



PDFTron PDF2Image SDK™
User Manual

Version 4.x

1.2 About This Manual

This manual is intended as a guide to the installation and use of PDF2Image SDK. It is intended for programmers and other users who are familiar with PDF documents, graphic image file creation, graphic file manipulation and general computer processes.

- [Section 1](#) introduces PDF2Image SDK and describes the manual.
- [Section 2](#) explains how to install, register and uninstall PDF2Image SDK.
- [Section 3](#) covers basic use of PDF2Image SDK and integration with third-party applications.
- [Section 4](#) contains the PDF2Image Command String Syntax, FAQ and Usage Examples.
- [Section 5](#) is where you will find all the support information you may require, such as how to report a problem with the software.

3.2 Reporting Progress Messages and Errors

To find out if PDF2Image processing was successful, the application can query the status code returned by PDF2ImageRun().

For example,

```

int ret = PDF2ImageRun(...);
if (ret == PDF2IMAGE_OK) {
    // No errors...
}
else if (ret == PDF2IMAGE_ERR_BADKEY ) {
    // bad license key...
}
else if (ret == PDF2IMAGE_ERR_DIRCREATE ) {
    // Failed to create the output directory
}
else {
    // Other error
}

```

A non-zero value indicates that an error was encountered. You can find the listing for all error code in 'include/pdf2image.h' header.

For more detailed error and message reporting, you can pass a pointer to the custom callback function in the second parameter of PDF2ImageRun(). The last parameter in PDF2ImageRun is a pointer to custom data that you may want to pass to the callback function.

A sample callback function may look as follows:

```

// Using C/C++
char* MyCallback(int mode, char* msg, void* user_data) {
    if (mode == PDF2IMAGE_ERROR) {
        cout << "Error: " << msg << endl;
    }
    else if (mode == PDF2IMAGE_MSG) {
        cout << msg;
    }
    else if (mode == PDF2IMAGE_GETPASS) {
        static string gl_pass;
        cin >> gl_pass;
        return (char*)gl_pass.c_str();
    }
    return 0;
}

' or in VB.NET...
Public Function MyCallback(ByVal mode As Integer, ByVal msg As String,
ByVal user_data As Int32) As Int32
    If mode = PDF2IMAGE_ERROR Then
        Console.WriteLine("Error: {0}", msg)
    ElseIf mode = PDF2IMAGE_MSG Then
        Console.Write("{0}", msg)
    End If
    Return 0
End Function      'MyCallback

```


		(DPI). The higher the DPI, the larger the image. Resolutions larger than 1000 DPI can be achieved by rendering image in tiles or stripes. The default resolution is 92 DPI.
--hres	--hres 100	The width of the output image, in pixels.
--vres	--vres 100	The height of the output image, in pixels.
-a or --pages	Render page 1,3, and 10: -a 1,3,10 Render all even pages: -a even Render pages in the range from 3-11 and page 50: --pages 3-11,50 Render all odd pages and all pages in the range from 100 to the last page: -a odd,100-	Specifies the list of pages to convert. By default, all pages are converted.
-b or --box	-b media	Specifies the page box/region to rasterize. Possible values are: <ul style="list-style-type: none"> ■ media ■ crop ■ trim ■ bleed ■ art <p>By default, page crop region will be rasterized.</p>
-c or --clip	-c 216,522,330,600	User definable clip box. By default, the clip region is identical to the current page 'box'.
-r or --rotate	-r 90	Rotates all pages by a given number of degrees, counterclockwise. The allowed values are: <ul style="list-style-type: none"> ■ 0 ■ 90 ■ 180 ■ 270 <p>The default value is 0.</p>
-g or --gray	--gray	Render and export the image in grayscale mode. Sets pixel format to 8 bits per pixel grayscale. By default, the image is rendered and exported in RGB color space.
-k or --cmyk	--cmyk -f tif	Render and export the image in CMYK mode. To export CMYK, the output image format must support CMYK pixel format. An example of image format that supports CMYK is TIFF (e.g. -f tif -k). By default, the image is rendered and exported in RGB color space.
--mono	--mono	Export the rendered image as 1 bit per pixel

		(monochrome) image. If the output format is TIFF, the image will be compressed using G4 CCITT compression algorithm. By default, the image is not dithered. To enable dithering use '--dither' option.
--dither	--dither	Enables dithering when the image is exported in palletized or monochrome mode (e.g. when export format is tif8, png8 or --mono).
--gamma	--gamma 0.3	Sets the gamma factor used for anti-aliased rendering. Typical values are in the range from 0.1 to 3. Gamma correction can be used to improve the quality of anti-aliased image output and can (to some extent) decrease the appearance common anti-aliasing artifacts (such as pixel width lines between polygons).
-q or --quality	-q 100	Compression quality is a number in the range from 1 to 100. Lower numbers usually result in better compression at the expense of image quality. The default setting is 80.
-m or --multipage	--multipage	If the output image format supports multi-page or multi-frame capability, store all output images in one file instead of separate files. Currently, this option is only relevant to TIFF output. By default, images will be saved in separate files.
--printmode		Renders annotations in the print mode. This option can be used to render 'Print Only' annotations and to hide 'Screen Only' annotations.
--noannots		Disables drawing of annotations and forms.
--nosmooth		Disables image smoothing.
--noprompt		Disables any user input. By default, the application will ask for a valid password if the password is incorrect.
-p or --pass	e.g. secret or "my pass"	The password for the input file. Not required if the input document is not secured.
--extension	--extension ".pdf"	The default file extension used to process PDF documents. The default is ".pdf".
-h or --help		Print a listing of available options.
-v or --version		Print the version information.
--verb	--verb 2	Set the verbosity level. Valid parameter values are 0, 1, and 2. The higher number results in more feedback. The default is 1.

