Software License Agreement

DOCX to Image Converter

For .NET

Vesion 15

2008-2024

ALL RIGHTS RESERVED BY

SUB SYSTEMS, INC.

3200 Maysilee Street

Austin, TX 78728

512-733-2525

Software License Agreement

The Software is protected by copyright laws and international copyright treaties, as well as other intellectual property laws and treaties. The Software is licensed, not sold. This LICENSE AGREEMENT grants you the following rights:

A. This product is licensed per developer basis only. Each developer working with this package needs to purchase a separate license.

B. The purchaser has the right to modify and link the DLL functions into their application. Such an application is free of distribution royalties with these conditions: the target application is not a stand-alone DOCX to Image Converter; the target application uses this product for one operating system platform only; and the source code (or part) of the editor is not distributed in any form.

C. The DESKTOP LICENSE allows for the desktop application development. Your desktop application using this product can be distributed royalty-free. Each desktop license allows one developer to use this product on up to two development computers. A developer must purchase additional licenses to use the product on more than two development computers.

D. The SERVER LICENSE allows for the server application development. The server licenses must be purchased separately when using this product in a server application. Additionally, the product is licensed per developer basis. Only an UNLIMITED SERVER LICENSE allows for royalty-free distribution of your server applications using this product.

E. ENTERPRISE LICENSE: The large corporations with revenue more than $50 million and large government entities must purchase an Enterprise License. An Enterprise license is also applicable if any target customer of your product using the Software have revenue more than $500 million. Please contact us at info@subsystems.com for a quote for an Enterprise License.

F. Your license rights under this LICENSE AGREEMENT are non-exclusive. All rights not expressly granted herein are reserved by Licensor.

G. You may not sell, transfer or convey the software license to any third party without Licensor's prior express written consent.

H. The license remains valid for 12 months after the issue date. The subsequent year license renewal cost is discounted by 20 percent from the license acquisition cost. The

license includes standard technical support, patches and new releases.

I. You may not disable, deactivate or remove any license enforcement mechanism used by the software.

This software is designed keeping the safety and the reliability concerns as the main considerations. Every effort has been made to make the product reliable and error free. However, Sub Systems, Inc. makes no warranties against any damage, direct or indirect, resulting from the use of the software or the manual and can not be held responsible for the same. The product is provided 'as is' without warranty of any kind, either expressed or implied, including but not limited to the implied warranties of suitability for a particular purpose. The buyer assumes the entire risk of any damage caused by this software. In no event shall Sub Systems, Inc. be liable for damage of any kind, loss of data, loss of profits, interruption of business or other financial losses arising directly or indirectly from the use of this product. Any liability of Sub Systems will be exclusively limited to refund of purchase price.

Sub Systems, Inc. offers a 30 day money back guarantee with the product. Must call for an RMA number before returning the product.

Getting Started

This chapter describes the contents of the software diskettes and provides a step by step process of incorporating DOCX to Image Converter into your application.

Files

The package contains the din.dll and TERN files necessary to incorporate this product into your application.

The package also includes a set of files to construct a demo program. The demo program shows by example the process of linking the DLL to your program.

DLL Demo Files:

The following demo files are included in the c_demo.zip file.

demo.cs   Source code for the demo program

demo.exe	Executable demo program
demo.csproj	The project file to compile the demo.
AssemblyInfo.cs	Assembly information file

Visual Basic Interface and Demo Files:

Form1.vb	vb source file
dmo_vbn.vbproject	The project file for the visual basic demo program.
AssemblyInfo.vb	Assembly information file for the demo program.

License Key

Your License Key and License number are e-mailed to you after your order is processed.
You would set the license information using the DisSetLicenseInfo static function. This
should be preferably done before creating the Din object to avoid pop-up nag screens.

int DisSetLicnseInfo(String LicenseKey, String LicenseNumber, String CompanyName);

LicenseKey:	Your license key is available in the product delivery email sent to you upon the purchase of the product. It consists of a string in the form of "xxxxx-yyyyy-zzzzz".
LicenseNumber:	Your license number is also available in the product delivery email. The license number string starts with a "srab" or "smo" prefix.
CompanyName:	Your company name as specified in your order.

Return Value: This method returns 0 when successful. A non-zero return value indicates
an error condition. Here are the possible return values:

0	License application successful.
1	Invalid License Key.
2	Invalid License Number.
3	Ran out of available licenses. Please consider purchasing additional licenses.

Example:

result=Din.DisSetLicenseInfo("xxxxx-yyyyy-zzzzz","srabnnnnn-n","Your Company Name")

Replace the 'xxxxx-yyyyy-zzzzz' by your license key, replace "srabnnnnn-n" with your license number, and "Your Company Name" with your company name as specified in your order.

Note: *DisSetLicenseInfo method should be called only once at the beginning of your application. Calling this method for each conversion would degrade the conversion performance.*

Sample Conversion Code

Please ensure that din.dll and tesn31.dll files are available in the project directory. Set the reference for din.dll in your project. The tesn31.dll file is referenced indirectly by din.dll.

Now set the namespace for the product:

```
using SubSystems.DI;              // C# example
Imports SubSystems.DI            ' VB Example
```

Now set the product license key and create an DIN type object:

Din.DisSetLicenseInfo("xxxxx-yyyyy-zzzzz", "srabnnnnn-n","my company name")

```
Din di = new Din();  // C# example
dim di as Din          ' VB example


di.ProjectFolder = this.MapPath("");
```

Now use one of the following calls to convert from DOCX to an Image:

1. Convert an DOCX file to an Image file

```
// load DOCX file to determine the number of pages in the
// document

PageCount=di.DisLoadFile("test.docx");

di.ImageType=Din.PICT_JPG; // set output image type to
                           // jpg
Image image=di.DisGetPageImage(1); // get Image for the
                                   // first page
```

```
if (image!=null) {

   result=di.DisImageToFile(image,"test.jpg");

   image.Dispose();

}
```

2. Convert an DOCX string to an Image

```
// load DOCX file to determine the number of pages in the
// document

PageCount=di.DisLoadBuffer(DocxBytes);

di.ImageType=Din.PICT_JPG; // set output image type to
                            // jpg
Image image=di.DisGetPageImage(1); // get Image for the
                                    // first page
```

Control Methods

These methods allow you to convert from DOCX to an Image. Please set the namespace for the Din class before using these methods:

```
using SubSystems.DI;              // C# example

Imports SubSystems.DI             ' VB Example
```

In This Chapter
DisLoadBuffer
DisLoadFile
DisFileToBytes
DisGetImageType
DisGetLastMessage
DisGetPageImage
DisImageToBytes
DisImageToFile
DisImageToStr

DisLoadBuffer

Load DOCX or plain text string and determine the number of pages in the DOCX document.

int DisLoadBuffer(DocxBytes)

byte[] DocxBytes;   // Input byte array containing DOCX formatted document.

Return value: This function returns the number of pages in the DOCX document. A value of zero indicates an error condition.

Examples:

Load an DOCX byte array into the converter

```
di.ProjectFolder = this.MapPath("");


PageCount=di.DisLoadBuffer(DocxBytes);
```


DisLoadFile

Load an DOCX or plain text file and determine the number of pages in the DOCX document.

int DisLoadFile(InFile)

string InFile;   // Input file containing DOCX document or plain text.

Return value: This function returns the number of pages in the DOCX document. A value of zero indicates an error condition.

Examples:

Load an DOCX file into the convertere.

```
PageCount=di.DisLoadFile("test.docx")
```

DisFileToBytes

Read a docx file into a byte array.

byte[] DisFileToBytes(DocxFile)

String DocxFile;   // Input docx file name.

Return value: This function returns a byte array from the given file. This method can be used to obtain the byte array to supply to the DisLoadBuffer method.

A null return values indicates an error.

Example:

```
InData = di.DisFileToBytes(InFile)


PageCount = di.DisLoadBuffer(InData)
```

DisGetImageType

Return the picture type constant corresponding to the extension of the given file.

int DisGetImageType(InFile)

string InFile;   // Picture file name

Return value: This function returns one of the PICT_ constant enumerations corresponding to the extension of the file name provided using the first parameter.

This information can be used to set the ImageType property. The ImageType property determines the type of image returned by the subsequent call to the DisGetPageImage method.

Examples:
```
di.ImageType=di.DisGetImageType("test.jpg")
```

```
        Image image=di.DisGetPageImage(1);
```

DisGetLastMessage

Get the last message.

int DisGetLastMessage(DisMessage, DebugMessage);

string DisMessage;   // Returns the default user message text in English

string DebugMsg;   // Returns any debug message associated with the last
                          message. The debug message need not be displayed to
                          the user.

Return Value: This function returns the last message generated by the editor. This value is valid only if saving of the messages is enabled by setting the DIFLAG_RETURN_MSG_ID flag. This flag is set using the DisSetFlags method.

DisGetPageImage

Return the image for the requested page number for the currently loaded DOCX document.

Image DisGetPageImage(PageNo)

int PageNo;   // Page number. This value should be between 1
                       and the PageCount for the currently loaded DOCX
                       document.

Return value: This function returns the Image object for the requested page. A null value indicates an error condition.

Since this method returns an object of the Image class, the System.Drawing namespace must be included in your source module to use this method.

Examples:
```
        PageCount=di.DisLoadFile("test.docx");


        di.ImageType=Din.PICT_JPG;
```

```
      Image image=di.DisGetPageImage(1);
```

DisImageToBytes

Return the the requested image data in a byte array.

byte[] DisImageToBytes(image)

Image image   // Image object to save.

Return value: This method returns a byte array containing requested image data. A
null value indicates an error condition.

This byte array returned by this method can be assigned to a web Response object
to deliver to the client machine.

Examples:

```
  // write to response object

  Response.Clear();
  Response.Charset = "";
  Response.ContentType = "application/jpg";

  string strFileName = "test.jpg";
  Response.AddHeader("Content-Disposition",
                   "inline;filename=" + strFileName);

  Din hi = new Din();
  di.InWebServer = true;
  di.ProjectFolder = this.MapPath("");
  di.ImageType = Din.PICT_JPG;

  int PageCount = di.DisLoadBuffer(DocxBytes);
  if (PageCount > 0)
  {
     System.Drawing.Image image = di.DisGetPageImage(1);
     byte[] bytes = di.DisImageToBytes(image);
     image.Dispose();
```

```
        Response.BinaryWrite(bytes);
    }

    Response.Flush();
    Response.Close();
    Response.End();
```

DisImageToFile

Save the current image to the requested file name.

bool DisImageToFile(image, OutFile)

Image image   // Image object to save.

String OutFile   // Output image file name

Return value: This method returns True when successful.

Examples:

```
    PageCount=di.DisLoadFile("test.docx");

    di.ImageType=Din.PICT_JPG;

    Image image=di.DisGetPageImage(1);

    result=di.DisImageToFile(image,"test.jpg");
    image.Dispose();
```

DisImageToStr

Return the the requested image data in a string.

string DisImageToBytes(image)

Image image   // Image object to save.

Return value: This method returns a string containing requested image data. A null value indicates an error condition.

Examples:

```
PageCount=di.DisLoadFile("test.docx");

di.ImageType=Din.PICT_JPG;

Image image=di.DisGetPageImage(1);

string str=di.DisImageToStr(image);
image.Dispose();
```


DisResetLastMessage

Reset the last editor message.

bool DisResetLastMessage()

Description: This function can be called before calling any other function to reset the last error message.

Return Value: The function returns TRUE when successful.

See Also
DisGetLastMessage
DisSetFlags

DisSetFlags

Set certain flags or retrieve the values of the flags.

int DisSetFlags(set, flags)

bool set;   // TRUE to set the given flags, FALSE to reset the given flags

int flags;   // Flags (bits) to set or reset. Currently, the following flag values are available:

DIFLAG_RETURN_MSG_ID   Do not display the error messages. Save the error code to be later retrieved using the DisGetLastMessage function.

Return value: This function returns the new value of all the flags. Call this function with the 'flags' parameter set to zero to retrieve flag values without modifying it.

DisSetImageSize

Set the image size.

bool DisSetImageSize(ImageWidthTwips, ImageHeightTwips)

int ImageWidthTwips;   The image width in twips units (1440 twips = 1 inch).

int ImageHeightTwips;   The image height in twips units (1440 twips = 1 inch).

Return Value: The function returns TRUE when successful.

Comment: This function is used to override the default image size when converting an DOCX document to the PDF format. This function should be called before calling the DisConvertFile or DisConvertBuffer if you wish override the image size. The default image size is derived from the paper-size specification embedded in the docx file.

This method works the same as the DisSetPaperSize method when called with the 'size' parameter set to 0.

DisSetPageMargin

Set the page margins for Image output.

bool DisSetPageMargin(left, right, top, bottom)

int left;   Left margin in twip units (1440 twips = 1 inch)

int right;   Right margin in twip units

int top;   Top margin in twip units

int bottom   Bottom margin in twip units

Return Value: The function returns TRUE when successful.

Comment: This function is used to override the default page margins when converting an DOCX document to the Image. This function should be called before calling the DisLoadFile or DisLoadBuffer methods if you wish override the page margin values.

DisSetPaperOrient

Set the page orientation for Image output.

bool DisSetPaperOrient(id, IsPortrait)

bool IsPortrait	Set to true to set to portrait orientation. Otherwise set to false.

Return Value: The function returns TRUE when successful.

Comment: This function is used to override the default portrait orientation when converting an DOCX document to the PDF format. This function should be called before calling the DisLoadFile or DisLoadBuffer methods if you wish override the paper orientation.

DisSetPaperSize

Set the page size for Image output.

bool DisSetPaperSize(kind, PageWidth, PageHeight)

PaperKind kind;	Use one of the PaperKind enumerations defined by .NET
int PageWidth;	The page width in twips units (1440 twips = 1 inch). This argument is used only if kind is set to PaperKind.Custom.
int PageHeight;	The page height in twips units (1440 twips = 1 inch). This argument is used only if kind is set to PaperKind.Custom.

Return Value: This method returns TRUE when successful.

Comment: This method is used to override the default letter size paper when converting an DOCX document to the PDF format. This method should be called before calling the

DisLoadFile or DisLoadBuffer methods if you wish override the paper size.

Control Properties

The control properties can be before the conversion to affect the Image output. The control supports the following properties:

InWebServer

This property should be set to True when this control is used in a web server. When this property is set to True, the control suppress the display of any dialog and message boxes.

ProjectFolder

Set this property to the folder containing your project, such as c:\inetpub\wwwroot\MyProject. This information helps the converter locate the images which use relative path. It is also used for creating any temporary files.

ImageType

Use this property to request a particular type of image (default is Bitmap file):

PICT_BMP	Bitmap image
PICT_EMF	Enhanced metafile
PICT_TIF	Tiff image
PICT_JPG	Jpeg image
PICT_PNG	PNG image
PICT_GIF	GIF image
PICT_ICO	Icon image
PICT_EXF	Exif Image

SizePercent

Use this property to change the size of the output image. The default value for this property is 100. You can specify a value small than 100 to obtain a smaller image. Similarly you can specify a value greater than 100 to obtain a larger image.

ImageRes

Use this property to specify the resolution of an image. The default value is 96 dpi.

MetaRes

Use this property to specify the resolution of a metafile image. The default value is 300.

ShrinkToFit

Eliminate the ending white spaces to shrink the image height. This property is only effective for one page DOCX documents.

AutoWidth

Adjust the image width to fit the contents.