnCompleteTimeout" Value="180000"/>
    <!-- End of converter overridables -->

    <add Name="SessionWaitForAllJobsCompletedTimeout" Value="300000"/>
    <add Name="ThreadResetSleepBeforeSignalRunningState" Value="20000"/>

    <add Name="MaxRetryAttempts" Value="2"/>
    <add Name="MaxSpooledPagesAllowed" Value="0"/>
  </Settings>
</General>
```

Restoring the Configuration File

A backup copy of the original configuration file is stored in the following location for easy recovery.

Configuration file location:

C:\Program Files\Document Conversion Service 3.0\Core\Backup\PNJobItemProcessor.exe.config

Controlling the Converters

By default Document Conversion Service automatically attempts to load all included converters. For a converter that requires a native application to load, that application must also be installed. Each converter also uses an application pool (multiple running instances of the application) to allow for parallel document processing.

You can reduce the amount of resources Document Conversion Service uses by only loading the converters for file types that you need to convert.



Applications Factories and Required Applications

Most converters use an application to print the file to the Document Conversion Service to convert the file.

See [What Files Can I Convert?](#) for a complete list of each converter and its associated application. If you need to use that converter you will also have the matching application installed.

The converters are each defined in their own sections in Document Conversion Service's application configuration file. Each converter definition consists of an application factory component and a converter factory component that uses the application factory.

The application factory component controls if the converter will be loaded and the maximum number of instances of each application that can be running at any one time (*application pooling*).

It also controls when any one in-use application instance is closed and a new one started to replace it in the pool (*recycled*). The recycling of an application is controlled both by number of documents processed and by the virtual size of the running application.

The converter factory component is responsible for any custom conversion settings particular to that converter and its application.

The Application Factory Component

Each converter is described in the Document Conversion Service application configuration file as an `<AppFactory>`.

The configuration file contains an `<AppFactories>` collection of `<AppFactory>` items; one for each converter. Each application factory is described in its own `<AppFactory>` section using a collection of key-value pairs in the `<Settings>` collection.

The `<AppFactories>` collection also has its own `<Settings>` collection that is used to describe default settings for all application factories. If any individual application factory does not contain one of the settings the *default setting* from this section is used.

See [Application Factory Settings](#) for a complete list of all settings.

The Conversion Component

The Document Conversion Service application configuration file contains the `<PluginFactories>` collection of `<PluginFactory>` items. There is a matching `<PluginFactory>` for each converter, or `<AppFactory>` above. Each `<PluginFactory>` section is described using a collection of key-value pairs in the `<Settings>` collection.

The `<PluginFactories>` collection also has its own `<Settings>` collection that is used to describe default settings for the `<PluginFactory>` section for each converters . If any individual `<PluginFactory>` does not contain one of the settings the *default setting* from this section is used.

Enabling and Disabling the Converters

Each converter, or AppFactory can be enabled or disabled. You will likely want to disable converters that convert file types you are not interested in converting. See [Enabling and Disabling Converters](#) to learn how to customize the list of available converters.

The Application Pool

Document Conversion Service uses an application pool for each converter to provide the ability to process multiple documents of the same type at the same time.

The application factory for each converter controls the maximum size of its application pool through its **MaxInstances** setting. This value is set to "auto" when first installed, meaning that Document Conversion Service will automatically set the application pool size to a value appropriate to the capabilities of your computer, and only limited by your license model. This is the recommended setting to get the best experience from Document Conversion Service. Setting this to a value that is too high for the capabilities of the computer can cause the computer to work very slowly.

The application pool is dynamic and self regulating. Each pool starts with a single instance at the beginning and adds new instances, up to the maximum allowed, as they are needed to accommodate the volume of documents of that type that are being converted.

After an application in the pool has been idle, meaning it has not been used by a converter for a set period of time, that instance is removed from the pool, freeing up resources. This idle timeout period can be configured if needed, or set to zero (0) to have the applications stay in the pool indefinitely. For best performance we recommend leaving the idle timeout set to its default of an hour.

Each application in the application pool itself can be recycled at preset intervals based on the number of documents processed and the virtual size of the running application. This allows you to tailor the resources used to meet the capabilities of the computer the conversion service is running on.



Application Virtual Size

To check the Virtual Size of a running application we recommend using **Process Explorer** from [SysInternals](#) and adding the appropriate column. You cannot see the virtual size as a single column on Task Manager's Process tab.

The application factory for each converter uses the following settings to control the application pool. Each of these settings can be set individually on the `<AppFactory>` for each converter, or at the global `<AppFactories>` level to control all converters.

Setting	Value
MaxInstances	<p>The maximum size of the application pool for this converter. For best performance leave this set to "auto" to have the size of the application pool tailored to the capabilities of your computer. If this setting is not provided, or set to 0 or less, a single application instance will be created.</p> <p>The application pool is dynamic and will start with a single application in the pool with new applications added as needed. If an application in the pool is idle, meaning it has not processed any conversions, for a certain amount of time it is removed from the pool. This is controlled by the AppTeardownIdleTimeout setting below.</p>
MaxRetryAttempts	Controls the number of times to retry converting a document if it was not successful on printing.

Setting	Value
	<p>Minimum value is 0, meaning we will not retry, and the maximum number of retries is 5. The default is 2.</p> <p>Setting this value in the application pool level will override this setting in the Document Conversion Service Printer Pool section.</p>
MaxSpooledPagesAllowed	<p>Sets the maximum number of pages that are allowed to be printed/spooled. The default value value is 0, meaning there is no limit. If a document exceeds this count, it enters an error state and no file is created. To limit how many pages to convert see the <i>PageRange</i> setting in General Converter Options.</p> <p>Setting this value in the application pool level will override this setting in the Document Conversion Service Printer Pool section.</p> <p>This option can also be overridden on a per document basis using profiles as described in Creating and Customizing Profiles.</p>
RecycleThreshold	<p>Maximum number of documents each application can process before it is recycled and a new instance started to replace it.</p> <p>This is set to 0 by default, meaning the application doesn't recycle.</p>
ReadyThreshold	<p>The maximum length of time to wait after the application has been initialized before Document Conversion Service initiates communication with the application. This value may need to be increased for machines running high volume with many other applications running.</p>
AppInitializationThreshold	<p>Some applications need more time than others to complete their initialization. Enter in the length of time, in microseconds, to wait for the application to initialize.</p>
AppTeardownIdleTimeout	<p>The amount of time, in milliseconds, to wait before an idle application is closed and removed from the application pool. An idle application is one that has not processed any conversions in the specified time period. These idle applications are removed from the pool to free up resources. They are added back in on demand as needed.</p> <p>This is set to 3,600,000 milliseconds (1 hour) by default in the global <code><AppFactories></code> section.</p> <p>If this is set to 0, the applications in the pool will start dynamically but will not be dynamically removed from the pool. They will only be removed if they are</p>

Setting	Value
	recycled due to conversion failure or the settings for RecycleThreshold , RecycleVirtualSizeThreshold , RecycleGDIandUserHandleCountThreshold , and RecycleProcessHandleCountThreshold .
AppSynchronousPrintModeCheckPrintQueue	Some applications print synchronously, meaning control doesn't return to Document Conversion Service until the file has been sent to the printer. In some cases we need to check the printer queue to see if the print action actually submitted a job. If it has not we fail the conversion gracefully. This setting is false for most applications.
RecycleVirtualSizeThreshold	The size (in 1024KB blocks) at which to recycle the application. For example, 1400000 is 1.4GB meaning the application will be recycled when its virtual size is larger than 1.4 GB. Is it important to keep this value below the 2GB virtual size for 32-bit applications. While you can disable the application recycling based on Virtual Size by setting this to 0 or removing the value completely, we do not recommend this.
RecycleGDIandUserHandleCountThreshold	The maximum number of combined user and GDI handles allowed for each application instance. When this number of user and GDI handles exceed this threshold the application will be recycled and a new instance started to replace it in the application pool. If this value is not set, or set to zero, the maximum number of combined handles is 8000.
RecycleProcessHandleCountThreshold	The maximum number of process handles allowed for each application instance. When this number exceeds this threshold the application will be recycled and a new instance started to replace it in the application pool. If this value is not set, or set to zero, the maximum number of combined handles is 2000.
ZeroByteFiles	Determines if files with a size of zero (0 bytes) are skipped or failed when processed. When set to Fail , an error is produced. When set to Skip , the file is skipped and a message is produced instead of an error. Default behaviour is to fail the file.

Modifying the Application Pool

As the application pool is dynamic and self-regulating, in most cases you should not need to configure the individual instances of the application pool on a per-converter basis. If you do decide you need to, the following steps show you how this can be done.

Opening the Configuration File

Go to **DCS Dashboard** - DCS-Settings - Edit DCS Configuration to edit the configuration file using a visual GUI.

You can also go to Start - All Programs - PEERNET Document Conversion Service 3.0

- Edit DCS Configuration File to edit the configuration file using the DCS Editor.

The configuration file can also be opened in any XML editor and can be found here:

Configuration file location:

C:\Program Files\Document Conversion Service 3.0\Core\PNJobItemProcessor.exe.config

Changing the Application Pool Size

As an example, if you mainly need to convert Word and PDF documents, and only occasionally need to convert Excel documents, you can reduce the size of the application pool for the Excel converter and increase the pools used by the Word and PDF converters. This would give you higher throughput on the documents you need to convert more often.

The sample below shows a possible configuration for this scenario:

- The Microsoft Word and Adobe Acrobat Reader converter will both have an application pool of 5.
 - The Microsoft Excel converter has an application pool of 2.
 - The default *MaxInstances*, if not provided in the `<AppFactory>` section, is *auto* as set in the `<Settings>` section at the bottom of the `<AppFactories>` section. When set to *auto* the size of the application pool is tailored based on the capabilities of your computer using the same formula as [Configuring Parallel Processing](#).
1. In the `<AppFactories>` section find the `<AppFactory>` section for the converter whose application pool you want to adjust.
 2. Set the *MaxInstances* value to an appropriate higher or lower value.



AppFactories Configuration Section

```
<AppFactories>
  <Factories>

    <AppFactory Name="Microsoft Word"
      Type="PEERNET.PNDocConv.Applications.PNWordApplicationFactory"
      Assembly="PNWordApplicationFactory">
      <Settings>
        <add Name="Enabled" Value="auto"/>
        <add Name="MaxInstances" Value="5"/>
        <add Name="RecycleThreshold" Value="100"/>
      </Settings>
    </AppFactory>

    <AppFactory Name="Adobe Acrobat Reader"
      Type="PEERNET.PNDocConv.Applications.PNAcrobatReaderApplicationFactory"
      Assembly="PNAcrobatReaderApplicationFactory">
      <Settings>
        <add Name="Enabled" Value="auto"/>
        <add Name="MaxInstances" Value="5"/>
      </Settings>
    </AppFactory>

    <AppFactory Name="Microsoft Excel"
      Type="PEERNET.PNDocConv.Applications.PNExcelApplicationFactory"
      Assembly="PNExcelApplicationFactory">
      <Settings>
        <add Name="Enabled" Value="auto"/>
        <add Name="MaxInstances" Value="2"/>
      </Settings>
    </AppFactory>

    ...

  </Factories>
  <Settings>
    <!-- Global factory settings -->
    <add Name="MaxInstances" Value="auto"/>
    <add Name="RecycleThreshold" Value="0"/>
    <add Name="ReadyThreshold" Value="5000" />
    <add Name="AppInitializationThreshold" Value="30000" />
    <add Name="RecycleVirtualSizeThreshold" Value="1400000"/>
    <add Name="RecycleGDIandUserHandleCountThreshold" Value="8000"/>
    <add Name="RecycleProcessHandleCountThreshold" Value="2000"/>
    <add Name="AppTeardownIdleTimeout" Value="3600000"/>

  </Settings>
</AppFactories>
```


Changing the Application Recycle Count and Threshold

You can also change how often an application in the pool is recycled. Recycling an application keeps long running applications from slowly consuming resources.

An application is recycled for three reasons:

- The *RecycleThreshold* for the number of document processed by this instance has been met.
- The *RecycleVirtualSizeThreshold* value for the virtual size of the running application has been exceeded.
- If a file fails to convert Document Conversion Service will automatically recycle the application. This cannot be changed.

The sample below shows a possible configuration for the following:

- The Microsoft Word converter will be recycled after 100 documents or if the application's virtual size exceeds the 1.7GB limit set in the global settings section at the end of the `<AppFactories>` section
- The Adobe Acrobat Reader converter has a custom *RecycleVirtualSizeThreshold* of 1GB but it does not have a setting for *RecycleThreshold*. It will default to the *RecycleThreshold* value of 200 in the global settings section at the end of the `<AppFactories>` section.

1. In the `<AppFactories>` section look for the `<AppFactory>` section for the converter whose recycle count or size threshold you want to adjust.
 - a. Set the `RecycleThreshold` to the desired value. Take care when adjusting this value too low as recycling an application takes time; recycling too often will decrease the throughput and the Document Conversion Service will spend too much time stopping and restarting the application.
 - b. Set the `RecycleVirtualSizeThreshold` value to the desired size. This value is specified in 1024KB blocks (1=1024KB).
2. You can change the global `RecycleThreshold` and `RecycleVirtualSizeThreshold` for all converters in the `<Settings>` section at the bottom of the `<AppFactories>` section. These values will be used if they are not specified in the converters' `<AppFactory>` section.



AppFactories Configuration Section

```
<AppFactories>
  <Factories>

    <AppFactory Name="Microsoft Word"
      Type="PEERNET.PNDocConv.Applications.PNWordApplicationFactory"
      Assembly="PNWordApplicationFactory">
      <Settings>
        <add Name="Enabled" Value="auto"/>
        <add Name="MaxInstances" Value="3"/>
        <add Name="RecycleThreshold" Value="100"/>
      </Settings>
    </AppFactory>

    <AppFactory Name="Adobe Acrobat Reader"
      Type="PEERNET.PNDocConv.Applications.PNAcrobatReaderApplicationFactory"
      Assembly="PNAcrobatReaderApplicationFactory">
      <Settings>
        <add Name="Enabled" Value="auto"/>
        <add Name="MaxInstances" Value="3"/>
        <add Name="RecycleVirtualSizeThreshold" Value="100000"/>
      </Settings>
    </AppFactory>

    ...

  </Factories>
  <Settings>
    <!-- Global factory settings -->
    <add Name="MaxInstances" Value="3"/>
    <add Name="RecycleThreshold" Value="200"/>
    <add Name="RecycleVirtualSizeThreshold" Value="1700000"/>
  </Settings>
</AppFactories>
```

Restoring the Configuration File

A backup copy of the original configuration file is stored in the following location for easy recovery.

Configuration file location:

C:\Program Files\Document Conversion Service 3.0\Core\Backup\PNJobItemProcessor.exe.config

Enabling and Disabling Converters

The application factory for each converter controls if that converter will be loaded or not through its **Enabled** setting. The Enabled setting can be one of three values:

Enabled	Result
auto	When set to <i>auto</i> , Document Conversion Service will check the converter's requirements, and load it only if the requirements are met. In most cases the requirements are usually the native application the converter uses to help do the conversion.
true	Document Conversion Service will always try to load the converter. If the converter requires a separate application and that application is not installed this setting will cause Document Conversion Service to fail its initialization and the service will not start.
false	The converter is not loaded.

Opening the Configuration File

Go to **DCS Dashboard** - DCS-Settings - Edit DCS Configuration to edit the configuration file using a visual GUI.

You can also go to Start - All Programs - PEERNET Document Conversion Service 3.0 - Edit DCS Configuration File to edit the configuration file using the DCS Editor.

The configuration file can also be opened in any XML editor and can be found here:

Configuration file location:

C:\Program Files\Document Conversion Service 3.0\Core\PNJobItemProcessor.exe.config

Enabling or Disabling the Converters through the Application Factory

The sample below shows how to disable the converter for Microsoft Word.

1. In the `<AppFactories>` section, look for the `<AppFactory>` section for the converter you want to disable.
2. Set the Enabled value to *false* to disable the converter.
 - a. Set this value to *true* to always load the converter or *auto* to have Document Conversion Service automatically detect if the converter can be used.



AppFactories Configuration Section

```
<AppFactories>
  <Factories>

    <AppFactory Name="Microsoft Word"
      Type="PEERNET.PNDocConv.Applications.PNWordApplicationFactory"
      Assembly="PNWordApplicationFactory">
      <Settings>
        <add Name="Enabled" Value="false"/>
        <add Name="MaxInstances" Value="5"/>
        <add Name="RecycleThreshold" Value="1000"/>
      </Settings>
    </AppFactory>

    <AppFactory Name="Adobe Acrobat Reader"
      Type="PEERNET.PNDocConv.Applications.PNAcrobatReaderApplicationFactory"
      Assembly="PNAcrobatReaderApplicationFactory">
      <Settings>
        <add Name="Enabled" Value="auto"/>
        <add Name="MaxInstances" Value="2"/>
      </Settings>
    </AppFactory>

    ...

  </Factories>
  <Settings>
    <!-- Global factory settings -->
    <add Name="MaxInstances" Value="5"/>
    <add Name="RecycleThreshold" Value="100"/>
  </Settings>
</AppFactories>
```

Restoring the Configuration File

A backup copy of the original configuration file is stored in the following location for easy recovery.

Configuration file location:

C:\Program Files\Document Conversion Service 3.0\Core\Backup\PNJobItemProcessor.exe.config

Custom Converter Behaviour

When printing documents, some converters may require more or less time than others at certain stages of the printing process. For instance, some applications may need more time to spool the document to the printer than others, or a particular converter is handling files that are consistently larger and need more time to complete. These settings are normally set globally in the [Document Conversion Service Printer Pool](#) section but can also be overridden on a per-converter basis if needed.

The converter factory for each converter uses the following settings to control the printing timeouts:

Setting Name	Value
COMRetryLaterMaxTimeout	Sets a time limit, in milliseconds, on how long to retry Office automation COM calls when automating returns the error <code>RPC_E_SERVERCALL_RETRYLATER</code> . Currently used only by Excel and should never need to be modified. The default value is 2000ms.
PrintSessionWaitTimeout	How long the converter factory will wait to get access to a printing session.
PrintSessionFirstJobTimeout	This setting is applied to the printing session used by the converter factory and determines how long the printing session will wait for a job to start spooling in the printer queue after a document is printed before releasing the printing session back into the printer pool.
PrintSessionAvailableTimeout	This setting is applied to the printing session used by the converter factory and determines how long to wait between jobs entering the queue before releasing the printing session back into the printer pool.
PrintSessionWaitOnSpoolingTimeout	How long the converter factory will wait for each job to start spooling in the printer queue.
PrintSessionWaitOnCompleteTimeout	The maximum amount of time the converter factory will wait for the document to finish printing in the printer queue.
UsesPrintingProtocol	This is true for all converter factories that print to the Document Conversion Service to convert the document, false for any converter factories that do not use the printer. In most cases this setting never needs to be modified.
MaxRetryAttempts	Controls the number of times to retry converting a document if it was not successful on printing. Minimum value is 0, meaning we will not retry, and the maximum number of retries is 5. The default is 2. Setting this value in the converter settings will override this setting if set in the The Application Pool or in the Document Conversion Service Printer Pool section.

Setting Name	Value
MaxSpooledPagesAllowed	<p>Sets the maximum number of pages that are allowed to be printed/spooled. The default value is 0, meaning there is no limit. If a document exceeds this count, it enters an error state and no file is created. To limit how many pages to convert see the <i>PageRange</i> setting in General Converter Options.</p> <p>Setting this value in the converter settings will override this setting if set in the The Application Pool or in the Document Conversion Service Printer Pool section.</p> <p>This option can also be overridden on a per document basis using profiles as described in Creating and Customizing Profiles.</p>
ZeroByteFiles	<p>Determines if files with a size of zero (0 bytes) are skipped or failed when processed. When set to Fail, an error is produced. When set to Skip, the file is skipped and a message is produced instead of an error. Default behaviour is to fail the file.</p>

These variables control the maximum amount of time to wait on the open and close calls to the converter to ensure the conversion threads are not blocked by the underlying application. These values are entered in microseconds (1 second = 1000 microseconds). If not specified the default value is 60000ms, and can be no smaller than 20000ms.

DocumentOpenTimeout	The maximum amount of time to wait for the converter to open the document.
DocumentConvertTimeout	The maximum amount of time to wait for the converter to convert the document
DocumentCloseTimeout	The maximum amount of time to wait for the converter to close the open document.
DocumentCloseAllTimeout	The maximum amount of time to wait for the converter to close all open documents.
ApplicationCloseTimeout	The maximum amount of time to wait for the application to close.

Changing the Converter Timeouts

In most cases these timeouts should not have to be changed from the defaults provided.

Opening the Configuration File

Go to **DCS Dashboard** - DCS-Settings - Edit DCS Configuration to edit the configuration file using a visual GUI.

You can also go to Start - All Programs - PEERNET Document Conversion Service 3.0 - Edit DCS Configuration File to edit the configuration file using the DCS Editor.

The configuration file can also be opened in any XML editor and can be found here:

Configuration file location:

C:\Program Files\Document Conversion Service 3.0\Core\PNJobItemProcessor.exe.config

Configuring Converter Factories

The sample below shows both the Microsoft Word and Adobe Acrobat Reader converter factory definitions. The Adobe Acrobat Reader converter shown is overriding the `PrintSessionWaitOnSpoolingTimeout` with a timeout value of 10000ms (10 seconds). Both converters will use the `UsesPrintingProtocol` setting of true as defined in the global settings section as they

1. In the `<PluginFactories>` section, look for the `<PluginFactory>` section for the converter whose timeouts you want to adjust.
2. Set new timeouts as desired.



PluginFactories Configuration Section

```
<PluginFactories>
  <Factories>
    <PluginFactory Name="Microsoft Word"
      Type="PEERNET.PNDocConv.Converters.PNWordConverter"
      Assembly="PNWordConverter" AppFactory="Microsoft Word">
      <Settings>
        <!-- Add custom converter settings here -->
      </Settings>
    </PluginFactory>

    <PluginFactory Name="Adobe Acrobat Reader"
      Type="PEERNET.PNDocConv.Converters.PNAcrobatReaderConverter"
      Assembly="PNAcrobatReaderConverter"
      AppFactory="Adobe Acrobat Reader">
      <Settings>
        <!-- Add any custom converter settings here -->
        <add Name="PrintSessionWaitOnSpoolingTimeout" Value="10000"/>
      </Settings>
    </PluginFactory>
  </Factories>
  <Settings>
    <!-- Global converter factory settings-->
    <add Name="UsesPrintingProtocol" Value="true"/>
  </Settings>
</PluginFactories>
```

Restoring the Configuration File

A backup copy of the original configuration file is stored in the following location for easy recovery.

Configuration file location:

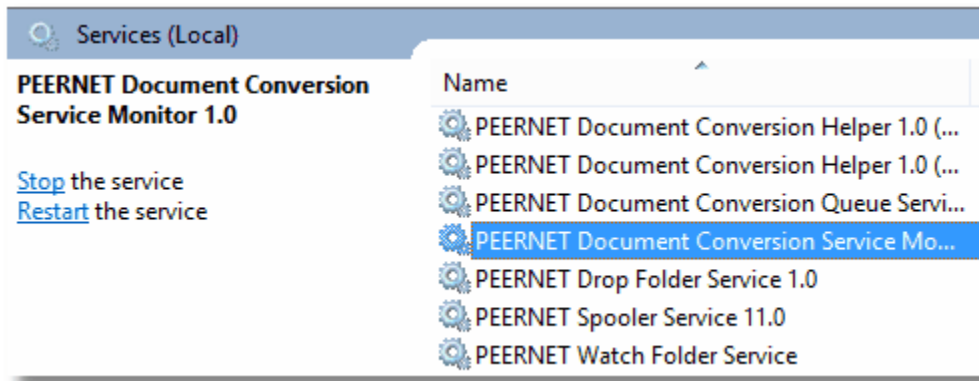
C:\Program Files\Document Conversion Service 3.0\Core\Backup\PNJobItemProcessor.exe.config

Changing Document Conversion Service's Startup Mode

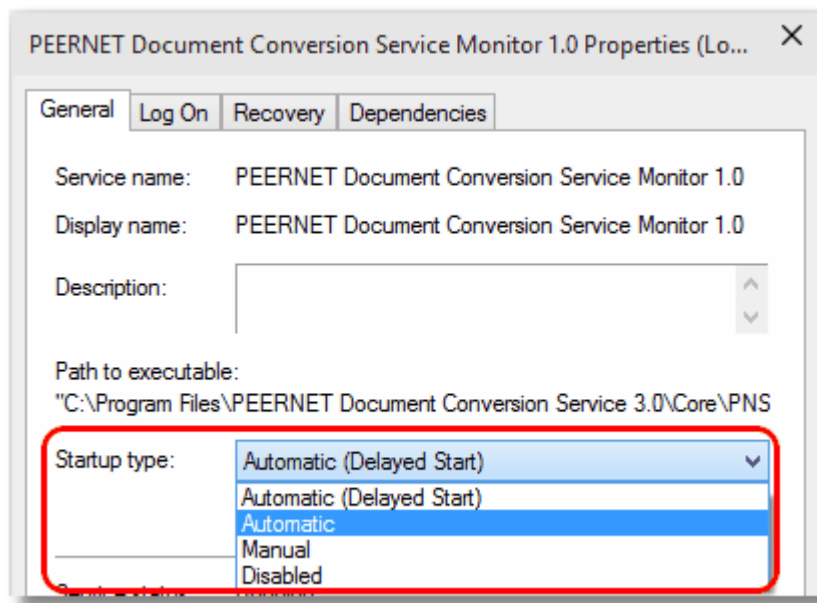
Document Conversion Service is managed by the PEERNET Document Conversion Service Monitor 1.0 service. This monitoring service is installed as an automatic service with a delayed start. This means that each time the computer is started, the monitoring service will start Document Conversion Service after a short delay. While not recommended, this can be changed through the service's control panel applet.

Changing the Service's Start Mode

1. Go to Start - Control Panel - System and Security - Administrative Tools - Services (or type "Services" into the search field on the **Start** menu).
2. In the Services control panel applet select the PEERNET Document Conversion Service Monitor 1.0 service. The service can be running, but any changes will not take place until the service is restarted.



3. Double-click the service in the list to open the Properties dialog.
4. On the General tab change the Startup type to the desired mode.



Appendix

- [General Application Settings](#) - all settings used to define the application configuration, such as number of documents converted in parallel.
- [Application Factory Settings](#) - lists all application factory settings
- [Converter Factory Settings](#) - lists all converter factory settings.

General Application Settings

These options control the number of documents that can be converted concurrently. This is limited by your license model and your available system resources such as CPU and memory.

Setting Name	Value
NumberOfDocumentsInParallel	Number of documents that can be converted at the same time. Set to "auto" to use the system resources to automatically determine an appropriate value.
NumberOfPrinters	Controls the size of the Document Conversion Service printer pool. This value should match <i>NumberOfDocumentsInParallel</i> for best performance.

These variables control the overall behavior of the Document Conversion Service.

Setting Name	Value
RunSelfHealForCoreServices	Detects proper installation of required components and will attempt to self-heal if any components are found missing. This check is always performed by default. We do not recommend disabling this check.
RunSelfHealForOtherServices	Optional detection and self-heal of secondary components; detects proper installation and will attempt to self-heal if any components are found missing. This check is performed by default.
ThreadInitBeforeSignalRunningState	How long to wait for the converter factory threads to initialize and be ready to process documents.
MaxWaitForProcessingTimeoutInMinutes	The maximum amount of time, in minutes, to wait for a document to signal that it is being converted. The minimum timeout is 5 minutes, the default is 30 minutes.
SessionWaitForAllJobsCompletedTimeout	The maximum amount of time to wait for all documents to finish printing when shutting Document Conversion Service down.
WaitForSrv10ToClose	The Document Conversion Service uses the PNSrv10 component and cannot close until that component has exited first. The default amount of time to wait is 60 seconds, this component normally exits in just over 30 seconds.
RestartServiceInHours	When set to the default value of 0, the Document Conversion Service is never restarted. If desired, the service can be set to be automatically restarted anywhere from

Setting Name	Value
	every hour up to every seven days (168 hours).

These variables control the maximum amount of time to wait on the open and close calls to the converter to ensure the conversion threads are not blocked by the underlying application. These values are entered in microseconds (1 second = 1000 microseconds). If not specified the default value is 60000ms, and can be no smaller than 20000ms. Any marked with (*) can be overridden by the converter factory if needed (see [Converter Factory Settings](#)).

DocumentOpenTimeout*	The maximum amount of time to wait for the converter to open the document.
DocumentConvert*	The maximum amount of time to wait for the converter to convert the document
DocumentCloseTimeout*	The maximum amount of time to wait for the converter to close the open document.
DocumentCloseAllTimeout*	The maximum amount of time to wait for the converter to close all open documents.
ApplicationCloseTimeout*	The maximum amount of time to wait for the application to close.

The following variables control the behavior of the Document Conversion Service printer pool such as how long to wait for a printer to become available. Any marked with (*) can be overridden by the converter factory if needed (see [Converter Factory Settings](#)).

Setting Name	Value
PrintSessionWaitTimeout*	How long the converter will wait to get access to a printing session. This value is entered in microseconds (1 second = 1000 microseconds).
PrintSessionFirstJobTimeout*	This setting is applied to the printing session used by the converter and determines how long the printing session will wait for a job to start spooling in the printer queue before releasing the printing session back into the printer pool. This value is entered in microseconds (1 second = 1000 microseconds).
PrintSessionAvailableTimeout*	This setting is applied to the printing session used by the converter and determines how long to wait between jobs entering the queue before releasing the printing session back into the printer pool. This value is entered in microseconds (1 second = 1000 microseconds).

Setting Name	Value
PrintSessionWaitOnSpoolingTimeout*	How long the converter will wait for each job to start spooling in the printer queue. This value is entered in microseconds (1 second = 1000 microseconds).
PrintSessionWaitOnCompleteTimeout*	This is NOT the total amount of time for the document to convert, it is the amount of idle time used to determine when to cancel a document being created. If the converter does not see any progress (pages being converted) in this amount of time the document is canceled.
SessionWaitForAllJobsCompletedTimeout	The maximum amount of time to wait for all documents to finish printing when shutting Document Conversion Service down. This value is entered in microseconds (1 second = 1000 microseconds).
MaxRetryAttempts*	Controls the number of times to retry converting a document if it was not successful on printing. Minimum value is 0, meaning we will not retry, and the maximum number of retries is 5. The default is 2.
MaxSpooledPagesAllowed*	<p>Sets the maximum number of pages that are allowed to be printed/spooled. The default value when this is not provided is 0, meaning there is no limit. The install sets this value to 20,000 pages to prevent extremely large documents for stalling the converting process. If a document exceeds this count, it enters an error state and no file is created. To limit how many pages to convert see the PageRange setting in General Converter Options.</p> <p>This option can also be overridden on a per document basis using profiles as described in Creating and Customizing Profiles.</p>
ZeroByteFiles*	Determines if files with a size of zero (0 bytes) are skipped or failed when processed. When set to Fail , an error is produced. When set to Skip , the file is skipped and a message is produced instead of an error. Default behaviour is to fail the file.

* These settings can be overridden by the individual settings for the converters in their `<PluginFactory>` section if needed.

Application Factory Settings

These settings can be used in both the application factory settings collection and in the global application factory settings collection. Settings in the application factory will override the global default settings.

Setting	Value
Enabled	Set to <i>auto</i> to automatically try and start the converter, <i>true</i> to enable the converter and make it required, and <i>false</i> to disable it.
MaxInstances	<p>The maximum size of the application pool for this converter. For best performance leave this set to "auto" to have the size of the application pool tailored to the capabilities of your computer. If this setting is not provided, or set to 0 or less, a single application instance will be created.</p> <p>The application pool is dynamic and will start with a single application in the pool with new applications added as needed. If an application in the pool is idle, meaning it has not processed any conversions, for a certain amount of time it is removed from the pool. This is controlled by the AppTeardownIdleTimeout setting below.</p>
MaxRetryAttempts	<p>Controls the number of times to retry converting a document if it was not successful on printing. Minimum value is 0, meaning we will not retry, and the maximum number of retries is 5. The default is 2.</p> <p>Setting this value in the application pool level will override this setting in the Document Conversion Service Printer Pool section.</p>
MaxSpooledPagesAllowed	<p>Sets the maximum number of pages that are allowed to be printed/spooled. The default value value is 0, meaning there is no limit. If a document exceeds this count, it enters an error state and no file is created. To limit how many pages to convert see the <i>PageRange</i> setting in General Converter Options.</p> <p>Setting this value in the application factory level will override this setting in the General Application Settings section.</p> <p>This option can also be overridden on a per document basis using profiles as described in Creating and Customizing Profiles.</p>
RecycleThreshold	Maximum number of documents each application can process before it is recycled and a new instance started to replace it.

Setting	Value
	This is set to 0 by default, meaning the application will not recycle.
ReadyThreshold	The maximum length of time to wait after the application has been initialized before Document Conversion Service initiates communication with the application. This value may need to be increased for machines running high volume with many other applications running.
AppInitializationThreshold	Some applications need more time than others to complete their initialization. Enter in the length of time, in microseconds, to wait for the application to initialize.
AppTeardownIdleTimeout	<p>The amount of time, in milliseconds, to wait before an idle application is closed and removed from the application pool. An idle application is one that has not processed any conversions in the specified time period. These idle applications are removed from the pool to free up resources. They are added back in on demand as needed.</p> <p>This is set to 3,600,000 milliseconds (1 hour) by default in the global <code><AppFactories></code> section.</p> <p>If this is set to 0, the applications in the pool will start dynamically but will not be dynamically removed from the pool. They will only be removed if they are recycled due to conversion failure or the settings for RecycleThreshold, RecycleVirtualSizeThreshold, RecycleGDIandUserHandleCountThreshold, and RecycleProcessHandleCountThreshold.</p>
AppSynchronousPrintModeCheckPrintQueue	Some applications print synchronously, meaning control doesn't return to Document Conversion Service until the file has been sent to the printer. In some cases we need to check the printer queue to see if the print action actually submitted a job. If it has not we fail the conversion gracefully. This setting is false for most applications.
RecycleVirtualSizeThreshold	The size (in 1024KB blocks) at which to recycle the application. For example, 1400000 is 1.4GB meaning the application will be recycled when its virtual size is larger than 1.4 GB. Is it important to keep this value below the 2GB virtual size for 32-bit applications. While you can disable the application recycling based on Virtual Size by setting this to 0 or removing the value completely, we do not recommend this.
RecycleGDIandUserHandleCountThreshold	The maximum number of combined user and GDI handles allowed for each application instance. When this number of user and GDI handles

Setting	Value
	exceed this threshold the application will be recycled and a new instance started to replace it in the application pool. If this value is not set, or set to zero, the maximum number of combined handles is 8000.
RecycleProcessHandleCountThreshold	The maximum number of process handles allowed for each application instance. When this number exceeds this threshold the application will be recycled and a new instance started to replace it in the application pool. If this value is not set, or set to zero, the maximum number of combined handles is 2000.
ZeroByteFiles	Determines if files with a size of zero (0 bytes) are skipped or failed when processed. When set to Fail , an error is produced. When set to Skip , the file is skipped and a message is produced instead of an error. Default behaviour is to fail the file.

These settings are used for development purposes only. They should not be used in a production system.

Setting Name	Value
RunVisible	<p>This flag should be <i>false</i> or removed completely on a production system. Not recommended when Starting and Stopping the Service.</p> <p>Used for development purposes only. If the application can be run visible, and not all can be, it is shown on screen.</p>

Converter Factory Settings

These settings can be used in both the converter factory settings collection and in the global converter factory settings collection. Settings in the converter factory will override the global default settings.

Setting Name	Value
UsesPrintingProtocol	This is true for all converter factories that print to the Document Conversion Service to convert the document, false for any converter factories that do not use the printer. In most cases this setting never needs to be modified.
COMRetryLaterMaxTimeout	Sets a time limit, in milliseconds, on how long to retry Office automation COM calls when automating returns the error <code>RPC_E_SERVERCALL_RETRYLATER</code> . Currently used only by Excel and should never need to be modified. The default value is 2000ms.

These settings are normally only set in the [General Application Settings](#) section of the application configuration file but can be overridden as needed in the individual converter factory settings.

Setting Name	Value
PrintSessionWaitTimeout	How long the converter will wait to get access to a printing session.
PrintSessionFirstJobTimeout	This setting is applied to the printing session used by the converter and determines how long the printing session will wait for a job to start spooling in the printer queue after a document is printed before releasing the printing session back into the printer pool
PrintSessionAvailableTimeout	This setting is applied to the printing session used by the converter and determines how long to wait between jobs entering the queue before releasing the printing session back into the printer pool.
PrintSessionWaitOnSpoolingTimeout	How long the converter will wait for each job to start spooling in the printer queue.
PrintSessionWaitOnCompleteTimeout	This is NOT the total amount of time for the document to convert, it is the amount of idle time used to determine when to cancel a document being created. If the converter does not see any progress (pages being converted) in this amount of time the document is canceled.
MaxRetryAttempts	Controls the number of times to retry converting a document if it was not successful on printing. Minimum value is 0, meaning we will not retry, and the maximum number of retries is 5. The default is 2. Setting this value in the converter settings will override this setting if set in the The Application Pool or in the Document Conversion Service Printer Pool section.
MaxSpooledPagesAllowed	Sets the maximum number of pages that are allowed to be printed/spooled. The default value value is 0,

Setting Name	Value
	<p>meaning there is no limit. If a document exceeds this count, it enters an error state and no file is created. To limit how many pages to convert see the <i>PageRange</i> setting in General Converter Options.</p> <p>Setting this value in the converter settings will override this setting if it is set in either Application Factory Settings or in the General Application Settings section.</p> <p>This option can also be overridden on a per document basis using profiles as described in Creating and Customizing Profiles.</p>
ZeroByteFiles	Determines if files with a size of zero (0 bytes) are skipped or failed when processed. When set to Fail , an error is produced. When set to Skip , the file is skipped and a message is produced instead of an error. Default behaviour is to fail the file.

These variables control the maximum amount of time to wait on the open and close calls to the converter to ensure the conversion threads do not blocked by the underlying application. These values is entered in microseconds (1 second = 1000 microseconds). If not specified the default value is 60000ms, and can be no smaller than 20000ms.

DocumentOpenTimeout	The maximum amount of time to wait for the converter to open the document.
DocumentConvertTimeout*	The maximum amount of time to wait for the converter to convert the document.
DocumentCloseTimeout	The maximum amount of time to wait for the converter to close the open document.
DocumentCloseAllTimeout	The maximum amount of time to wait for the converter to close all open documents.
ApplicationCloseTimeout	The maximum amount of time to wait for the application to close.