Version
3.0

PEERNET.ConvertUtility

Developer's Guide

PEERNET Inc.

Updated: 1/2/2024

Table of Contents

Introduction

Welcome to the PEERNET.ConvertUtility help. The table below outlines the different sections of this help manual.

Topic	Description
About PEERNET.ConvertUtility	Information about product releases, requirements and support are included in this section.
Installation and Deployment	This section covers installation of the PEERNET.ConvertUtility and what files need to be redistributed with your application.
Getting Started	Step-by step tutorials in this section explain how to call the PEERNET.ConvertUtility from your own code, and how to use the returned information to find the converted files, information messages or error messages.
Working With PEERNET.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 PEERNET.ConvertUtility files needed when deploying applications.
PEERNET.ConvertUtility Namespace	This reference section contains detailed descriptions of all classes in the PEERNET.ConvertUtility library.

About PEERNET.ConvertUtility

This section contains information about product releases and system requirements.

- Legal Notices
- Requirements

Legal Notices

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

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 operating systems are supported.

- Microsoft® Windows Server 2022
- Microsoft® Windows 11
- Microsoft® Windows Server 2019
- Microsoft® Windows Server 2016
- Microsoft® Windows Server 2012 R2
- Microsoft® Windows Server 2012
- Microsoft® Windows Server 2008 R2
- Microsoft® Windows Server 2008
- Microsoft® Windows 10 (up to version 1809)
- Microsoft® Windows 8, 8.1
- Microsoft® Windows 7
- Microsoft® Windows Vista
- Microsoft® Windows XP SP3

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

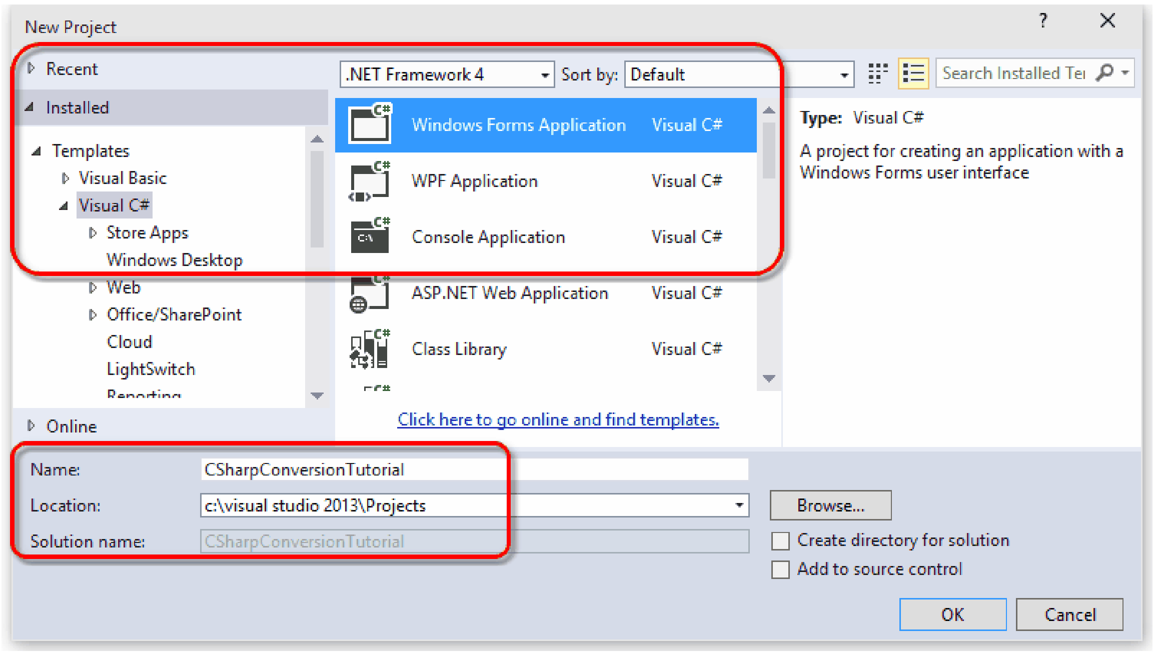
This tutorial will show you how to add file conversion to your C#.NET application using PEERNET.ConvertUtility. The tutorial creates a simple C# Windows forms application with a single button that converts a file when pressed and displays the results in a list box when finished. It also assumes that Document Conversion Service is installed on your local computer.

- Step 1: Creating a Simple Application
- Step 2: Adding the 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 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.



Type: Visual C#

A project for creating an application with a Windows Forms user interface

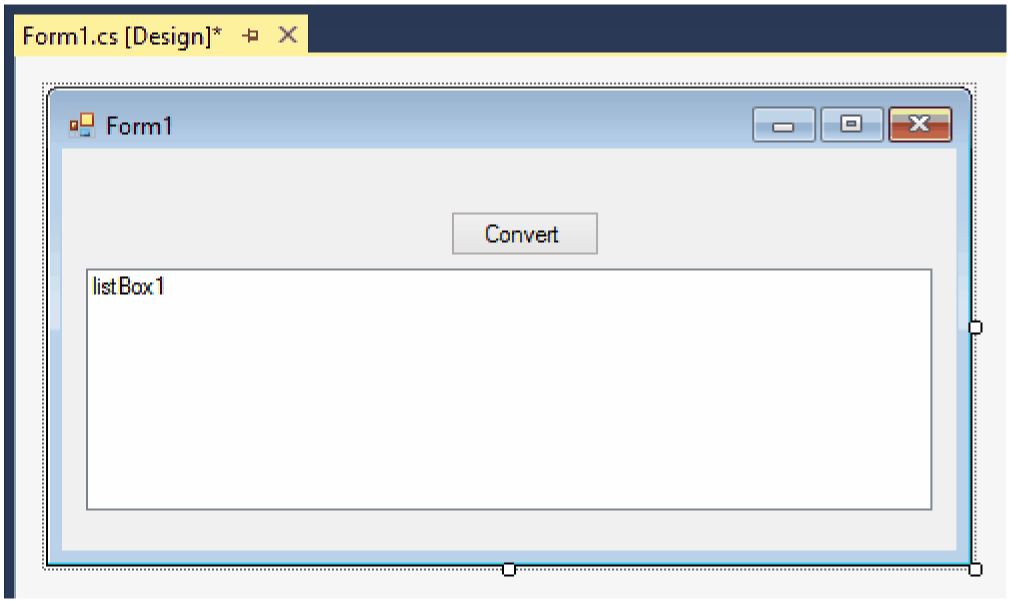
Name: CSharpConversionTutorial

Location: c:\visual studio 2013\Projects

Solution name: CSharpConversionTutorial

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.

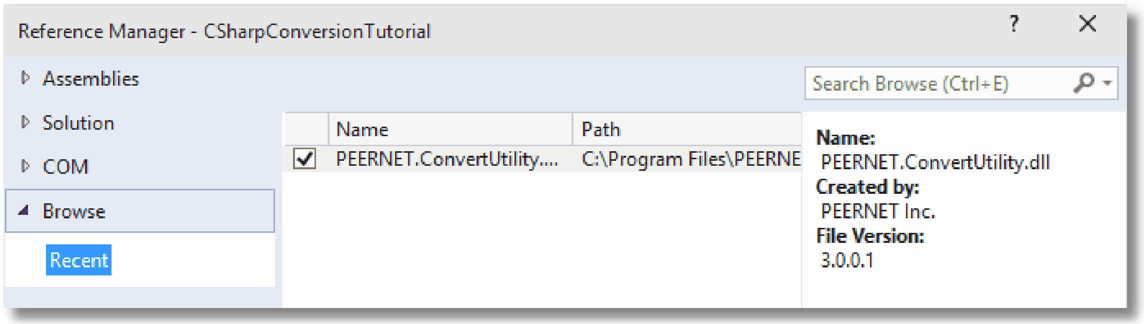
c. Change the width of both the form and the listbox to be able to display more information in the listbox.



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.



	Name	Path
✓	PEERNET.ConvertUtility....	C:\Program Files\PEERNE

Name:
PEERNET.ConvertUtility.dll
Created by:
PEERNET Inc.
File Version:
3.0.0.1

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.

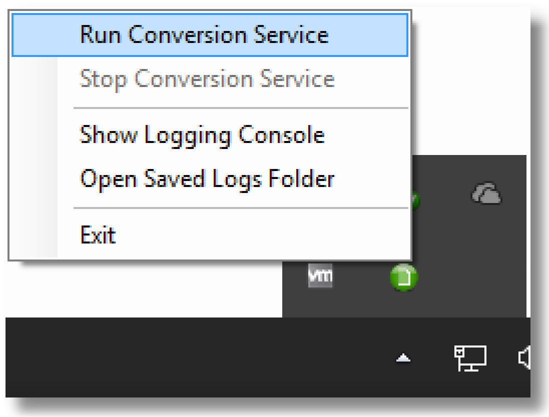
1. Add the following method into *Form1.cs*. This method will display the name of the file we tried to convert, and then will list the new file that was created. If the conversion failed, the error messages are displayed instead.

```
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 occured 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 where 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.");
    }
}
```

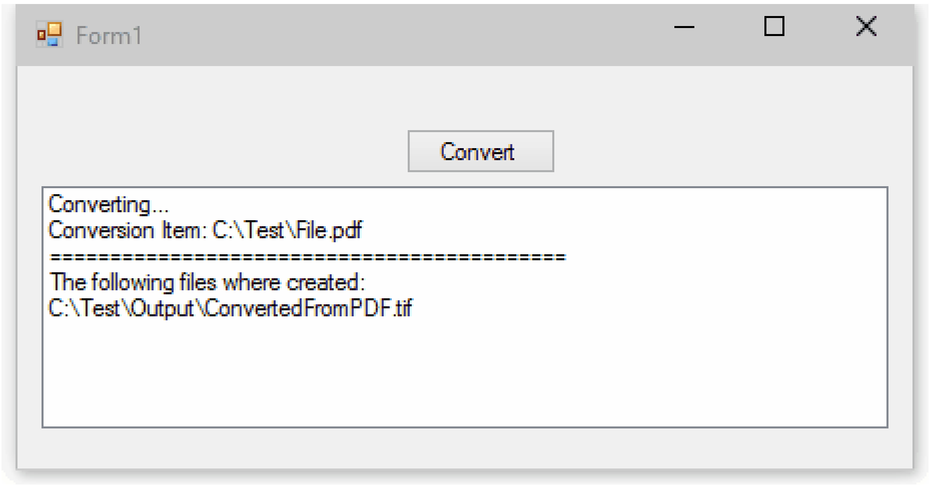
5. Testing the Application

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.



```
Converting...
Conversion Item: C:\Test\File.pdf
============================================
The following files where created:
C:\Test\Output\ConvertedFromPDF.tif
```

Visual Basic .NET Tutorial

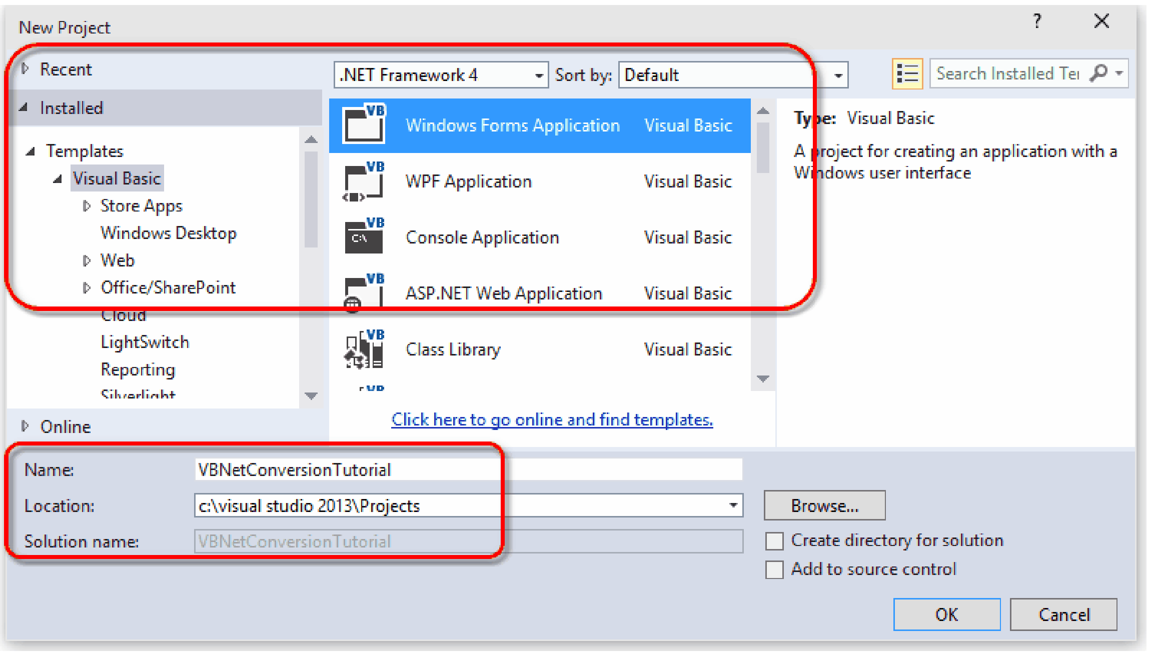
This tutorial will show you how to add file conversion to your Visual Basic .NET application using PEERNET.ConvertUtility. The tutorial creates a simple Visual Basic Windows forms application with a single button that converts a file when pressed and displays the results in a list box when finished. 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.


New Project
Recent
Installed
Templates
Visual Basic
Store Apps
Windows Desktop
Web
Office/SharePoint
Cloud
LightSwitch
Reporting
Silverlight
Online
Sort by:
Default
Search Installed Templates
Windows Forms Application Visual Basic
WPF Application Visual Basic
Console Application Visual Basic
ASP.NET Web Application Visual Basic
Class Library Visual Basic
Click here to go online and find templates.
Browse...
Create directory for solution
Add to source control
OK
Cancel

.NET Framework 4

Type: Visual Basic

A project for creating an application with a Windows user interface

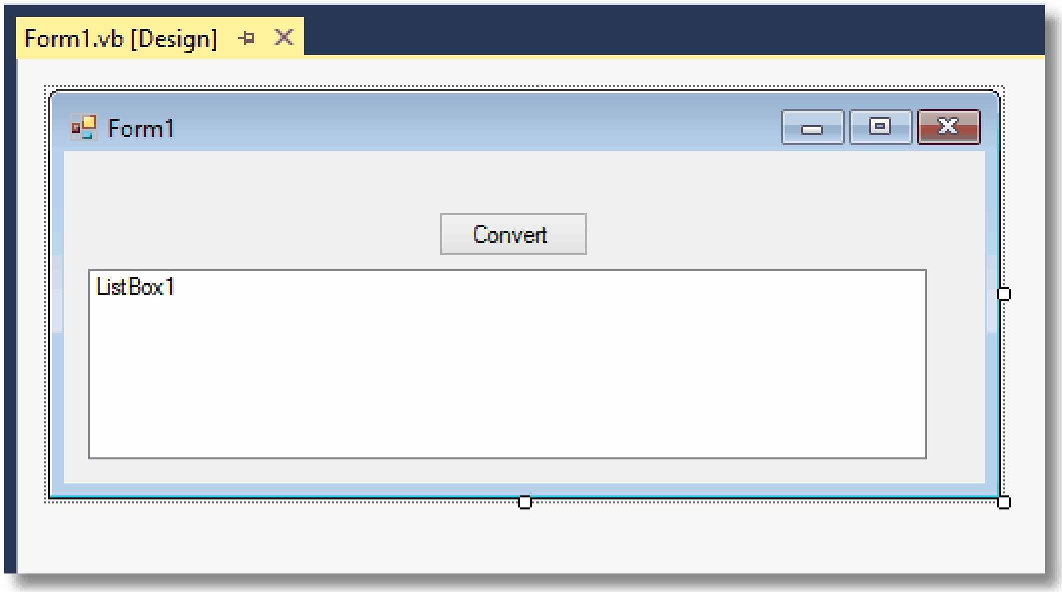
Name: VBNetConversionTutorial

Location: c:\visual studio 2013\Projects

Solution name: VBNetConversionTutorial

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.

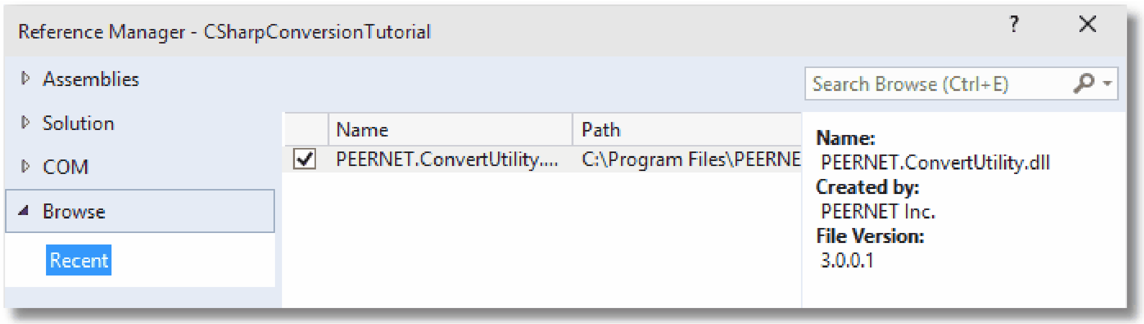
c. Change the width of both the form and the listbox to be able to display more information in the listbox.



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.



	Name	Path
✓	PEERNET.ConvertUtility....	C:\Program Files\PEERNE

Name:
PEERNET.ConvertUtility.dll
Created by:
PEERNET Inc.
File Version:
3.0.0.1

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.

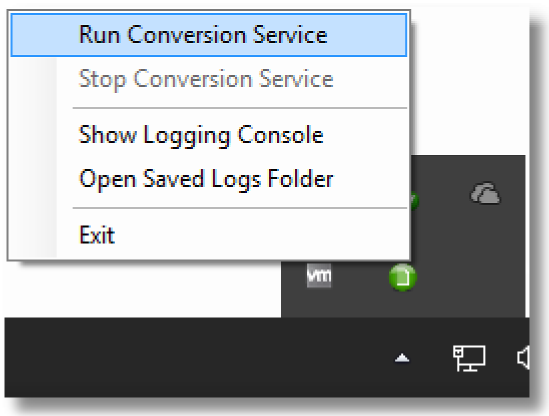
1. Add the following method into *Form1.vb*. This method will display the name of the file we tried to convert, and then will list the new file that was created. If the conversion failed, the error messages are displayed instead.

```
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 lis
           ' 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 occured 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 where 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
```

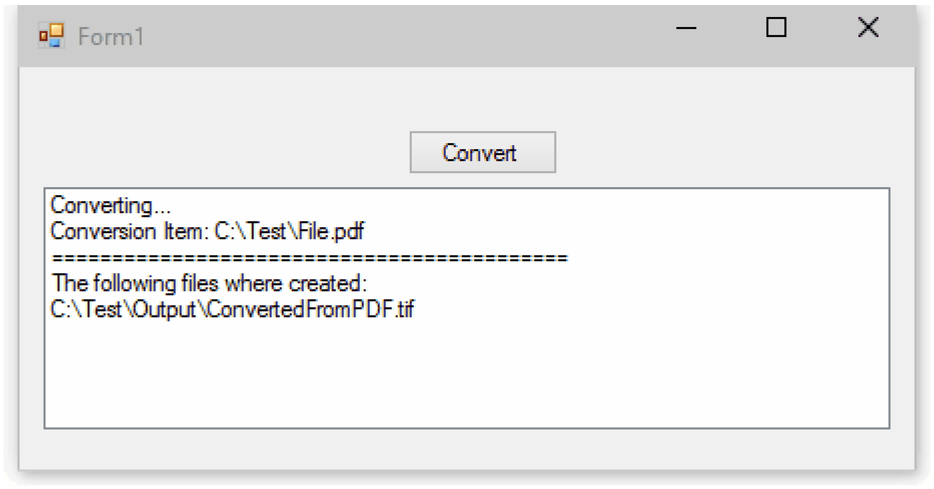
5. Testing the Application

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.



```
Converting...
Conversion Item: C:\Test\File.pdf
=============================================
The following files where created:
C:\Test\Output\ConvertedFromPDF.tif
```

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 PEERNET.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 PEERNET.ConvertUtility you should start with the tutorials in the Getting Started section.

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 *ExcudeFilter* 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 *Draft_*.

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 wil 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.

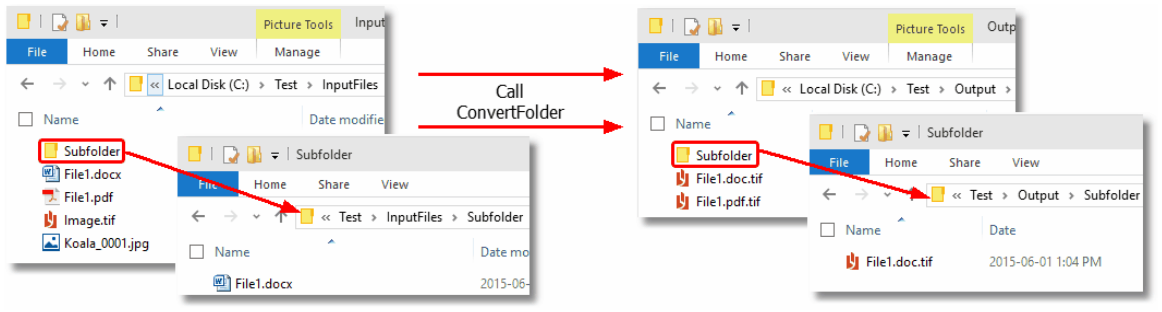
- **Ascending** - sorted the files from low to high: 0-9, A-Z.
- **Descending** - sorts the files from high to low: Z-A, 9-0.

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, // extensison 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.Asccending); // 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.



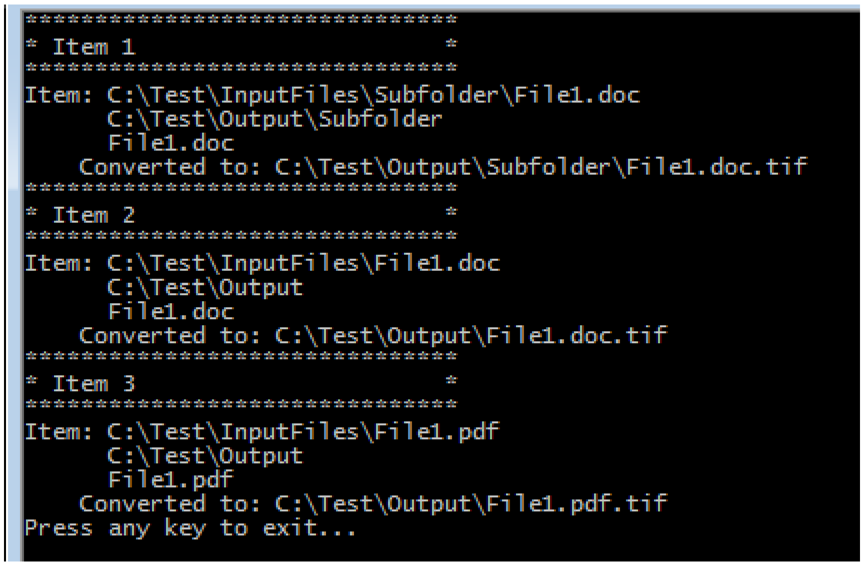
Reading the Results Collection

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.

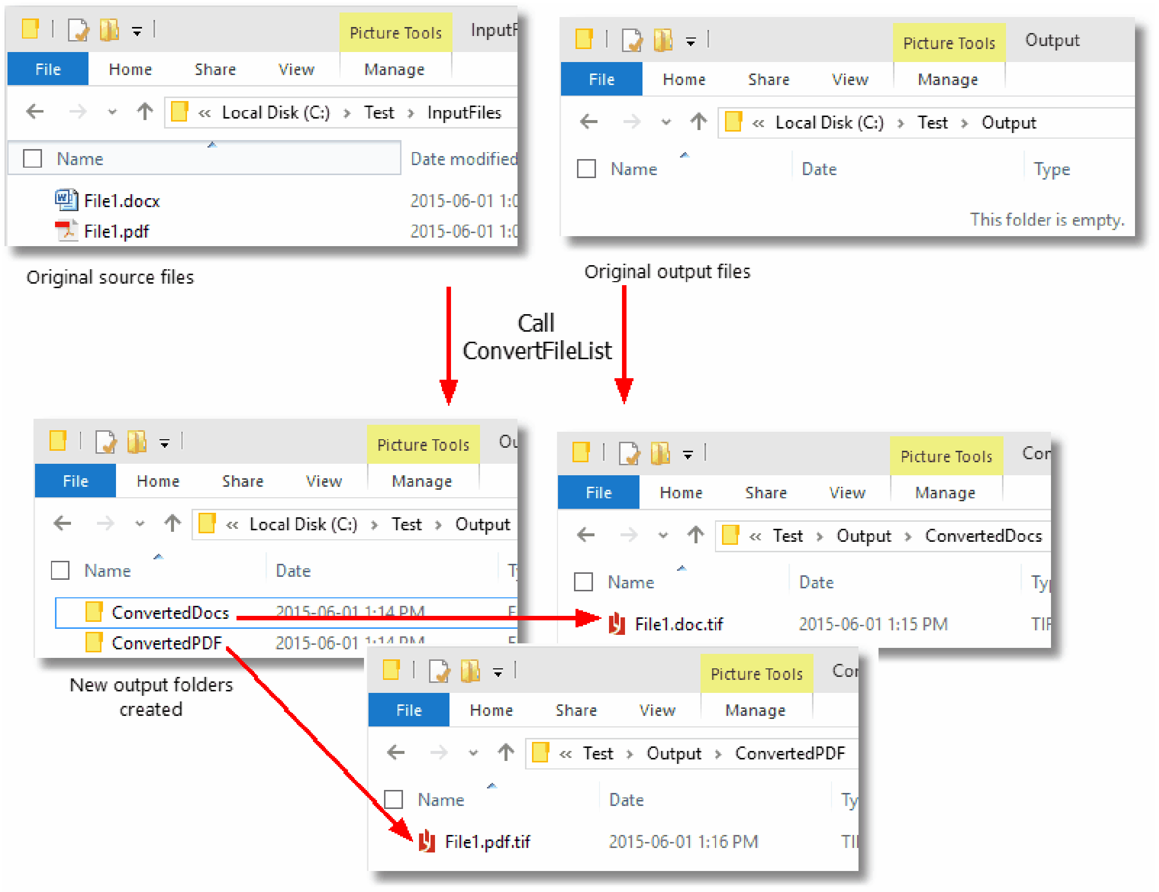
The first file in the list will use only the conversion settings from the profile *TIFF 200dpi OptimizedColor*.

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, // extensison 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.



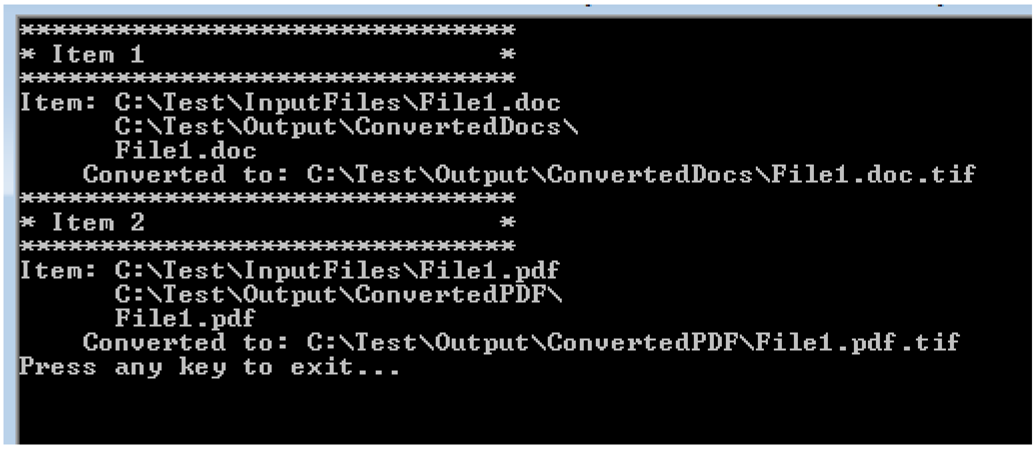
Reading the Results Collection

When converting a list of files, the results are returns 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 returns 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 IList collection 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.

```
IList<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");
```

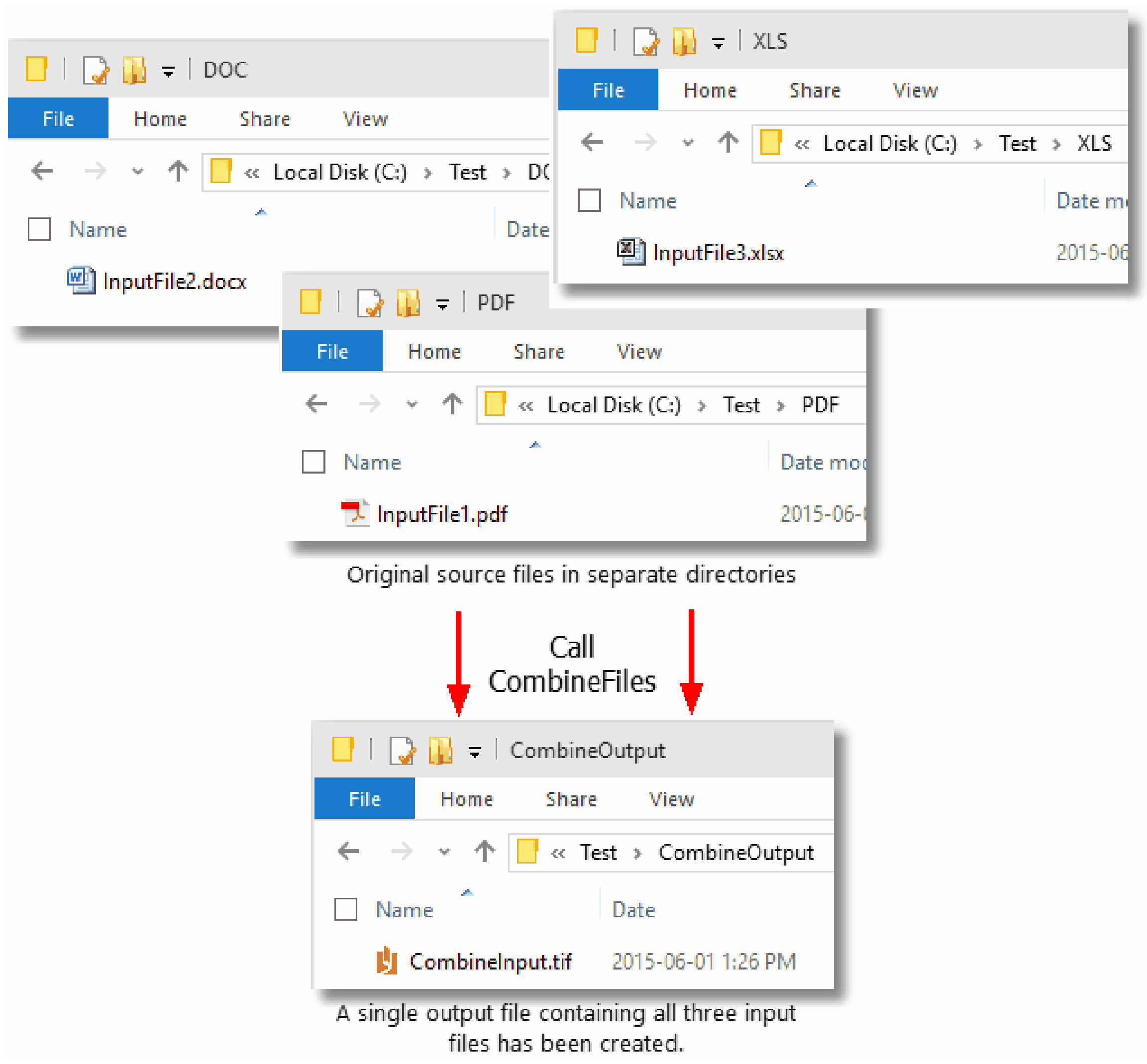
Combining the List of Files

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



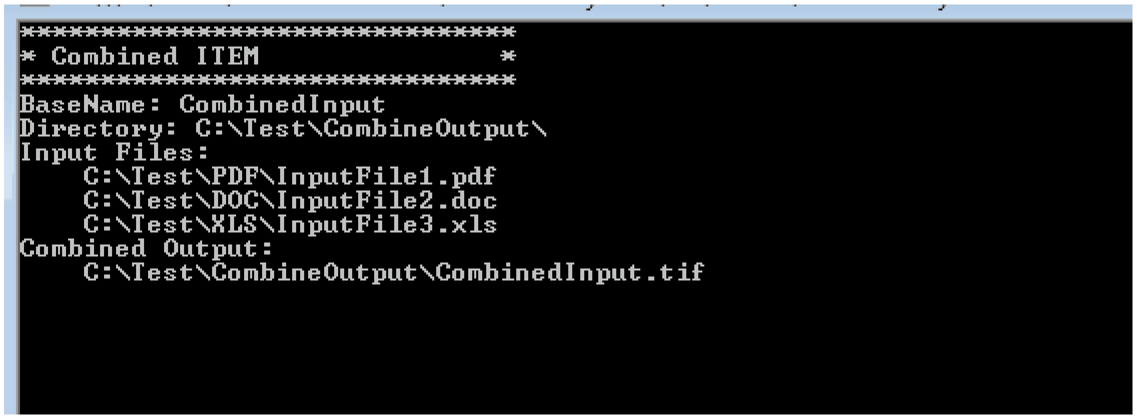
Reading the Results

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);
        }
    }
}
```

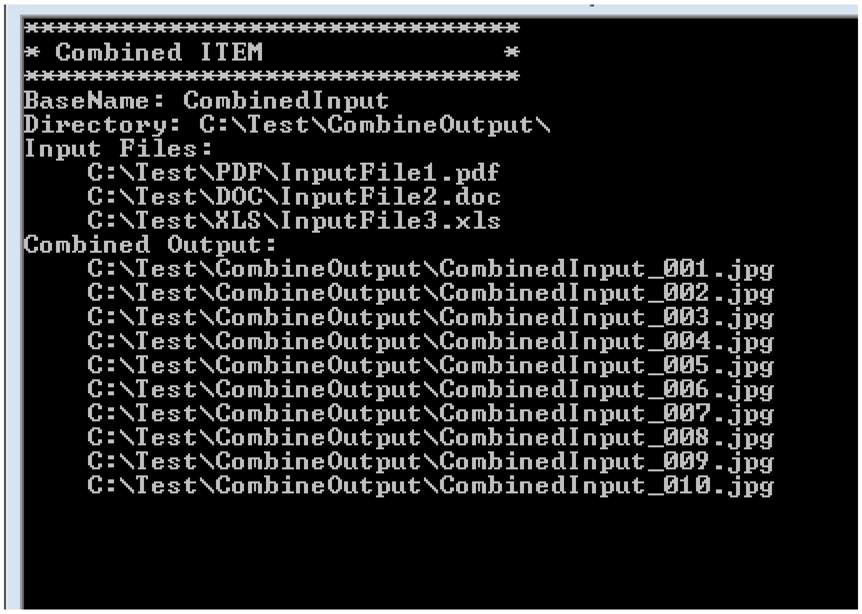
The console output from the above code is shown below.



```
********************************
* 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 *ExcudeFilter* 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 *Draft_*.

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 wil set the created date to the time of the copy.

- o **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.

- o **Ascending** - sorted the files from low to high: 0-9, A-Z.
- o **Descending** - sorts the files from high to low: Z-A, 9-0.

Combining a Folder of Files

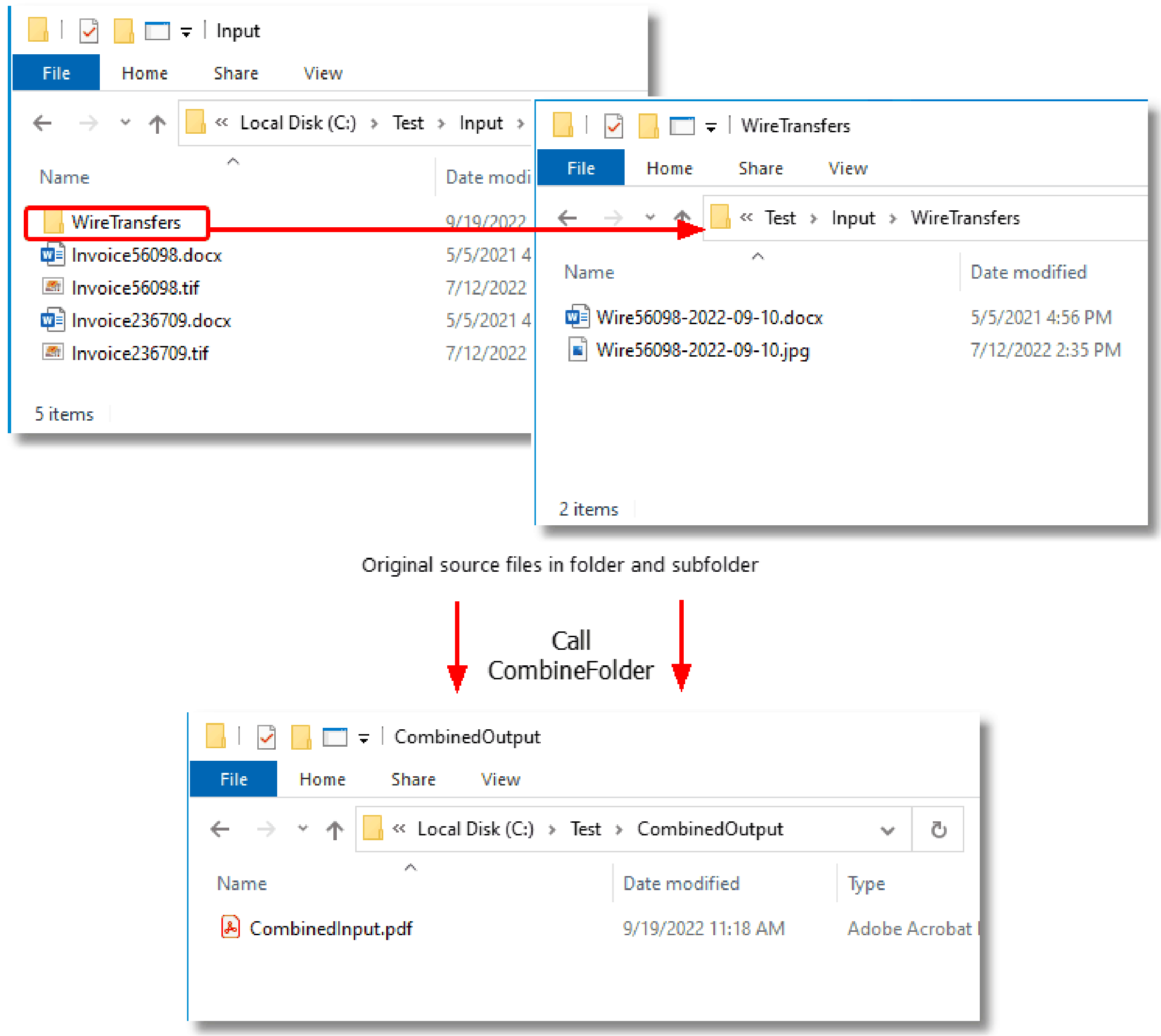
The code sample below will convert all files except TIFF, JPEG and BMP images from the folder *C:\Test\Input\.* Any subfolders are also be searched for files to convert.

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.Asccending); // 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.


Input
File
Home
Share
View
« Local Disk (C:) › Test › Input ›
WireTransfers
File
Home
Share
View
« Test › Input › WireTransfers
CombinedOutput
File
Home
Share
View
« Local Disk (C:) › Test › CombinedOutput

Name	Date modi
WireTransfers	9/19/2022
Invoice56098.docx	5/5/2021 4
Invoice56098.tif	7/12/2022
Invoice236709.docx	5/5/2021 4
Invoice236709.tif	7/12/2022

5 items

Name	Date modified
Wire56098-2022-09-10.docx	5/5/2021 4:56 PM
Wire56098-2022-09-10.jpg	7/12/2022 2:35 PM

2 items

Original source files in folder and subfolder

Call CombineFolder

Name	Date modified	Type
CombinedInput.pdf	9/19/2022 11:18 AM	Adobe Acrobat

A single output file containing all three input files has been created.

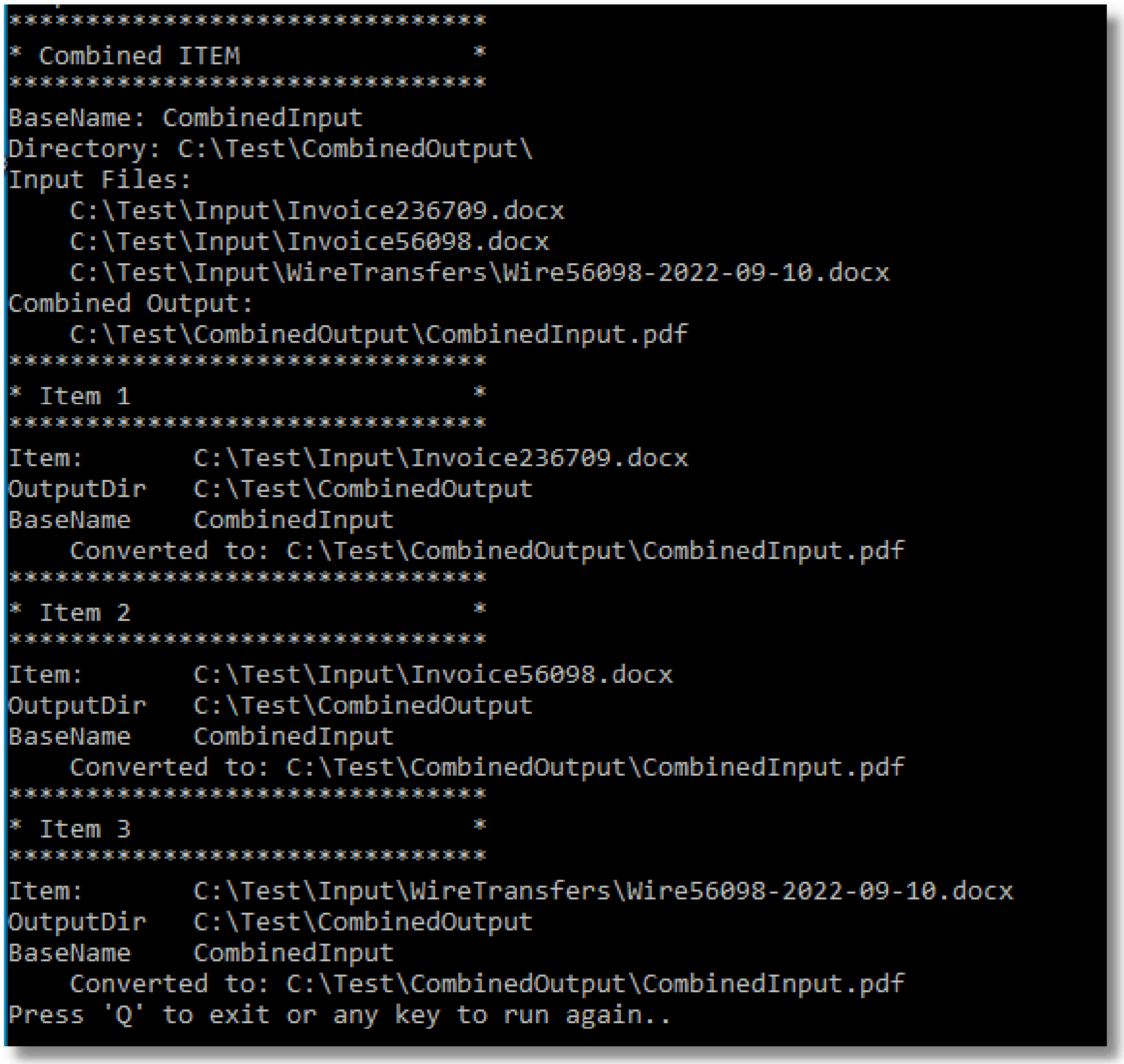
Reading the Results

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 a an IList collection 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 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.

```
IList<PNConvertFileInfo> fileInfoList = new List<PNConvertFileInfo>();
IList<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.

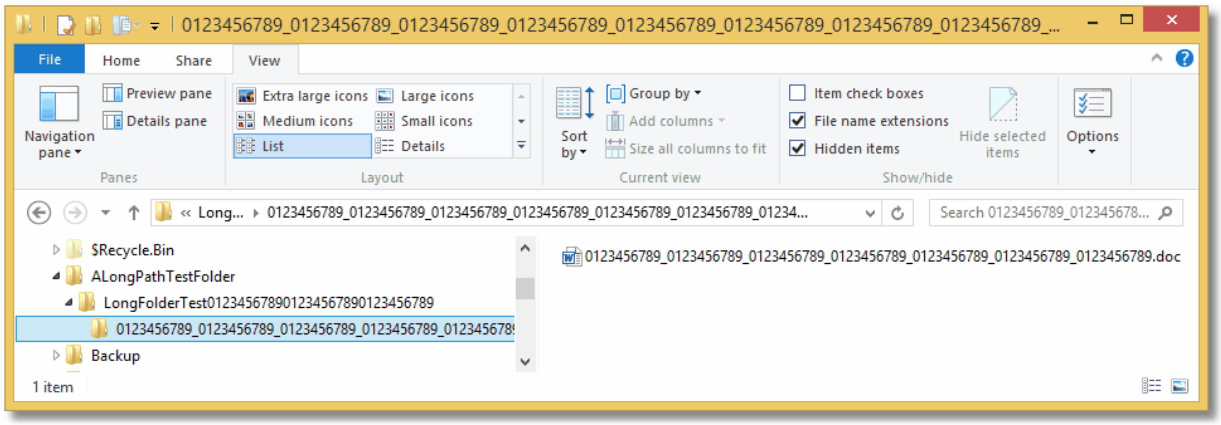
MAXPATH Limitation in Microsoft .NET

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" +
                     "1234567890123456789012345678901234567890123456789012345678901234567890" +
                     "1234567890123456789012345678901234567890123456789012345678901234567890" +
                     "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, // extension 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, // extension profile
                                       String.Empty, // MIME profile
                                       customSettings, // User settings
                                       String.Empty, // not using remote conversion (DCOM)
                                       String.Empty, // use default working folder
                                       String.Empty);
```

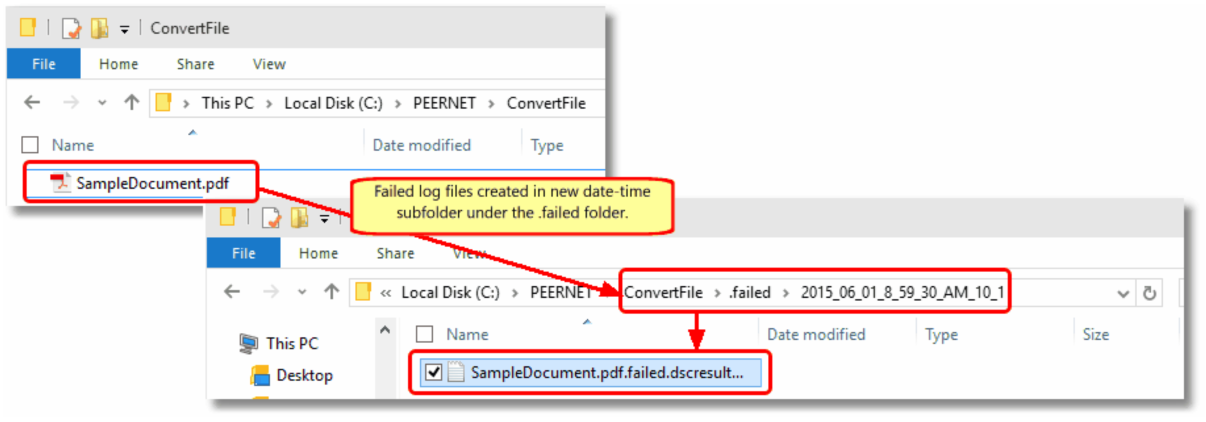
Controlling the Failed Results File Location

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.


ConvertFile
File
Home
Share
View
This PC > Local Disk (C:) > PEERNET > ConvertFile
Name
Date modified
Type
SampleDocument.pdf
Failed log files created in new date-time subfolder under the .failed folder.
« Local Disk (C:) > PEERNET > ConvertFile > .failed > 2015_06_01_8_59_30_AM_10_1
Size
This PC
Desktop
SampleDocument.pdf.failed.dcsresult...

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, // extension 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, // extensison 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\SILogging\* 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\SILogging\";
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, // extensison 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, // extensison 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 PEERNET.ConvertUtility methods.

Custom Setting	**Description**
RemoveDateTimePrefixOnProcessingLoggingFiles	Pass *True* to disable the adding of the unique date, time and thread prefix when a custom file name has not been specified in the *ConvertFileProcessLoggingPath* parameter.
KeepFailedProcessingLoggingFiles	Pass as *False* to disable the automatic creation of SmartInspect logging files when conversion fails. This setting can be overridden by *AlwaysKeepProcessingLoggingFiles*.
AlwaysKeepProcessingLoggingFiles	When set to *True*, the SmartInspect logging files are always created in the %TEMP% or other specified folder for both successful and failed conversions. If set to *False*, no logging files are created. This setting will override the *KeepFailedProcessingLoggingFiles* 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, // extensison 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, // extension profile
                                       String.Empty, // MIME profile
                                       customSettings, // User settings
                                       String.Empty, // not using remote conversion (DCOM)
                                       String.Empty, // use default working folder
                                       String.Empty);
```

Conversion Settings

Conversion settings are used to describe the output created by PEERNET.ConvertUtility and consist of a collection of name-value pairs. These settings can also be used to control the behavior of the individual converters used by Document Conversion Service, such as configuring Word to pass a password or telling Excel to ignore the print areas when printing worksheets.

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);
```

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.
Endorsement Options	Endorsements are header and footer information that can be stamped onto each page of the output created by Document Conversion Service.
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.
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.

Options	Description of Settings
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.
Watermark Stamping	Settings to create a text watermark diagonally across the page.

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:

```
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.

```
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.

Profile Name	Profile Description
Adobe PDF Multipage	Creates Adobe PDF files. The PDF files created using this profile are, where possible, *vector* PDF files. Vector PDF files are also known as *searchable* PDF files. The other PDF profiles provided create *raster*, 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.
BMP 100dpi Color	Creates Windows Bitmap images (one image for each page) at 100dpi. Bitmap images are always serialized.
JPEG 60dpi Color JPEG 120dpi Color JPEG 200dpi Color JPEG 300dpi Color	Creates color JPEG images (one image for each page) at the dots per inch (dpi) specified. JPEG files are always serialized.

Profile Name	Profile Description
JPEG 600dpi Color	
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.
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.
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.

Profile Name	Profile Description
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 *TIFF 300dpi Otimized Color* 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.
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 *SplitFileEveryNPages* setting. When auto-splitting files, serialized naming profile is always used to name each file in the sequence.

Profile Name	Profile Description
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 *SplitFileSizeThresholdInBytes* 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.

E-Discovery Profiles	Profile Description
eDiscovery - Excel - PDF 300dpi Convert Charts Only eDiscovery - Excel - TIFF 300dpi Convert Charts Only	For use with Excel documents, these profiles will print only the embedded charts and any chart tabs in the document.
eDiscovery - Excel - PDF 300dpi Show Formulas eDiscovery - Excel - TIFF 300dpi Show Formulas	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.
eDiscovery PDF 300dpi AutoField Replace eDiscovery TIFF 300dpi AutoField Replace	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.
eDiscovery PDF 300dpi Monochrome Fit On Page eDiscovery TIFF 300dpi Monochrome Fit On Page	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.
eDiscovery PDF 300dpi Span Pages eDiscovery TIFF 300dpi Span Pages	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.

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 methods 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.

An 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 Passthough Converter to skip processing TIFF files, or one that uses Ghostscript to process PDF files instead of Adobe Reader.

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 Adobe Reader, Adobe Acrobat, 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.

Code Sample - File Extension to Converter Mapping

```xml
<?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 Acrobat Reader;Ghostscript;Outside-In AX"/>
    ...
  </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)
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)
Ghostscript	Postscript Files (*.ps) Encapsulated Postscript Files (.eps) Adobe PDF Documents (*.pdf)
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
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)
Microsoft Visio	Visio Drawings (*.vsd)

Converter Name	Supported Document Types
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.

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 if 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.

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.

Conversion Settings - Endorsements Header and Footer Options	
Name:	**Endorsements;Enable**
Values:	**0** - Do not add endorsements 1 - Add specified endorsements to each page
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.

Conversion Settings - Endorsements Header and Footer Options	
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.
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.

Conversion Settings - Endorsements Header and Footer Options	
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 header.

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 &amp; and &quot; 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**

&amp;P	&P	This code is replaced by the current page number.
		
		XML Example:
		`<add Name="Endorsements;HeaderLeftFormat"`
		`     Value="Page &amp;P"/>`
		
		String Example:
		`item.Set("Endorsements;HeaderLeftFormat",`
		`         "Page &P")`

&amp;N	&N	This code is replaced by the total number of pages in the output file.
		
		XML Example:
		`<add Name="Endorsements;HeaderLeftFormat"`
		`     Value="Page &amp;P of &amp;N"/>`
		
		String Example:
		`item.Set("Endorsements;HeaderLeftFormat",`
		`         "Page &P of &N")`

&amp;B	&B	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.
		
		XML Example:
		`<add Name="Endorsements;HeaderLeftFormat"`
		`     Value="&amp;BInternal Use&amp;B -`
		`Confidential"/>`
		
		String Example:
		`item.Set("Endorsements;HeaderLeftFormat",`
		`         "&BInternal Use Only&B -`
		`Confidential")`

&amp;I	&I	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.
		
		XML Example:
		`<add Name="Endorsements;HeaderLeftFormat"`
		`     Value="&amp;IDo Not Copy&amp;I -`
		`Confidential"/>`
		
		String Example:
		`item.Set("Endorsements;HeaderLeftFormat",`

Header and Footer Formatting Codes		
XML Code	**String Code**	**Description**
		`"&IDo Not Copy&I - Confidential")`

&amp;U	&U	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.
		
		XML Example:
		`<add Name="Endorsements;HeaderLeftFormat"`
		`Value="&amp;UDo Not Copy&amp;U - Confidential"/>`
		
		String Example:
		`item.Set("Endorsements;HeaderLeftFormat",`
		`"&UDo Not Copy&U - Confidential")`

&amp;S	&S	Turns font strike though formatting on and off. All text after the first occurrence of the formatting code will be struck though (a line down the middle of the text) until the same formatting code is encountered again.
		
		XML Example:
		`<add Name="Endorsements;HeaderLeftFormat"`
		`Value="&amp;SInternal Use&amp;S - Confidential"/>`
		
		String Example:
		`item.Set("Endorsements;HeaderLeftFormat",`
		`"&SInternal Use Only&S - Confidential")`

&amp;X	&X	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.
		
		XML Example:
		`<add Name="Endorsements;HeaderLeftFormat"`
		`Value="This is &amp;Xsuperscript text&amp;X - Confidential"/>`
		
		String Example:
		`item.Set("Endorsements;HeaderLeftFormat",`
		`"This is &Xsuperscript text&X - Confidential")`

&amp;Y	&Y	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.

Header and Footer Formatting Codes

XML Code	String Code	Description
		XML Example: `<add Name="Endorsements;HeaderLeftFormat" Value="This is &amp;Ysubscript text&amp;Y - Confidential"/>` String Example: `item.Set("Endorsements;HeaderLeftFormat", "This is &Ysubscript text&Y - Confidential")`
&amp;'fontname'	&'fontname'	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. XML Example: `<add Name="Endorsements;HeaderLeftFormat" Value="This is Arial and &amp;'Verdana'this is Verdana."/>` String Example: `item.Set("Endorsements;HeaderLeftFormat", "This is Arial and &'Verdana'this is Verdana.")`
&amp;*n*	&*n*	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. XML Example: `<add Name="Endorsements;HeaderLeftFormat" Value="&amp;14This is Arial 14 point."/>` String Example: `item.Set("Endorsements;HeaderLeftFormat", "&14This is Arial 14 point.")`
&amp;K000000	&K000000	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. XML Example: `<add Name="Endorsements;HeaderLeftFormat" Value="This is &amp;KFF0000Red, this is &amp;K00FF00Green."/>`

Header and Footer Formatting Codes		
XML Code	String Code	Description

		String Example:
		`item.Set("Endorsements;HeaderLeftFormat",`
		`"This is &KFF0000Red, this is &K00FF00Green.")`

&amp;&amp;	&&	Allows the insertion of an ampersand character into the text.
		
		XML Example:
		`<add Name="Endorsements;HeaderLeftFormat"`
		`Value="Printed by Company &amp;&amp;Company"/>`
		
		String Example:
		`item.Set("Endorsements;HeaderLeftFormat",`
		`"Printed by Company && Company")`

&#x0A;	\r\n	Allows the insertion of a newline character into the text.
		
		XML Example:
		`<add Name="Endorsements;HeaderLeftFormat"`
		`Value="Line 1&#x0A;Line 2"/>`
		
		String Example:
		`item.Set("Endorsements;HeaderLeftFormat",`
		`"Line 1\r\nLine 2.")`

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.

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.
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.

Conversion Settings - Word Printing Options	
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 *True* to display any comments in the document. Must be used with *Microsoft.Word.ActiveWindow.View.MarkupMode* to display the comments as balloons or inline, and *Microsoft.Word.Document.PrintOut.Item* set to print document markup.
Values:	String value "**True**" or "False".
Name:	**Microsoft.Word.ActiveWindow.View.ShowFormatChanges** Pass *True* to display any formatting changes made to a document with Track Changes enabled. Must be used with *Microsoft.Word.ActiveWindow.View.MarkupMode* to display the comments as balloons or inline, and *Microsoft.Word.Document.PrintOut.Item* set to print document markup.
Values:	String value "**True**" or "False".
Name:	**Microsoft.Word.ActiveWindow.View.ShowHiddenText** Pass *True* to display any text that was formatted as hidden.
Values:	String value "True" or "False".

Conversion Settings - Word Printing Options	
Name:	**Microsoft.Word.ActiveWindow.View.ShowHighlight** Pass *True* to have highlighted text displayed with the highlighted background.
Values:	String value "True" or "False".
Name:	**Microsoft.Word.ActiveWindow.View.ShowInkAnnotations** Pass *True* to to show handwritten ink annotations in the document. Must be used with *Microsoft.Word.Document.PrintOut.Item* set to print document markup.
Values:	String value "**True**" or "False".
Name:	**Microsoft.Word.ActiveWindow.View.ShowInsertionsAndDeletions** Pass *True* to display any insertions and deletions made to a document with Track Changes enabled. Must be used with *Microsoft.Word.ActiveWindow.View.MarkupMode* set to display the changes as balloons or inline, and *Microsoft.Word.Document.PrintOut.Item* set to print document markup.
Values:	String value "**True**" or "False".
Name:	**Microsoft.Word.ActiveWindow.View.ShowMarkupAreaHighlight** Pass *True* to have the markup area that shows revision and comment ballons 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.
Name:	**Microsoft.Word.PageSetup.BookFoldRevPrinting** Pass *True* 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 *True* 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.

Conversion Settings - Word Page Setup Printing Options	
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
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.

Conversion Settings - Word Page Setup Printing Options	
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".
Name:	**Microsoft.Word.PageSetup.OddAndEvenPagesHeaderFooter** Pass *True* 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.

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.
Name:	**Microsoft.Excel.PrintHiddenWorksheets** Choose whether to print hidden worksheets or not.
Values:	**False** - do not print hidden worksheets. True - print hidden worksheets.

Conversion Settings - Excel General Formatting & Printing Options	
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 *Microsoft.Excel.PrintOut* 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 *Microsoft.Excel.PrintHiddenWorksheets* is true, then hidden sheets are included. Applies to the **Microsoft.Excel.PrintOut** options *PrintOutWorkbookOnly*, *PrintOutChartsOnly*, *PrintOutChartsThenWorkbook* and *PrintOutWorkbookThenCharts.* The range applies to both sheets and charts in the workbook. This print filter can be combined with *Microsoft.Excel.PrintSheetsRangeByName*, *Microsoft.Excel.PrintFirstNSheets, Microsoft.Excel.PrintLastNSheets,* and *Microsoft.Excel.PrintIfSheetNameMatchesRegex*.
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.Exce.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** 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 carriange return before any current comment text. This must be used with *Microsoft.Excel.PageSetup.PrintComments* set to *PrintSheetEnd* to include the cell formulas listed by cell reference at the end of each sheet. To append the formula to the cell contents instead of inserting at the beginning, set *Microsoft.Excel.Worksheet.PrependCellFormulaToCommentText* to False.
Values:	False - do not add/update existing comments with the cell formula. **True** - add/update existing comment with the cell formula.
Name:	**Microsoft.Excel.Worksheet.PrependCellFormulaToCommentText** When using *Microsoft.Excel.Worksheet.IncludeCellFormulasAsComments*, 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 *False*.
Values:	False - append the cell formula to the end of any existing comment text. **True** - prepend the cell formula to the beginning of any existing comment text.
Name:	**Microsoft.Excel.Worksheet.PrintOut.IgnorePrintAreas** When set to *True*, any print areas set on the worksheet will be ignored and the entire worksheet printed. Use with *Microsoft.Excel.Worksheet.PrintOut.ResetAllPageBreaks* to print the worksheet differently from the printing options in the worksheet.
Values:	**False** - prints using any print area set on the worksheet. True - prints the entire worksheet.
Name:	**Microsoft.Excel.Worksheet.ShowAllData** 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 *Microsoft.Excel.AutoFitRows* and *Microsoft.Excel.AutoFitColumns.*
Values:	**False** - Leave data filtered (hidden). True - Show all the data on the worksheet.

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.UnfreezePanes** If the spreadsheeet has any non-scrolling, "frozen" panes, 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**".

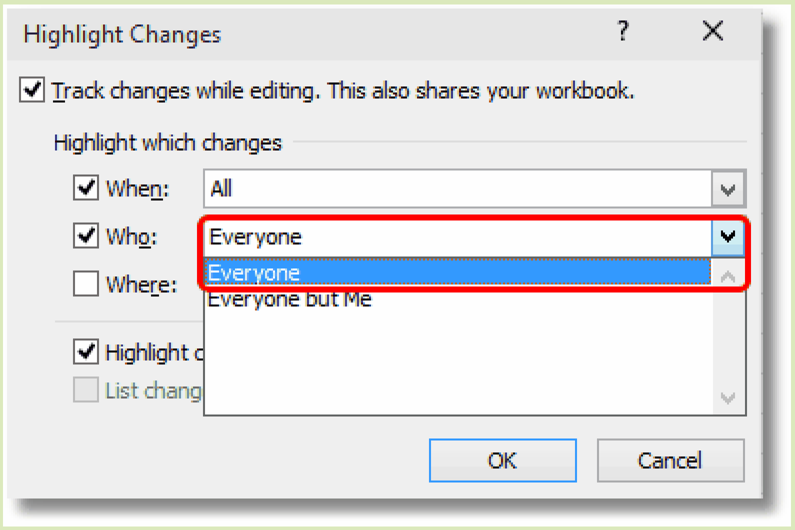
Name: **Microsoft.Excel.TrackChanges.ListChangesOnNewSheet**

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 that language to list the tracked changesfor 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 Hightlight Changes dialog in your version of Excel. The English version is shown below.


Highlight Changes

☑ Track changes while editing. This also shares your workbook.

Highlight which changes

☑ When: All

☑ Who: Everyone

☐ Where:

Everyone
Everyone but Me

☑ Highlight c

☐ List chang

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 *True* a different header or footer is used for the first page of the worksheet (*applies to Office 2007 or higher)*.
Values:	String value "True" or "**False**".
Name:	**Microsoft.Excel.PageSetup.Draft** Prints the worksheet without graphics when set to *True*.
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 *True*.
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 *True*.
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 *True* 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 *True* then grid lines will be printed on each spreadsheet.
Values:	String value "True" or "False".
Name:	**Microsoft.Excel.PageSetup.PrintHeadings** If set to *True* then column and row headings will be printed on each spreadsheet.
Values:	String value "True" or "False".
Name:	**Microsoft.Excel.PageSetup.PrintNotes** Set to *True* 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 *DevMode settings;Resolution* 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 *True* 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.

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
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

Conversion Settings - PowerPoint Page Setup	
	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".
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".

Conversion Settings - PowerPoint Print Options	
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
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".

Conversion Settings - PowerPoint Print Options	
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.

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**".

Conversion Settings - Adobe Reader Print Options	
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 the 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.
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".

Conversion Settings - Adobe Reader Print Options

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.

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.
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, *font-style: italic;* and *font-weight: bold* 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.

Conversion Settings - Page Setup

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.

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 *Internet Explorer* section in the *ApplicationFactory* section of the Document Conversion Service application configuration file. See also Application Level Configuration Settings.

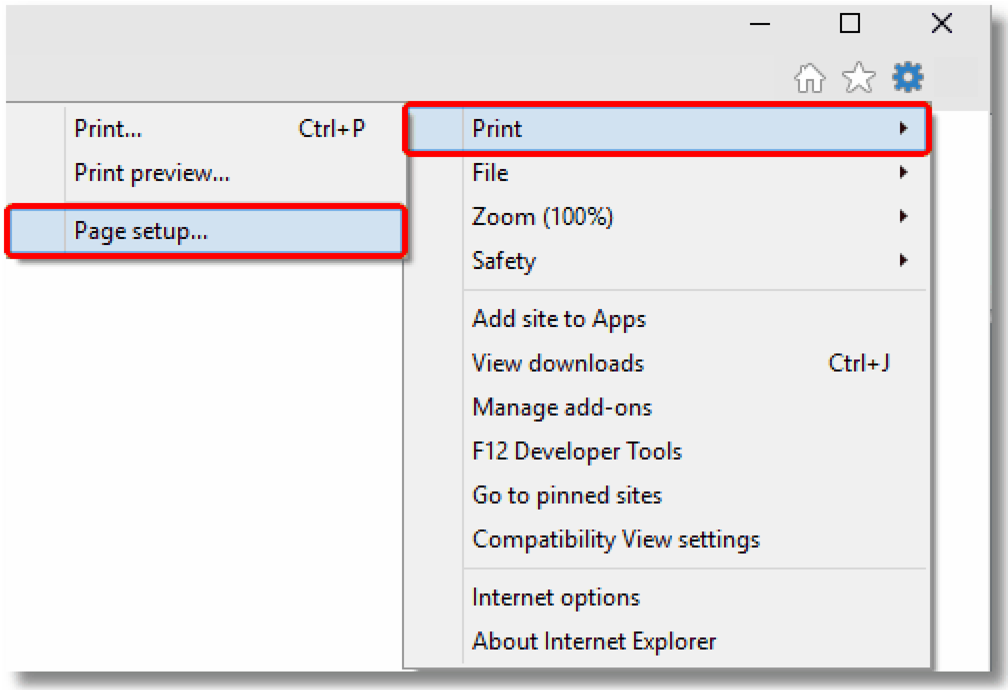
```
<AppFactory Name="Internet Explorer"
            Type="PEERNET.PNDocConv.Applications.PNInternetExplorerApplicatio
            Assembly="PNInternetExplorerApplicationFactory">
  <Settings>
    ...
    <add Name="ConverterPlugIn.PNIExplorer.ShrinkToFitScaleMin" Value="30" />
  </Settings>
</AppFactory>
```

Values: String value "**True**" or "False".

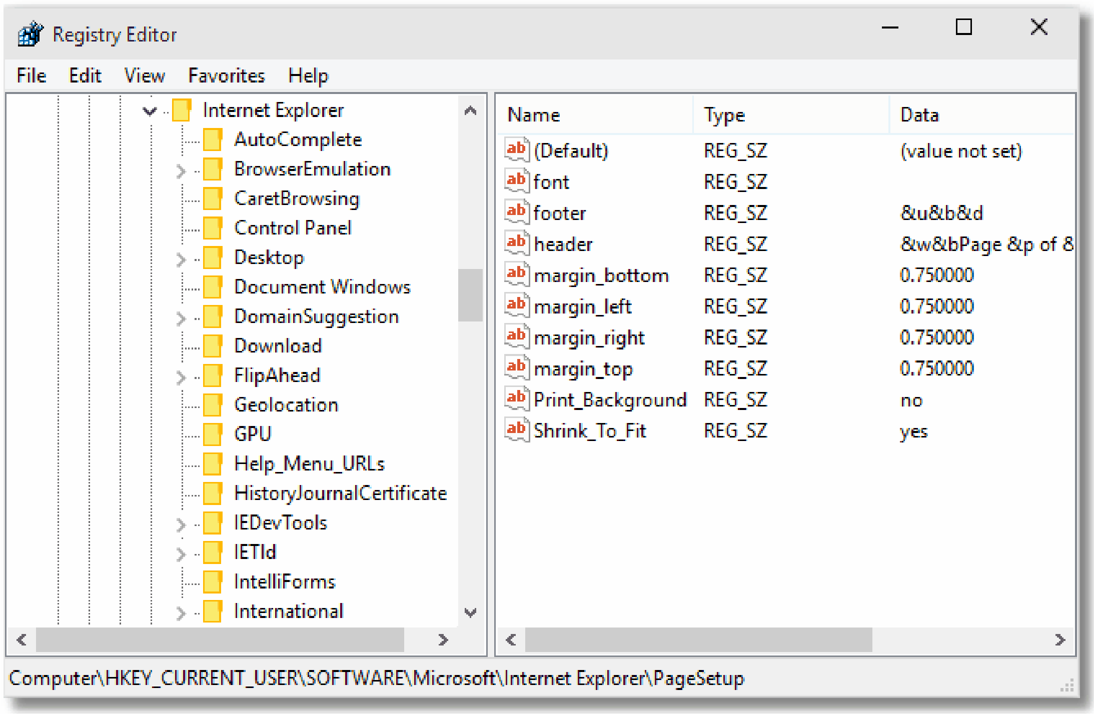
Adding Headers, Footers and Fonts to HTML Conversion

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.

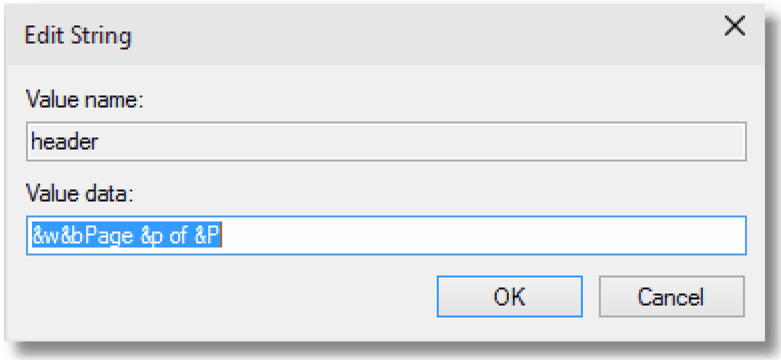


- Internet Explorer
  - AutoComplete
  - BrowserEmulation
  - CaretBrowsing
  - Control Panel
  - Desktop
  - Document Windows
  - DomainSuggestion
  - Download
  - FlipAhead
  - Geolocation
  - GPU
  - Help_Menu_URLs
  - HistoryJournalCertificate
  - IEDevTools
  - IETld
  - IntelliForms
  - International

Name	Type	Data
(Default)	REG_SZ	(value not set)
font	REG_SZ	
footer	REG_SZ	&u&b&d
header	REG_SZ	&w&bPage &p of &
margin_bottom	REG_SZ	0.750000
margin_left	REG_SZ	0.750000
margin_right	REG_SZ	0.750000
margin_top	REG_SZ	0.750000
Print_Background	REG_SZ	no
Shrink_To_Fit	REG_SZ	yes

Computer\HKEY_CURRENT_USER\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 &amp; for the string to be parsed correctly.



Value name: header

Value data: &w&bPage &p of &P

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, IE1
          <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>
```

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.

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 *Fonts* 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.

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.
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 fasted 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.

Conversion Settings - Toolkits and Scaling Modes	
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.
Name:	**ConverterPlugIn.PNImageConverter.ResampleImageToMaxWidthOrHeightInPixels** Dynamically sample the output image to a specific maximum width or height, which ever 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.

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.
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.

Conversion Settings - OutsideIn AX Printing	
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.

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.
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)

Conversion Settings - Save	
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.
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

Conversion Settings - Save	
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**.
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.

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

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. Note Envelope #9 Envelope #10 Envelope #11 Envelope #12 Envelope #14 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

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. 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 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	Include JobID	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.

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 *Advanced File Naming;Variables S* below.
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

Conversion Settings - Advanced File Naming	
Name:	**Advanced File Naming;Variables S** Comma-delimited list of variables that correspond to the placeholders in the format string supplied in *Advanced File Naming;Format string S* 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 *Advanced File Naming;Variables SJ* 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 *Advanced File Naming;Format string SJ* 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 *Advanced File Naming;Variables M* below.
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

Conversion Settings - Advanced File Naming	
Name:	**Advanced File Naming;Variables M** Comma-delimited list of variables that correspond to the placeholders in the format string supplied in *Advanced File Naming;Format string M* 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 *Advanced File Naming;Variables MJ* 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 *Advanced File Naming;Format string MJ* 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.

Variable	Type and Format String Place Holder	Description
$(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).
$(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)
		$(FileNumber)
		
		`C:\Test\Invoice_001.tif`
		`C:\Test\Invoice_002.tif`
		`C:\Test\Invoice_003.tif`
		`...`

Serialized w/ JobID	%3d_%s_%3d	$(JobID)
		$(OutputFileName)
		$(FileNumber)
		
		`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.

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)
Name:	**Image Options;Fax Use Printer Resolution**
Values:	**0** - Do not use printer resolution 1 - Use printer resolution

Conversion Settings - Image Options	
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
Name:	**Image Options;Fax Page Scaling H Align**
Values:	Left - Align image left **Middle** - Align image in the center Right - Align image right

Conversion Settings - Image Options	
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.

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
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.

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
Name:	**PDF File Format;Content encoding**
Values:	None - No compression ZIP - ZIP compression RLE - Packbits (run length) compression **LZW** - LZW compression

Conversion Settings - PDF File Format	
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.

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
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

Conversion Settings - PDF Security	
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.

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.

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 *Trim Threshold* 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 *Trim Threshold* below.
Values:	**0** - Do not trim top of page 1 - Trim top of page
Name:	**Processing;Trim right** Trim all areas from the right side of the page, based on the *Trim Threshold* 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 *Trim Threshold* below.
Values:	**0** - Do not trim bottom of page 1 - Trim bottom of page

Conversion Settings - Processing	
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
Name:	**Processing;Crop left** Applies when Crop Option is set to *crop region*.
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 *crop region*.
Values:	Same as *Processing;Crop left* above

Conversion Settings - Processing	
Name:	**Processing;Crop width** Applies when Crop Option is set to 0 for *crop region*.
Values:	Same as *Processing;Crop left* above.
Name:	**Processing;Crop height** Applies when Crop Option is set to 0 for *crop region*.
Values:	Same as *Processing;Crop left* above
Name:	**Processing;Crop margin left** Applies when Crop Option is set to 1 for *crop margins*.
Values:	Same as *Processing;Crop left* above
Name:	**Processing;Crop margin top** Applies when Crop Option is set to 1 for *crop margins*
Values:	Same as *Processing;Crop left* above
Name:	**Processing;Crop margin right** Applies when Crop Option is set to 1 for *crop margins*
Values:	Same as *Processing;Crop left* above
Name:	**Processing;Crop margin bottom** Applies when Crop Option is set to 1 for *crop margins*
Values:	Same as *Processing;Crop left* above
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

Conversion Settings - Processing	
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 *Processing;Copy to width* above.
Name:	**Processing;Copy to IAM Left** The desired left area margin settings for the new image.
Values:	Same as *Processing;Copy to width* above
Name:	**Processing;Copy to IAM Top** The desired top area margin settings for the new image.
Values:	Same as *Processing;Copy to width* above
Name:	**Processing;Copy to IAM Right** The desired right area margin settings for the new image.
Values:	Same as *Processing;Copy to width* above
Name:	**Processing;Copy to IAM Bottom** The desired bottom area margin settings for the new image.
Values:	Same as *Processing;Copy to width* above

Conversion Settings - Processing	
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
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

Conversion Settings - Processing	
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**.
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**

Conversion Settings - Processing	
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
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.

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
Name:	**Advanced Features;Custom Paper Height** Specify the height of the custom paper size. *Custom Paper Enable* 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

Conversion Settings - Advanced Features	
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
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

Conversion Settings - Advanced Features	
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**
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

Conversion Settings - Advanced Features	
Name:	**Advanced Features;Extract Text Emit Page Breaks**
Values:	0 **1**
Name:	**Advanced Features;Control Strings Enabled**
Values:	**0** 1

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.

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.

Deploying Applications

When deploying applications build with PEERNET.ConvertUtility, the following files must be included with your application.

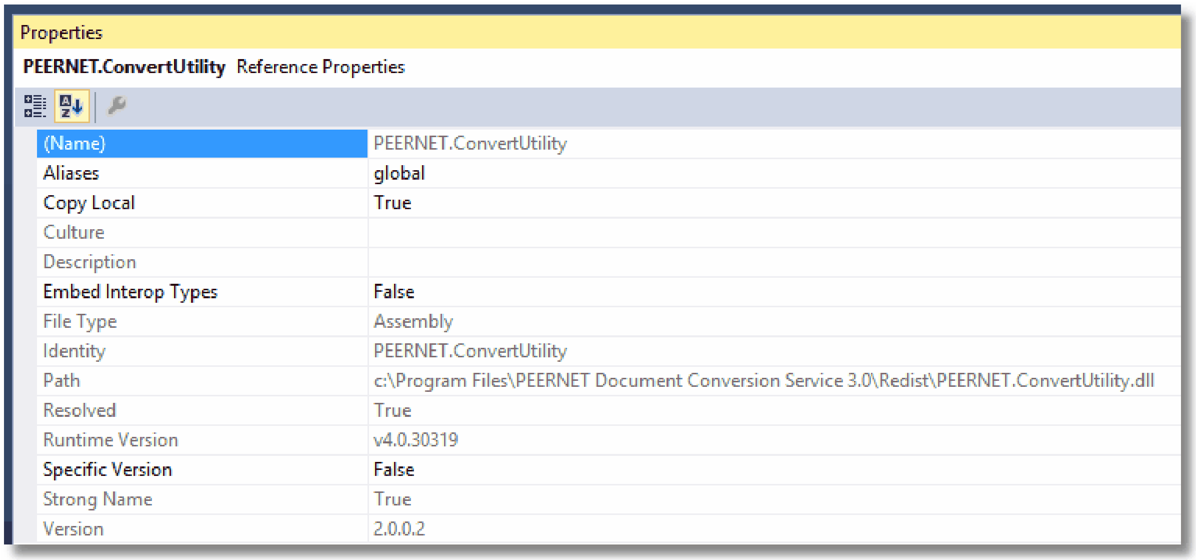
Changes to Files starting with version 3.27

Starting with Document Conversion Service 3.27, the Xfinium library used by PEERNET.ConvertUtility is **Xfinium.Pdf.Win.dll**. Any programs using PEERNET.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.

- PEERNET.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 PEERNET.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.

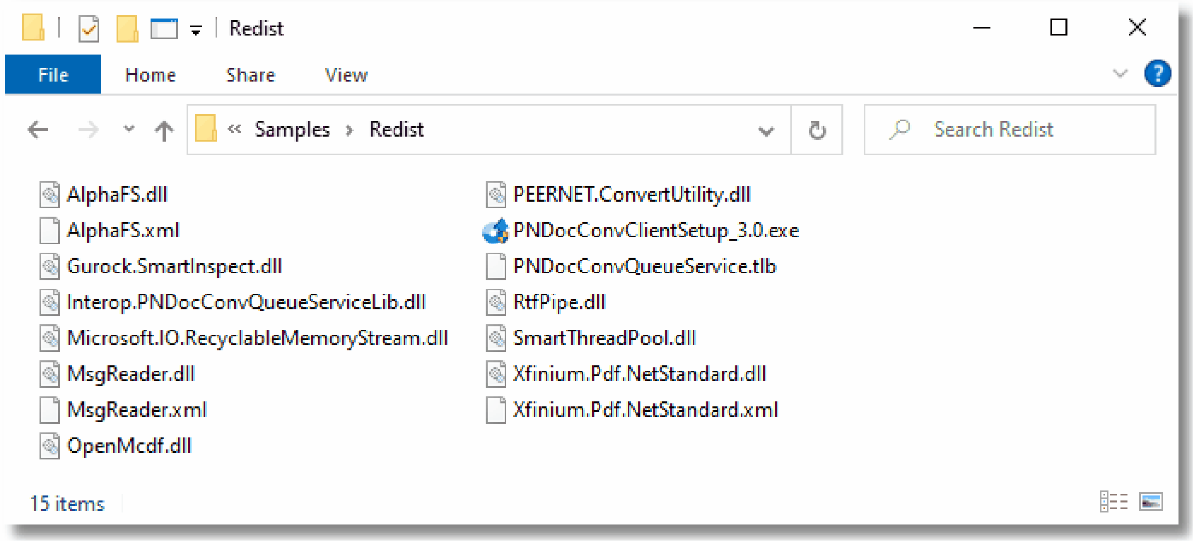


Properties

PEERNET.ConvertUtility Reference Properties

(Name)	PEERNET.ConvertUtility
Aliases	global
Copy Local	True
Culture	
Description	
Embed Interop Types	False
File Type	Assembly
Identity	PEERNET.ConvertUtility
Path	c:\Program Files\PEERNET Document Conversion Service 3.0\Redist\PEERNET.ConvertUtility.dll
Resolved	True
Runtime Version	v4.0.30319
Specific Version	False
Strong Name	True
Version	2.0.0.2

If you need to manually copy these files you can find them in the **\Samples\Redist** folder under the Document Conversion Service installation tree.



AlphaFS.dll
AlphaFS.xml
Gurock.SmartInspect.dll
Interop.PNDocConvQueueServiceLib.dll
Microsoft.IO.RecyclableMemoryStream.dll
MsgReader.dll
MsgReader.xml
OpenMcdf.dll
PEERNET.ConvertUtility.dll
PNDocConvClientSetup_3.0.exe
PNDocConvQueueService.tlb
RtfPipe.dll
SmartThreadPool.dll
Xfinium.Pdf.NetStandard.dll
Xfinium.Pdf.NetStandard.xml

15 items

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 ore 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 ore 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 ore 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 suppplied 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 an 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 *SettingsProfile*. An empty list was passed for *SettingsList.* An empty, or badly formatted profile was passed for *ExtensionsProfile*. Null or empty string passed for *InputFile*. A name for *RemoteComputerName* was passed but no corresponding *ConversionWorkingFolder* specified.
FileNotFoundException	*InputFile* doesn't exist.
DirectoryNotFoundException	When a path to *OutputFolder* is specified but does not exist or is invalid. When *ConversionWorkingFolder* 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 IList 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

IList<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.

IList<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 an 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 *SettingsProfile*. An empty list was passed for *SettingsList.*

	An empty, or badly formatted profile was passed for *ExtensionsProfile*. An empty list was passed for *FileInfoList* or *FileList*. A name for *RemoteComputerName* was passed but no corresponding *ConversionWorkingFolder* specified.
FileNotFoundException	One of the input files in the *FileInfoList* or *FileList* does not exist or cannot be accessed.
DirectoryNotFoundException	One of the output paths in the *FileInfoList* does not exist, or when *OutputFolder* is specified but the path does not exist or is invalid. When *ConversionWorkingFolder* 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 specfied.

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 *SettingsProfile* An empty list was passed for *SettingsList* An empty, or badly formatted profile was passed for *ExtensionsProfile*. Null or empty string passed for *InputFile*. A name for *RemoteComputerName* was passed but no corresponding *ConversionWorkingFolder* specified.
FileNotFoundException	*InputFile* doesn't exist.
DirectoryNotFoundException	The path to *InputFolder* is specified but does not exist or is invalid. The path to *OutputFolder* is specified but does not exist or is invalid. The path to *ConversionWorkingFolder* 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 *SettingsProfile*. An empty list was passed for *SettingsList.* An empty, or badly formatted profile was passed for *ExtensionsProfile*. An empty list was passed for *FileList*. An empty name was passed for *OutputName*. A name for *RemoteComputerName* was passed but no corresponding *ConversionWorkingFolder* specified.
FileNotFoundException	One of the input files in the *FileList* does not exist or cannot be accessed.
DirectoryNotFoundException	The output path for *OutputFolder* is specified but the path does not exist or is invalid. When *ConversionWorkingFolder* 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 specfied.

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 *SettingsProfile*. An empty list was passed for *SettingsList.* An empty, or badly formatted profile was passed for *ExtensionsProfile*. 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 *OutputName*. A name for *RemoteComputerName* was passed but no corresponding *ConversionWorkingFolder* specified.
FileNotFoundException	One of the files found to process was removed or cannot be accessed when conversion was attempted.
DirectoryNotFoundException	The output path for *OutputFolder* is specified but the path does not exist or is invalid.

	When *ConversionWorkingFolder* 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	*Optional*. 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.OutputPathe
```

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 **IList<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	*Reserved for future use.*
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 that 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*

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 PNCombineItem, 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 PNCombineItem to an XML file on disk.

Syntax

```
expression.SerializeToXML(FilePath)
```

where `expression` is a PNCombineItem 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 PNCombineItem 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 PNCombineItem 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 PNCombineItem 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 PNCombineItem 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 PNCombineItem 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 PNCombineItem 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 PNCombineItem 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 PNCombineItem 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 *Portrait* or *Landscape*.
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 an 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 an 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 *Portrait* or *Landscape*.
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 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 **IList<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 No sorting is performed, files are listed as returned from the system. *Default.*	0
Name Files are sorted by file name.	1
DateCreated Files are sorted using the file creation date on the file.	2
DateModifed Files are sorted using the last modified date on the file.	3

PNFileSortOrder

Description

Determines the sorting order, if any, applied when picking up files from an input folder.

Details

Name	Value (int)
Ascending Sorts the files from low to high: 0-9, A-Z.	0
Descending Sorts the files from high to low: Z-A, 9-0.	1