TiffDLL200

32/64-bit

Version 2

Informatik Inc

User Guide

26 March 2015

Check frequently for software upgrades at
http://informatik.com/updates/tiffdll200.html

For latest user guide go to
www.informatik.com/manuals.html

Contents

Important: Always make sure that all affected files are backed up. Always check the output. TiffDLL200 is supplied as is, without any liability, direct or indirect. Before you TiffDLL200 read and agree with the terms as shown in 'License, Warranty, Disclaimer' section below.

Check frequently for software upgrades at http://informatik.com/updates/tiffdll200.html.

Introduction

TiffDLL200 is a developer's kit for Tiff and Graphics files processing. TiffDLL200 includes the following functions:

- Text annotations
- Redactions
- Highlighting
- Watermarks
- Resizing
- Canvas resizing
- Cropping
- Shifting
- Border cleanup
- Deskew
- Resolutions
- Rotation
- Flip
- Inversion
- Brightness
- Color flooding
- Serialization (split)

The following graphics formats are supported:

Single and multipage Tiff, PNG, BMP, Gif, JPEG).

PDF format is not supported; however sample code for conversions from Tiff to PDF is available for download. See PDF section below.

The extension name of the output file controls the output graphics format. If the extension is nonspecific, you can specify the format in the `obj.init.Format` property; otherwise the file

will be saved in the Tiff format. Color depth, Tiff compressions and JPEG quality can also be specified in the Format property.

For a Strong Name version please contact technical support.

A separate **command-line version** is also available, free to TiffDLL200 licensees.

Important: Always make sure that all affected files are backed up. Always check the output. TiffDLL200 is supplied as is, without any liability. Before you useTiffDLL200 read and agree with the terms as shown in 'License, Warranty, Disclaimer' section below. Check frequently for software upgrades at http://informatik.com/updates/tiffdll200.html.

Demo Version

The trial version is free to use, but subject to the terms as shown in 'License, Warranty, Disclaimer' section below. Please read and agree with the terms before you use the software.

At random, instead of processing a file, the trial version creates a separate demo nag file (not affecting the users' files). If you run the demo version in real mode you must check the output and make sure that all files had been processed.

TiffDLL100 DLL licensees: The upgrade to TiffDLL200 is free for TiffDLL100 licensees. Use the same license code.

When you purchase a license, you receive a registration code (unlock code). In your program code, enter this registration code in the obj.init.Licensecode= statement, for example:
obj.init.Licensecode = " SHXYZBEC3042124123456789";

Syntax – Must-Read

The code samples below are generally given in the C# format. The VB.NET syntax is virtually the same, except for the different commenting-out characters and the ending semi-colons.

The sample code uses `obj` as the TiffDLL200.TiffDLL object name. You can declare the object under any other name. The terms *`op`* (standing for Option/Operation) and `init` (standing for Initiate) are unchangeable structure names.

When writing your project, use the IntelliSense whenever possible. There is little typing. Simply type `obj.op` or `obj.init` and a dot, and **IntelliSense** should give you all the options to choose from. After you type an equal character (=), again IntelliSense should give you a list of options. In C# you may need to **press CTRL+J to see the options**. .

The values for location and size are in Inches or Centimeters, depending on the system's default metrics setting. The Metrics setting, if used, overwrites the system setting. The values are of Double/Floating data type, so they are entered with the 'f' suffix, for example 1.25f. In VB.NET the F suffix is not required.

The specifications for locations are in Points (PointX), X for horizontal, Y for vertical. If the values given are negative (minus values), the values refer to the right border or bottom border. For example, -1 means 1 Inch (or Centimeter) from the bottom up, or from the Right border. The specifications for size are in Rectangles (RectangleX), X for Left, Y for Top, Width and Height. Size number must always be positive, except when used as 'Relative' (increase/decrease over original size).

TiffDLL200 essentially processes one file per call. You can iterate through a folder with Visual Studio code such as:

```
foreach (string fname in System.IO.Directory.GetFiles("c:\\somedir", "*.tif"))
{
     System.Windows.Forms.MessageBox.Show(fname);
}
```

Each execution can cover several options. The options are executed in the following order:

1. Flooding
2. Rotation
3. Flip
4. Deskew
5. Brightness
6. Invert
7. Resolution
8. Border Cleanup
9. Crop
10. Shift
11. Resize
12. Canvas
13. Image insert
14. Annotation
15. Annotation-Multi
16. Redaction
17. Highlight
18. Watermark

There are two methods of Annotations:

- Single annotations with fixed or flexible locations, and
- Multiple annotations (maximum 100 entries). Both methods are explained below.

Most options are optional. In the samples below, the optional items are commented out.

Some options have many property selections. The samples typically show only one, with the '//etc.' mark. Use IntelliSense to see the full list.

Your First Project

You can download a basic starter project (VB or C#) from:

`www.informatik.com/TIFFDLL/TestProjectTiffDLL200_CSharp.zip`
`www.informatik.com/TIFFDLL/TiffDLL200_SampleProject_VB.zip`

The download includes complete sample projects for both C# and VB.NET. Just unzip the files and run the project. All required files are in the correct folders and referenced. The sample project may have versions of the TiffDLL200 that are not the latest. Always check for the latest available product version.

To create your own project, proceed as follows:

1. Create a Visual Studio project (Framework 4.0, 32-bit or 64-bit).

2. Copy the TiffDLL200.dll file to the project's Bin (or Bin/Release) folder.

3. In the project add a reference to the TiffDLL200 file.

   (In some cases, you may need to change the Platform Target in Project Properties | Build from 'Any CPU' to 'x86', or as appropriate.)

4. Add the following code (use that code as a template for your projects). Compile and test.

C# Code (Bold lines are mandatory)

```
TiffDLL200.TiffDLL obj = new TiffDLL200.TiffDLL();
//obj.init.Licensecode = "XXX...";
obj.init.File_in = "c:\\somedir\\someinputfile.tif";
obj.init.File_out = "c:\\somedir\\outputfile.tif";
//obj.init.OverwriteFile = true;
//obj.init.Pages_include = "1-5";
//obj.init.Pages_change = "1-2";
//obj.init.Metrics = TiffDLL200.InchCentimeter.Inches;
//obj.init.SerialSuffix = "0001";
//obj.init. Format.TiffCompression = TiffDLL200.TiffCompression.LZW_mono_or_color;
//obj.init.ErrorLogfile = "c:\\somedir\\errorlog.txt";
//obj.init.ErrorMessage_off = true;
//.....add code for image processing, as detailed below
//.....add code for image processing, as detailed below
//.....add code for image processing, as detailed below
int err = obj.run();
//.....add code for error handling
//System.Windows.Forms.MessageBox.Show(err.ToString() + obj.info.lasterror);
obj = null;
System.GC.Collect();
```

VB.NET Code (Bold lines are mandatory)

```
Dim obj As TiffDLL200.TiffDLL = New TiffDLL200.TiffDLL()
'obj.init.Licensecode = "XXX..."
obj.init.File_in = "c:\\somedir\\someinputfile.tif"
obj.init.File_out = "c:\\somedir\\outputfile.tif"
'obj.init.Pages_include = "1-5"
'obj.init.Pages_change = "1-2"
'obj.init.OverwriteFile = true
'obj.init.Metrics = TiffDLL200.InchCentimeter.Inches
'obj.init.SerialSuffix = "0001"
'obj.init. Format.TiffCompression = TiffDLL200.TiffCompression.LZW_mono_or_color
'obj.init.ErrorLogfile = "c:\\somedir\\errorlog.txt"
'obj.init.ErrorMessage_off = true
' ..... add code for image processing, as detailed below.
' ..... add code for image processing, as detailed below.
' ..... add code for image processing, as detailed below.
Dim err As Integer = obj.run
'.....add code for error handling
System.Windows.Forms.MessageBox.Show(err.ToString & obj.info.lasterror)
obj = Nothing
System.GC.Collect()
```

Basic Settings

Licensecode

If you purchased a license, enter the 24-characterlong license code.

```
obj.init.Licensecode = "XXXXXXXXXXXXXXXXXXXXXXXX";
```

File-in, File-out

These are two required entries.

The File-out path must refer to an existing directory.
Do not save to a root directory, such as c:\. Use a subfolder.

The extension name of the output file will control the graphics format of the file. For example, an output file named 'somefile.png' creates a PNG file.

If the extension name of the output file is nonspecific, you can specify the graphics format in the `obj.init.Format` property; otherwise the file will be saved in the Tiff format. Color depth, Tiff compressions and JPEG quality can also be specified in the Format property.

```
obj.init.File_in = "c:\\somedir\\someinputfile.tif";
obj.init.File_out = "c:\\somedir\\outputfile.tif";
```

Pages_include, Pages_change

If you are processing all the pages of a multi-page file, keeping the pages in the original page order, then you do not use these two entries.

If you want only some of the pages included in the output file, or if you want to arrange the pages in a different page order, specify the range(s) in the `Pages_include`.

If you make edit changes (such as annotations, etc.) to only some of the pages, but want all the pages included in the output, in the original order, then specify the range(s) of pages that you want edited in `Page_change`.

You can also use both the Pages_include and the Pages_change in conjunction.

Samples of page specifications:

- 1-5
- 1-5,10-20,25-30
- 1-5,10,20-16
- 1-10-20-E
- 3,7,8,10-13
- EE-1

(E stands for Last page, EE stands for second last page, EEE third-last, EEEE forth-last)

```
obj.init.Pages_include = "1-5,10-20";
obj.init.Pages_change = "1-2";
```

Overwrite File

By default, existing files are not overwritten; an error message is generated. You can set the OverwriteFile to 'true' so that existing files are automatically overwritten without warning. Use with caution.

```
obj.init.OverwriteFile = true;
```

Metrics

The location and size parameters use either Inches or Centimeters, depending on the system setting. Normally you need not set or change this property. You can overwrite the system setting with:

```
obj.init.Metrics = TiffDLL200.InchCentimeter.Inches;
obj.init.Metrics = TiffDLL200.InchCentimeter.Centimeters;
```

For some of the options you can specify pixels instead of Inch or Centimeter.

Serial Suffix

If the output of a multipage file must be serialized (one file per page), specify incrementing the serial suffix, for example "0001". The serial suffix must be numeric. If you need a separator character between the file base name and the suffix, add the separator character to the output base name. If the serial must be in the files' extension, specify the output file name without an extension.

If a multi-page Tiff file is converted to a graphics file that does not support multi-pages (for example .png), a generic suffix number is automatically added, but you can specify your own serial.

It is suggested that you set the Overwrite to 'true' when serializing pages.

Error Log File

If you wish to post all errors to an error long, specify the error log file. If the folder does not exist, the system will create it.

```
obj.init.ErrorLogfile = "c:\\somedir\\errorlog.txt";
```

Regardless of the settings, the last error is always posted to C:\[user]\AppData\Roaming\ TiffDLL200\error.txt file.

Please also read the separate 'Errors' section below.

Error Messages

By default, error messages are displayed if an error is encountered. You may want to turn off the error messaging with the following line of code:

```
obj.init.ErrorMessage_off = true;
```

Errors are posted to C:\[user]\AppData\Roaming\ TiffDLL200\error.txt file.
If you do not want errors posted to the error log, set the following code:

```
obj. init.ErrorLog_off = true;
```

Please also read the separate 'Errors' section below.

Format (GraphicsFormat)

The extension name of the output file will control the graphics format of the file. For example, an output file named 'somefile.png' creates a PNG file.

If the extension name of the output file is non-specific, you can specify the graphics format in the *obj.init*. Format property; otherwise the file will be saved in the Tiff format. Color depth, Tiff compressions and JPEG quality can also be specified in the Format property. If not specified, the optimum of 8-bit or 24-bit color depth will be used.

If there is a conflict between the extension name and the GraphicsFormat, an error is generated, except that Tiff format has priority, i.e. if either the extension or the GraphicsFormat refers to Tiff, a Tiff file is created.

If you wish to apply the color depth and/or the Tiff compression to only some pages of a multi-page Tiff, use the init.Pages_change property.

If the output file has a valid file extension (.tif, .bmp, .png, jpg, gif) you should not specify the GraphicsFormat.
The available formatting codes are:

```
obj.init.Format.TiffCompression = TiffDLL200.TiffCompression.LZW_mono_or_color;
obj.init.Format.TiffCompression = TiffDLL200.TiffCompression. CCITT4_Group4_mono;
obj.init.Format.TiffCompression = TiffDLL200.TiffCompression. CCITT4_Group4_mono;
obj.init.Format.TiffCompression = TiffDLL200.TiffCompression.Uncompressed;

obj.init.Format.Colordepth = TiffDLL200.Colordepth.Mono_Black;
obj.init.Format.Colordepth = TiffDLL200.Colordepth. Color_8bit;
obj.init.Format.Colordepth = TiffDLL200.Colordepth. Color_TrueColor_24bit;
obj.init.Format.Colordepth = TiffDLL200.Colordepth.Mono_Raw; //see below

obj.init.Format.Graphicsformat = TiffDLL200.GraphicsFormat.TIF;
obj.init.Format.Graphicsformat = TiffDLL200.GraphicsFormat.PNG;
obj.init.Format.Graphicsformat = TiffDLL200.GraphicsFormat.BMP;
obj.init.Format.Graphicsformat = TiffDLL200.GraphicsFormat.JPG;
obj.init.Format.Graphicsformat = TiffDLL200.GraphicsFormat.GIF;
obj.init.Format.Graphicsformat = TiffDLL200.GraphicsFormat.PNG;

obj.init.Format.JPEGQuality = TiffDLL200.JQuality.Q60;  //etc., in increments of 10
```

Colordepth.Mono_Raw creates a monochrome Tiff image without dithering. Pixels below the 50 color brightness are rendered white; pixels above the threshold are rendered black. This option creates small file sizes and is useful for documents, but not for photo images. To adjust the threshold, use the Brightness option.

Tiff compression: If there are no edits to the Tiff pages, perhaps you just want to re-arrange and change the page sequence, the output will generally inherit the Tiff compression of the source file. If there are edits to at least one page, the output will be re-built. Unless specifically specified with the TiffCompression option, the following rules apply. Monochrome pages will be saved in CCITT4 (Group 4) format, color pages will be saved in the LZW format. JPEG-compressed format is not supported for output.

```
obj.init. Format.Graphicsformat = TiffDLL200.GraphicsFormat.PNG; //etc.
obj.init. Format.TiffCompression =
TiffDLL200.TiffCompression.Uncompressed; //etc.
obj.init. Format.Colordepth = TiffDLL200.Colordepth.Mono_Black; //etc.
obj.init. Format.JPEGQuality = TiffDLL200.JQuality.Q60; //etc.
```

run() function

The `run()` executes the conversion. After executing the run() you should check the return code to ensure that the process was successful, and close the object.

```
//.....
```

```
int err = obj.run();
if (err == 0){
System.Windows.Forms.MessageBox.Show(err.ToString() + " " +
obj.info.lasterror); }
obj = null;
System.GC.Collect();
```

A separate `merge()` function is also available; see section below.

Deployment

The TiffDLL200 file must be included with your deployment. The file must be placed in the same folder as your application executable. Do not distribute the help file and do not divulge the license code. See also Copyright sections below.

Edit and Markup Options

Text Annotation

```
obj.op.Annotation.Text = "Hello World";
//obj.op.Annotation.FixedLocation = TiffDLL200.FixedLocation.BottomCenter; //etc.
//obj.op.Annotation.Location.X = 1.25f;
//obj.op.Annotation.Location.Y = 1.25f;
//obj.op.Annotation.Alignment = TiffDLL200.AlignmentText.Center; //etc.
//obj.op.Annotation.Font = new System.Drawing.Font("Arial", 12);
//obj.op.Annotation.ColorText = System.Drawing.Color.Red;
//obj.op.Annotation.Framed = true;
//obj.op.Annotation.ColorBackground = System.Drawing.Color.Yellow;
```

For multiple annotations, use the MultiLine-Annotation option; see separate section below.

There are eight (8) settings for Fixed Location, including vertical text. The default is bottom-center. Use IntelliSense for a listing.

Specify any font. The default is Times New Roman 11. Font attributes can be specified with code such as:

```
new System.Drawing.Font("Arial", 12,System.Drawing.FontStyle.Bold |
System.Drawing.FontStyle.Italic)
```

(VB.NET: `System.Drawing.FontStyle.Bold Or System.Drawing.FontStyle.Italic`)

Text default color is black. Change to any color. For transparent color use code such as:

```
System.Drawing.Color.FromArgb(128,Color.Red)// Opacity value between 0 and 255
```

Placeholders:

- To insert an automatically incrementing number use the [+nnn] placeholder. Example: obj.op.Annotation.Text = "Number [+0001]"; will insert text of "Number 0001", incrementing to "Number 0002", "Number 0003', etc.
- To insert the current page use the placeholder [*p]; for number of pages use [*pp], p in lower case. Example: Page [*p] of [*pp].

For the background color to be effective, Opaque must be set to 'true'.

Watermark

```
obj.op.Watermark.Text = "HELLO";
//obj.op.Watermark.Type = TiffDLL200.WatermarkType.StandardCenterMedium; // etc
```

There are 18 options (style, size, location). The default setting is Standard-Center-Solid.

Resolution

```
obj.op.Resolution.Horizontal = 100;
obj.op.Resolution.Vertical = 100;
//obj.op.Resolution.Resample = true;
//obj.op.Resolution.Parity = TiffDLL200.Parity.Parity_8; //etc.
```

If Resample is set to 'true', pixels are added or removed so that the image is displayed at the original size. By default, the image is not re-sampled.

There are several Parity settings. Parity causes the width and height of the bitmap to be of a number that is divisible by the parity value.

Rotation

```
obj.op.Rotation = TiffDLL200.Rotation.Clockwise90; //etc.
```

The image can be rotated at any right-angle: 90, 180, 270. Conditional rotation options are also available, for example, rotate if width is larger than height, etc.

Flip

```
obj.op.Flip = TiffDLL200.FlipDirection.Horizontal; //etc
```

Deskew

```
obj.op.Deskew = true;
```

Invert (Negative)

```
obj.op.Invert = true;
```

Crop

```
obj.op.Crop.RectangleF.X = 1.25f;
obj.op.Crop.RectangleF.Y = 1.25f;
obj.op.Crop.RectangleF.Width = 5.5f;
obj.op.Crop.RectangleF.Height = 5.5f;
```

Shift

```
obj.op.Shift.PointF.X = 1.25f;
obj.op.Shift.PointF.Y = 1.25f;
```

Use negative values to shift lift or shift up.

Borders Cleanup

```
obj.op.Borders.Left = 0.75F;
//obj.op.Borders.Top = 0.75F;
//obj.op.Borders.Right = 0.75F;
//obj.op.Borders.Bottom = 0.75F;
```

The Border Cleanup acts like an 'inverted crop'; the area inside the specified borders is kept unchanged, while the outside area is 'whited' out. Border cleanup is an easy way to remove unwanted marks in the border areas. You can specify one or more of the borders. All values must be positive.

Resize Image

```
obj.op.Resize.SizeF.Width = 8.5F;
//obj.op.Resize.SizeF.Height = 8.5F;
//obj.op.Resize.AdjustResolution = true;
//obj.op.Resize.asPixels = true;
//obj.op.Resize.Relative = true;
```

Specify the new width or height, or both. If only one dimension is given the other dimension will be at the original width/height ratio. Instead of the new width or height, choose the 'Relative' option and specify the increase in size (minus value for decrease).

Resize Canvas

```
obj.op.Canvas.SizeF.Width = 12F;
obj.op.Canvas.SizeF.Height = 12F;
obj.op.Canvas.Alignment = TiffDLL200.AlignmentCanvas.Align_NorthEast; // etc.
//obj.op.Canvas.asPixels = true;
//obj.op.Canvas.Relative = true;
```

The canvas is the full dimension of the page. Changing the canvas size will not affect the size of the image; it just widens or shortens the borders. Specify the new width or height of the canvas,

or both. If only one dimension is given the other dimension will be original size of the image. Instead of the full width or height, choose the 'Relative' option and specify the increase in size.

If the given canvas dimensions are smaller than the size of the image, the dimensions of the image will be used.

Image Insert

```
obj.op.Image.File = "c:\\aaa\\matterhorn.jpg";
obj.op.Image.PointF.X = 1F;
obj.op.Image.PointF.Y = 1F;
//obj.op.Image.SizeF.Width = 3.5F;
//obj.op.Image.SizeF.Height = 2.5F;
//obj.op.Image.Opacity = TiffDLL200.Opacity.Opacity50;  //etc
//obj.op.Image.MakeWhiteTransparent = true;
```

The inserted image is displayed at its size, adjusted for the resolution.
If only the width or the height is given, the other dimension is set to scale.

256-color images cannot be used in conjunction with a transparency setting; use True Color or monochrome images.

Redaction

```
obj.op.Redaction.RectangleF.X = 1.5F;
obj.op.Redaction.RectangleF.Y = 1.5F;
obj.op.Redaction.RectangleF.Width = 2F;
obj.op.Redaction.RectangleF.Height =0.5F;
//obj.op.Redaction.OtherColor = System.Drawing.Color.White;
```

The default color is black.

Highlighting

```
obj.op.Highlight.RectangleF.X = 1.5F;
obj.op.Highlight.RectangleF.Y = 3.5F;
obj.op.Highlight.RectangleF.Width = 2F;
obj.op.Highlight.RectangleF.Height = 0.5F;
//obj.op.Highlight.Color = System.Drawing.Color.Orange;
```

The default color is yellow.

Brightness

```
obj.op.Brightness = 1.5F;
```

The value must be between 0.1 and 4.0. A value below 1.0 brightens the image; a value above 1.0 darkens the image. This option is useful if you want to convert a color image to

monochrome and want to render a below-threshold color to black, or vice versa. With the Brightness option , the Annotations option is disabled.

Color Flooding

```
obj.op.Flooding.Point1.X = 0.05F;
obj.op.Flooding.Point1.Y = 0.05F;
//obj.op.Flooding.Point2.X = -0.05F;
//obj.op.Flooding.Point2.Y = 0.05F;
//obj.op.Flooding.Point3.X = -0.05F;
//obj.op.Flooding.Point3.Y = -0.05F;
//obj.op.Flooding.Point4.X = 0.05F;
//obj.op.Flooding.Point4.Y = -0.05F;
//obj.op.Flooding.FloodColor = System.Drawing.Color.Black;
```

The Color Flooding option floods the contiguous same-color area of the chosen pixel. The function is used mainly to white-out black border areas. The code above samples the pixel in each corner and removes the black areas.

You can specify up to four (4) points.
The default color is white.

Multi-Line Annotation

```
obj.Specs[0] = "1.0;1.0;0;0;0;Some text line 1";
obj.Specs[1] = "1.0;2.0;0;0;0;Some text line 2";
obj.Specs[2] = "1.0;3.0;1;0;0;Some text line 3";
obj.Specs[3] = "1.0;4.0;2;1;1;Some text line 4";
// up to 100 entries
obj.op.MultilineAnnotation.Font1 = new Font("Arial", 12, FontStyle.Bold);
obj.op.MultilineAnnotation.Font2 = new Font("Arial", 12, FontStyle.Underline);
```

The Spec() string must have five values, each separated by a semi-colon:

1. Left position
2. Top position (for bottom, use minus value)
3. Font (optional). Referenced font must be created with Font1, Font2 or Font3. Default font is Times New Roman 11.
4. Color, if not black: 1 = Red, 2 = Green, 3 = Blue, 4 = White.
5. Text alignment: 1 = Centered, 2 = Right=adjusted.

There is a maximum of 100 entries.
Up to three (3) fonts can be specified.

Pagination (exclude, re-arrange pages)

To re-arrange pages, or to exclude pages, use the `info.Pages_include` property, see separate paragraph above.

Example to re-order the pages:
`info.Pages_include = "1-3,8,7,10-12";`

Merge Files

A separate function is available for merging Tiff files. The function runs separately from the run() function. The merge function does not itself have options for markups, etc. Only Tiff files can be merged. There are two methods: a) merging specified files, and b) merging the files of a folder.

Sample code:

```
TiffDLL200.TiffDLL obj = new TiffDLL200.TiffDLL();
// other initial settings
obj.op.Merge.File_out="C:\\somefile\\merged.tif";
obj.Mergefiles[0] = "C:\\somedir\\somefile1.tif";
obj.Mergefiles[1] = "C:\\somedir\\somefile2.tif";
obj.Mergefiles[2] = "C:\\somedir\\somefile3.tif";

//Alternative: use a directory instead of a list of files
//obj.op.Merge.Directory_in = "C:\\bbb";
//obj.op.Merge.Directory_Filter = "*.tif";

int err = obj.merge();
System.Windows.Forms.MessageBox.Show(err.ToString());
obj = null;
GC.Collect();
```

The default filter setting for the directory is *.* meaning that all files in the folder will be lined up for the merge. To include only certain files, use the filter property, for example *.tif, or abc*.tif (file names starting with 'abc'. Note, only Tiff files can be merged. Non-Tiff files need to be converted first in a separate operation.

Up to 1000 files can be specified. If you have more files, use the Folder method.

Always check the output to ensure that all files have been included and are correctly rendered.

Image Information

Use the following code to obtain image information, such as number of pages, width, height, resolutions, color depths, etc.

```
Bitmap bm = new Bitmap("c:\\aaa\\5pages.tif");
System.Drawing.Imaging.FrameDimension oFDimension;
oFDimension = new
System.Drawing.Imaging.FrameDimension(bm.FrameDimensionsList[0]);
int nrpages = bm.GetFrameCount(oFDimension);
bm.SelectActiveFrame(oFDimension, 0);  //zero-based,  0 = page 1
float horizontalresolution = bm.HorizontalResolution; //etc
int colordepth = Image.GetPixelFormatSize(bm.PixelFormat);
System.Windows.Forms.MessageBox.Show(nrpages.ToString());
System.Windows.Forms.MessageBox.Show(horizontalresolution.ToString());
System.Windows.Forms.MessageBox.Show(colordepth.ToString());
bm.Dispose();
bm = null;
```

While in a process, you can obtain the number of pages of a multi-page Tiff with the following code:

```
int numberofpages = obj.info.pages;
```

Printing

TiffDLL200 itself has no options for printing. Printing can easily be programmed in Visual Studio. A code sample for printing (VB and C#) can be downloaded from:

`www.informatik.com/TIFFDLL/TiffDLL200_SampleCode_Printing.txt`

PDF

TiffDLL200 itself does not support the PDF format, both read and write. DotNET sample code for a simple TIFF-to-PDF conversion, using the LibTiff library, with the necessary system files, can be downloaded from the link below. The Informatik technical support does not cover the LibTiff library.

`www.informatik.com/TIFFDLL/Tiff2PDF_Sample.zip`

Errors

The last error is posted to C:\[user]\AppData\Roaming\ TiffDLL200\error.txt file.

If you do not want the error posted to the error log, set the following:

```
obj. init.ErrorLog_off = true;
```

The functions return an error code and short description of the error. By default, the system displays a message with the error code and a short description. You can disable the error message with the following code.

```
obj.init.ErrorMessage_off= true;
```

The functions return an exit code: 0 if success or an error number if failed. It is recommended that you to check the error code and to handle any exceptions.

```
//……
int err = obj.run();
if (err == 0){
System.Windows.Forms.MessageBox.Show(err.ToString() + " " +
obj.info.lasterror); }
```

Most errors are caused by unsupported graphics formats and syntax errors. An error also is generated if you try to overwrite an existing file, unless you allow overwrites:

```
obj.init.OverwriteFile = true;
```

You can also have the system post all errors to an error log. To enable the error log, add the following code:

```
obj.init.ErrorLogfile = "c:\\somedir\\errorlog.txt”;
```

Technical Support

Before you contact Informatik Inc for support, please check the Frequently-Asked-Questions via www.informatik.com/faq/tiffdll200.html

The technical support contact information is shown in http://www.informatik.com/support.html.

License, Warranty, Disclaimer

Please read the terms carefully before installing and using the software, as such conduct will indicate your acceptance of all of the terms of this license agreement. If you do not agree with the terms, the software cannot be licensed to you and you must un-install and return the software to Informatik Inc, or its supplier or distributor.

This License Agreement is a legal agreement between Informatik Inc. ("Licensor"), a Pennsylvania Corporation, and you, the user ("Licensee"), and is effective the date Licensee installs the software.

This Agreement covers all materials associated with the TiffDLL200 software, including, without limitation, the downloadable software product, online documentation, and any additional supporting electronic files (herein, the "Software").

The evaluation version may be used for 30 days after installation. It is unlawful to use the software after the 30 day evaluation period without licensing the software and paying the license fees. If a license is not obtained before the expiration of the 30 day evaluation period, the Software must be un-installed and destroyed.

1. GRANT OF LICENSE

Licensor hereby grants to you, and you accept, a nonexclusive license to use the Software according to the following condition (based on license type):

The site license allows you to install the Software on an unlimited number of computers at one location of the Licensee. Applications developed with the Software may be deployed to an unlimited number of end-users of Licensee (personal computers or servers, but not web servers). For deployment of to non-licensees, please inquire. The command-line version of the Software follows the same terms. You may not distribute the user guides and you must keep the registration codes confidential. The TiffDLL200 functions must be a minor part of your application and it must be a desktop application. You may not develop tools that can be used to develop or be integrated with other applications. The Software may not be installed on a web server without a special license.

If you deploy your applications that use the Software, your conditions and terms must reflect the terms of this agreement relating to the use of this Software, especially as to clearly stating that the Licensor is not responsible for the Software directly or indirectly and does not give any warranties or guarantees whatsoever.

2. LICENSOR'S RIGHTS

Licensee acknowledges and agrees that the Software is proprietary to Licensor and protected under international copyright law. Licensee further acknowledges and agrees that all right, title, and interests in and to the Software, including associated intellectual property rights, are and shall remain with Licensor. The License Agreement does not convey to Licensee an interest in or to the Software, but only a limited right of use that may be revoked in accordance with the terms of this License Agreement.

3. OTHER RESTRICTIONS

Licensee agrees to make no more than one (1) back-up copy of the Software. Licensee agrees not to assign, sublicense, transfer, pledge, lease, rent, or share the rights assigned under this License Agreement. Licensee agrees not to reverse assemble, reverse compile, or otherwise translate the Software.

4. TERM

This License Agreement is effective when Licensee installs the Software and shall terminate only if the terms of this License Agreement are broken. Licensee agrees to destroy the Software upon termination of this License Agreement.

5. NO WARRANTY; LIMITATION OF LIABILITY

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL LICENSOR BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

6. GOVERNING LAW

This License Agreement shall be construed and governed in accordance with the laws of Pennsylvania.

7. SEVERABILITY

Should any court of competent jurisdiction declare any term of this License Agreement void or unenforceable, such declaration will have no effect on the remaining terms hereof.

8. NO WAIVER

The failure of either party to enforce any rights granted hereunder or to take action against the other party in the event of any breach hereunder shall not be deemed a waiver by that party as to subsequent enforcement of rights or subsequent actions in the event of future breaches.

Copyright and Trademarks