3-Heights™ PDF to Image Converter API

Version 4.1

日本語ユーザー・マニュアル

Contact: pdfsupport@pdf-tools.com

Owner: **PDF Tools AG**
Kasernenstrasse 1
8184 Bachenbülach
Switzerland
www.pdf-tools.com

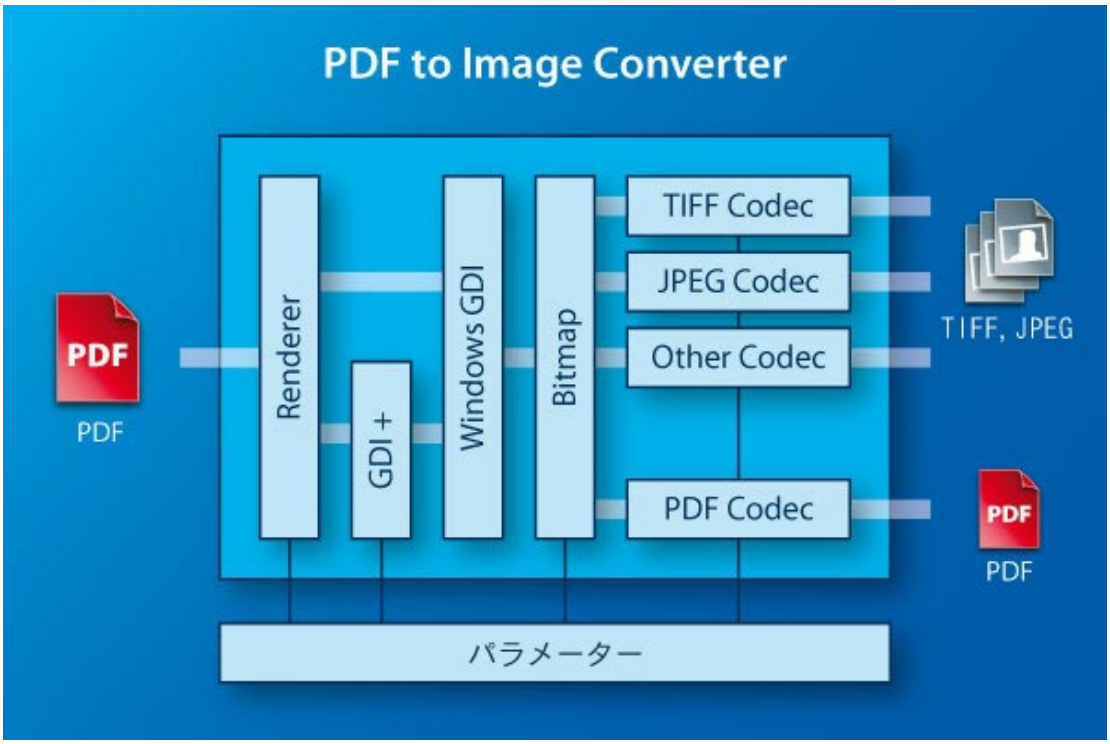
1 目次

2 はじめに

2.1 概要

“3-Heights™ PDF to Image Converter API”は、PDF 文書を単一もしくは複数ページで構成された画像データ(JPEG や TIFF 形式など)に変換します。また、この API は、画像形式のデータで構成された PDF にも変換します。

変換された画像や PDF は、Web 用のデータ、TIFF 画像ベースの DMS(Document Management System)、文書保管、およびワークフローシステム、さらに PDF 文書の保護などに利用できます。この変換 API は、その高速性と優れた品質を特徴としています。



2.2 機能

“3-Heights™ PDF to Image Converter API”は、複数の入力ファイルを統合して、1つまたは複数のファイルを作成します。このとき、色空間と画像サイズは処理プロセスによって自動で定義されます。この変換 API は、様々なスケール変換をサポートするとともに、PNG、TIFF、JBIG2、や JPEG2000 などの画像形式をサポートしています。

特徴

- 単一ページまたは複数ページの画像もしくは、ラスター形式の画像に変換された PDF 文書を生成
- 各ページを個々に変換

- PDF 文書に含まれているテキストやベクター形式の画像をすべてラスター化された画像の PDF に変換（テキストやベクター画像の抽出防止などを目的として）
- PDF ファイルを CCITT ファックスファイルに変換
- ページモードを設定
- ページサイズをポイントまたはピクセル単位で定義
- 回転を設定（元の PDF と同じ方向を利用する・横置きに回転・縦置きに回転）
- 画像の解像度を X と Y 方向に DPI（Dots per Inch）単位で設定
- ディザリング（フロイド-スタインバーグ、ハーフトーン ブロック、ハーフトーン 連続）
- 画像フィルター設定
- 色深度設定
- 色空間設定
- TIFF 形式での圧縮設定
- 非可逆圧縮での品質設定
- ファックスファイルでのビット充填順序を設定
- 最小線幅定義

フォーマット

入力のフォーマット:

- PDF 1.2, 1.3, 1.4, 1.5, 1.6, 1.7
- PDF/A-1, PDF/A-2

出力（変換後）のフォーマット:

- TIFF (Tagged Image File Format)
- JPEG (Joint Photographic Expert Group)
- PNG (Portable Network Graphics)
- GIF (Graphics Interchange Format)
- BMP (Window Bitmap)
- EPS (Encapsulated PostScript)
- JBIG2 (Joint Bi-level Image Experts Group)
- JPEG2000
- Extended JPEG2000
- PBM (Portable Bitmap File Format)

2.3 インターフェース

以下のインターフェースをサポートします。

- C
- Java
- .NET
- COM

2.4 OS（オペレーティング システム）

Windows 2000, XP, 2003, Vista, 2008, Windows 7, 2008-R2 – 32ビットおよび64ビット

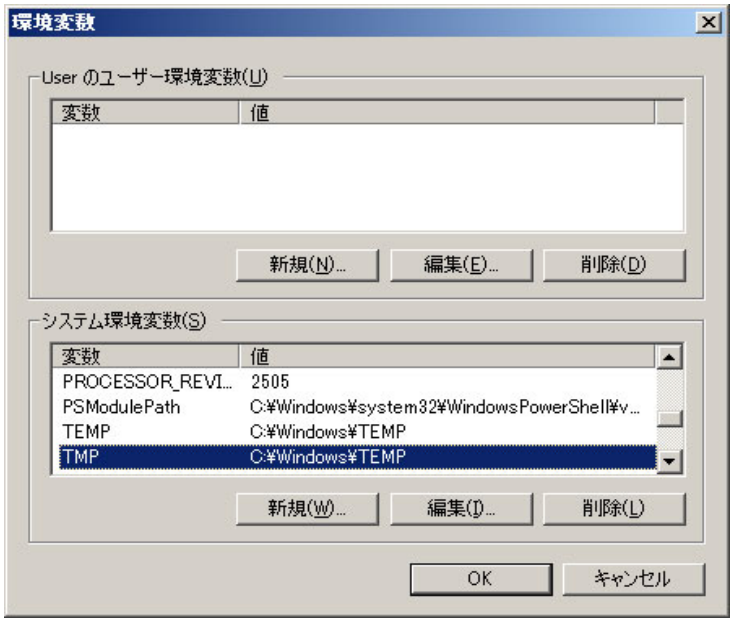
3 インストール

3.1 “PDF to Image Converter API”のインストール

“3-Heights™ PDF to Image Converter API”には、ランタイムの実行イメージファイル、説明書および、ライセンス条項などが含まれています。

1. ZIPまたは、インストール用の製品をwww.pdf-tools.comにある専用アカウントからダウンロードします。
2. ZIP データを解凍または、インストール用のデータを使ってインストールします。
3. ZIP ファイルは、適切なフォルダー（例 *C:\program files\pdf-tools*）に解凍してください。
4. 解凍または、インストールすると以下のようなサブフォルダーが生成されます。
   - *Bin:* ランタイムのバイナリ実行イメージファイルが含まれます。
   - *Bin\Fonts:* 2つの標準フォントとフォントマッピング ファイルが含まれます。2つのフォントは、システムフォントのフォルダー（*%systemroot%\fonts*、例 *C:\Windows\fonts*）にコピーしインストールします。
   - *Doc:* ドキュメントファイルが含まれます。
   - *Include:* C/C++プロジェクトで開発時に利用するヘッダーファイル（.h）が含まれます。
   - *Samples:* 様々なサンプルが含まれます。
5. 2つのシステム環境変数 TEMP と TMP が正しく定義されていることを確認します。これらのディレクトリは、PDF 文書に埋め込まれたフォントを一時的にインストールする際に使用します。

   コントロール パネルから環境設定ダイアログを起動します。



ファイルの概要

以下は、“PDF to Image Converter”に付属している DLL の概要です。

bin\Pdf2ImgOCX.dll	PDF 文書を画像データに変換するために利用する DLL です。
bin\PDf2PdfImgOCX.dll	PDF 文書をラスター化された PDF 文書に変換するために利用する DLL です。
bin\pdcjk.dll	アジアの言語向けに用意されたオプションの DLL です。実行モジュールのパスを使ってロードされます。
*bin*NET.dll*	.NET アセンブリです。 .NET インターフェースを使う場合に必要です。
Fonts\Symbol.ttf *Fonts\ZapfDingbats.ttf*	Symbol フォントと ZapfDingbats フォントは、システムフォントフォルダー（%systemroot%\fonts、例 C:\Windows\fonts）にコピーおよびインストールします。 一般に Windows のシステムフォントには、すでに Symbol フォントが含まれています。ここでインストールする Symbol フォントには、Windows の標準フォントに含まれていない Mac-Apple の文字が含まれています。 Symbol と ZapfDingbats フォントは、多くの PDF 文書に含まれている14の PDF 標準フォントが含まれています。
Fonts\fonts.ini	代替フォントを指定します。このファイルの使用はオプションです。

カラープロファイル

“3-Heights™ PDF”レンダリング エンジンは、RGB 色空間で動作します。他の色空間では、最初に RBG 色空間に変換してからレンダリングします。カラープロファイルが有効でない場合この変換は、カラープロファイルを使うかアルゴリズム的に行うことができます。

カラープロファイルを使った変換では、2つのファイル（*Icc\CMYK.icc* および *Icc\sRGB.icm*）を必要とします。2つのファイルは、*Pdf2ImgOCX.dll* が格納されたフォルダーの直接のサブフォルダーである *Icc* に格納されていなければなりません。

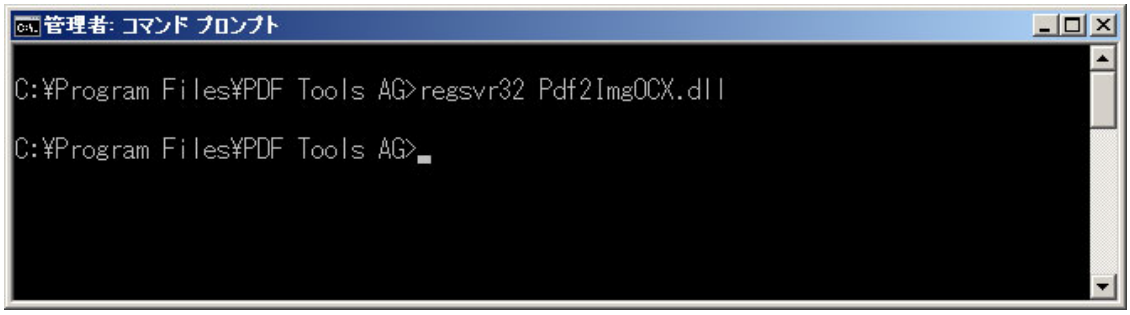
カラープロファイルは、解凍またはインストールされたフォルダーの“*Icc*”フォルダーにあるリンクからダウンロードできます。なお多くの場合 Windows OS では、すでにカラープロファイルが、*%systemroot%\system32\spool\drivers\color* にインストールされていますので、それをコピーして利用することもできます。ただしコピーする場合は、ファイル名を *CMYK.ic* と *sRGB.icm* に変更しなければなりません。

3.2 インターフェース

COMインターフェース

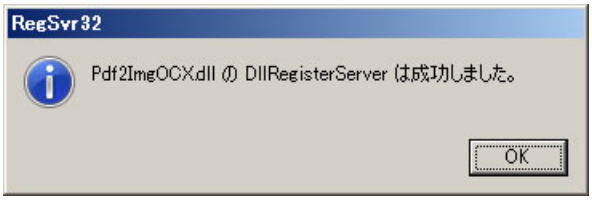
COM アプリケーション プログラムで“3-Heights™ PDF to Image Converter API”を使う場合は、その前に *regsvr32.exe*（Windows OS の System32 ディレクトリにあります。）を使ってそのコンポーネントを登録しなければなりません。

登録するために、コマンド プロンプトを起動します。Window Vista または Windows7 の場合は、管理者権限で起動します。



なお、お使いの Window システムが64ビット版の場合で32ビット版の“3-Heights™ PDF to Image Converter API”を登録する場合は、ディレクトリ WOW64（System32 ではなく）にある regsvr32 を使って登録してください。

登録が成功すると、以下のようなダイアログが表示されます。



これで、インストールは完了です。

なお、ラスター化された画像の PDF（画像データではなく）を生成する場合は上記と同様にして、*Pdf2PdfImgOCX.dll* を（***Pdf2ImgOCX.dll*** に換えて）登録してください。

Javaインターフェース

Java インターフェースを使う場合は、*P2IA.jar* を CLASSPATH に格納してください。さらに、*Pdf2ImgOCX.dll* を環境変数 PATH で示されたフォルダーもしくは java.library.path で示されたフォルダーに格納します。

また、ラスター化された画像の PDF を生成する場合は、 ***PCIA.jar*** および *Pdf2PdfImgOCX.dll* を同様に格納してください。

.NETインターフェース

“3-Heights™ PDF to Image Converter API”は、純粋な.NET ソリューションを提供していません。そのかわりに、.NET アセンブリからコールされる DLL が用意されています。このことは、アプリケーションのインストールおよび配布時に考慮しなければなりません。

.NET アセンブリ（**NET.dll*）は、参照用に追加されます。*Pdf2ImgOCX.dll* は、ネイティブ DLL であって.NET アセンブリではありませんし、プロジェクトの参照用に追加することもできません。

Pdf2ImgOCX.dll DLL は、.NET アセンブリ *Pdf2ImgNET.dll* にからコールされます。そのため、アプリケーションの実行時に *Pdf2ImgOCX.dll* は、Windows システムによってもつけられなければなりません。

そのための一般的な方法は、*Pdf2ImgOCX.dll* をプロジェクトに追加して、必要に応じて（新規の場合など）出力のフォルダーにコピーされるように設定します。

あるいは、*Pdf2ImgOCX.dll* を環境変数 PATH で示されたフォルダーに配置するか、単にその DLL を手動で出力のフォルダーにコピーします。

なお、ラスター化された画像の PDF（画像データではなく）を生成する場合は、*Pdf2PdfImgOCX.dll* を（*Pdf2ImgOCX.dll* に換えて）使います。

また、新しいレンダリングエンジン（R2）を使う場合は、*Pdf2ImgAPI.dll* に読み替えてください。

ネイティブCインターフェース

ネイティブ C インターフェースの場合は、*pdf2Imgocx_c.h* をインクルードすると共に、*Pdf2ImgOCX.lib* をリンクしてください。

3.3 アンインストール、更新バージョンのインストール

“3-Heights™ PDF to Image Converter API”をアンインストールする場合は、インストール時に行った作業をもとに戻します。たとえば、regsevr32 –u をつかって登録を解除したり、インストール（コピー）したファイルをすべて削除したりします。

このとき、期限の切れた評価用の DLL は削除できないことに注意してください。もし、その DLL を削除したいときは、まず新しい評価用の DLL をダウンロードして登録してください。これによって古い DLL は新しいものに書き換わります。その後に、その DLL を削除してください。

新しいバージョンをインストールした場合は、それ以前の古いバージョンをアンインストールする必要はありません。古いバージョンのファイルは、新しいバージョンのファイルで書き換えられます。COM インターフェースを使っている場合は、単に新しい DLL を登録してください、古いバージョンを登録解除する必要はありません。

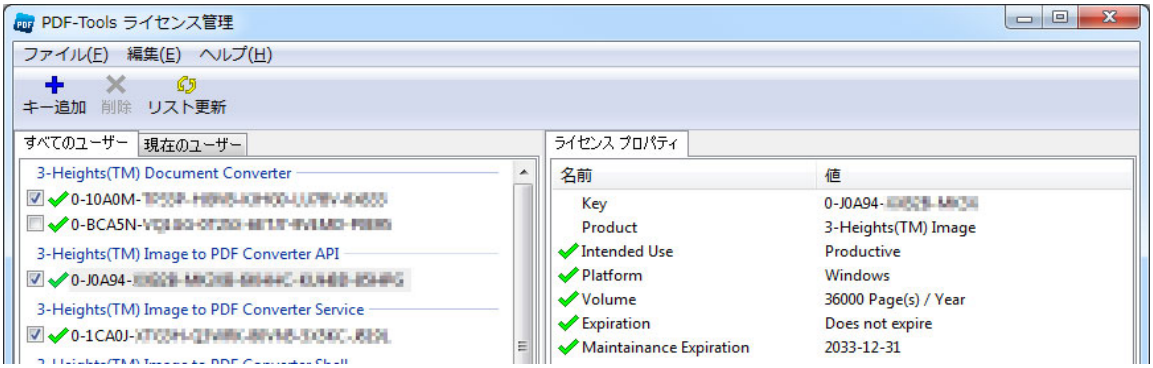
4 ライセンス管理

アプリケーションにライセンスキーを渡すには、以下の3つの方法があります。

1. ライセンスキーを GUI（Graphical User interface、グラフィカル ユーザー インターフェース）ツールでインストールする方法：これは、手動でライセンスキーを管理するには最も簡単な方法ですが、Windows OS のみで利用できます。
2. ライセンスキーをコマンドライン（Shell、シェル）ツールでインストールする方法：これは、すべて（Windows 以外）のシステムでのライセンス管理に適した方法です。
3. ライセンスキーをアプリケーションの実行時に“LicenseKey”プロパティを使って渡す方法：これは、OEM ライセンスを利用している場合に適した方法です。

4.1 GUIライセンス管理ツール

GUI（Graphical User interface、グラフィカル ユーザー インターフェース）ツール“*LicenseManager.exe*”は、製品をインストール（または解凍）したフォルダーの中の *bin* フォルダーに格納されています。



インストールされたすべてのライセンスキーのリスト

ライセンス管理ツールは、常に左ペインにインストールされているすべてのライセンスキーを表示します。このリストには、他の PDF Tools 製品のライセンスも含まれます。

ライセンスを以下のいずれかに対し有効にします。（タブで切り替えます。）

- すべてのユーザーに有効なライセンスとする。変更のために管理者権限が必要です。
- 現在のユーザーにのみ有効なライセンスとします。

ライセンスキーの追加及び削除

ライセンスキーは、ツールバーの「キー追加」および「削除」で追加または削除ができます。

- 「キー追加」ボタンは、タブで選択された（「すべてのユーザー」または「現在のユーザー」）リストにライセンスキーを追加します。
- 「削除」ボタンは、現在選択されているライセンスキーを削除します。ボタンは、ライセンスキーが選択されると有効になります。

ライセンスのプロパティを表示

ライセンスキーが選択されると、そのプロパティがウィンドウの右ペインに表示されます。

インストールされたすべてのライセンスキーのリスト

特定の製品に1つ以上のライセンスキーをインストールできます。複数のライセンスキーがインストールされた場合は、ライセンスキーの左側にあるチェックボックスにチェックがあるもの(1つだけ)が有効なライセンスキーとなります。

4.2 コマンドライン ライセンス管理 ツール

コマンドライン ライセンス管理ツール"*licmgr*"は、Windows システムを除くすべてのプラットフォームの *bin* ディレクトリに格納されています。

このコマンドのすべてのオプションとその説明は、パラメータを一切指定せず、以下のように、実行すると表示されます。

```
licmgr
```

インストールされたすべてのライセンスキーのリスト

```
licmgr list
```

現在有効なライセンスキーには、その左側に星印("*")が表示されます。

ライセンスキーの追加及び削除

新しいライセンスキーをインストールする場合:

```
licmgr store X-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX
```

古いライセンスキーを削除する場合:

```
licmgr delete X-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX
```

上記のどちらの場合も、ライセンスキーを有効にする対象を指定できます。オプション –s と共に以下を指定します。

- `g`: すべてのユーザーが対象
- `u`: 現在のユーザーが対象

1つの製品に複数のライセンスキーが登録された場合

```
licmgr select X-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX
```

4.3 ライセンスキーの格納場所

ライセンスキーは、プラットフォームに応じてそれぞれの格納場所に格納されます。

Windows

ライセンスキーは、以下のレジストリに格納されます。

- HKLM\Software\PDF Tools AG (すべてのユーザー)
- HKCU\Software\PDF Tools AG (現在のユーザー)

Mac OS X

ライセンスキーは、以下のファイルシステムに格納されます。

- /Library/Application Support/PDF Tools AG (すべてのユーザー)
- ~/Library/Application Support/PDF Tools AG (現在のユーザー)

Unix / Linux

ライセンスキーはファイルシステムに格納されます。

- /etc/opt/pdf-tools (すべてのユーザー)
- ~/.pdf-tools (現在のユーザー)

注意：上記ディレクトリのユーザー、グループおよびそれらのアクセス許可は、ライセンス管理ツールによって適切に設定されます。

ライセンスキーがすべてのユーザーによって読み取れるように、以下のような変更が必要な場合があります。

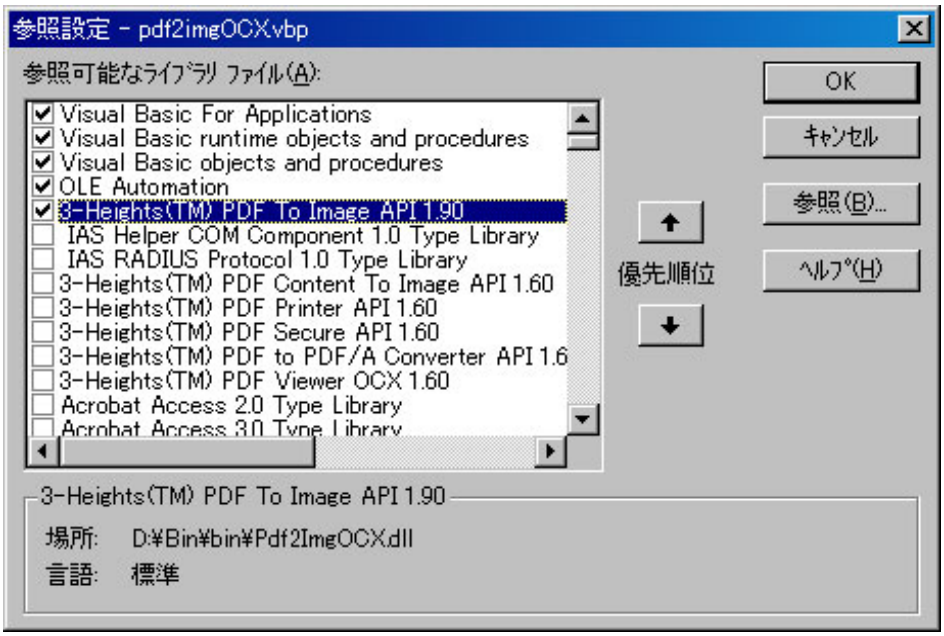
```
chmod -R go+rx /etc/opt/pdf-tools
```

5 プログラミング インターフェース

5.1 Visual Basic 6

“3-Heights™ PDF to Image Converter API”のインストール(または解凍)の後には、Visual Basic のサンプル *pdf2imgOCX.vbp* が *samples/VB/*に格納されています。新しいアプリケーションは、このサンプルをベースにして作成することもできますし、ゼロから作成することもできます。

もし、ゼロから開始するならば、最初に標準 EXE プロジェクトを作成し、“3-Heights™ PDF to Image Converter API”のコンポーネントをそのプロジェクトに追加します。



作成したプロジェクトで、アプリケーションを以下のように変更します。

1. コマンドボタンを追加し、それに適当な名前を設定します。
2. コマンドボタンをダブルクリックしハンドラーを表示します。表示されたハンドラーに以下のようにコードを追加します。このとき、ファイル名には変換される PDF ファイルと変換後の画像データのファイル名を指定してください。

```
Private Sub Command1_Click()
    Dim conv As New PDF2IMGOCXLib.Pdf2Img
    conv.ConvertFile "C:\pdf\in.pdf", "C:\image\out.tif", ""
End Sub
```

上記の2行は非常にシンプルです。それぞれ、(1) Pdf2Img インスタンスを生成し、(2) 入力用の PDF を開き、PDF ファイルの全ページをレンダリング(出力ファイルが TIFF 形式の場合は、複数ページで構成された画像データとなります)し、出力の画像ファイルに書き出します。

PDF ファイルを画像のページに変換する方法は2つあります。単純なアプローチは上記のとおりです。もうひとつの方法は、すこしコード量は増えますが強力なアプローチで、上記のような(全ページを変換するといった)大きな処理をいくつかの小さな処理に分割します。これによって、異なった複数の PDF ファイルを順に開き、指定したページをレンダリングし、1つの画像データに複数ページとして出力することができるようになります。

この処理を行うコードは、以下のとおりです。

```
Private Sub Command1_Click()
      Dim conv As New PDF2IMGOCXLib.Pdf2Img
      conv.CreateImage "C:\image\out.tif"
      conv.Open "C:\pdf\in1.pdf", ""
      conv.RenderPage 1
      conv.Close
      conv.Open "C:\pdf\in2.pdf", ""
      conv.RenderPage 3
      conv.RenderPage 6
      conv.Close
      conv.CloseImage
End Sub
```

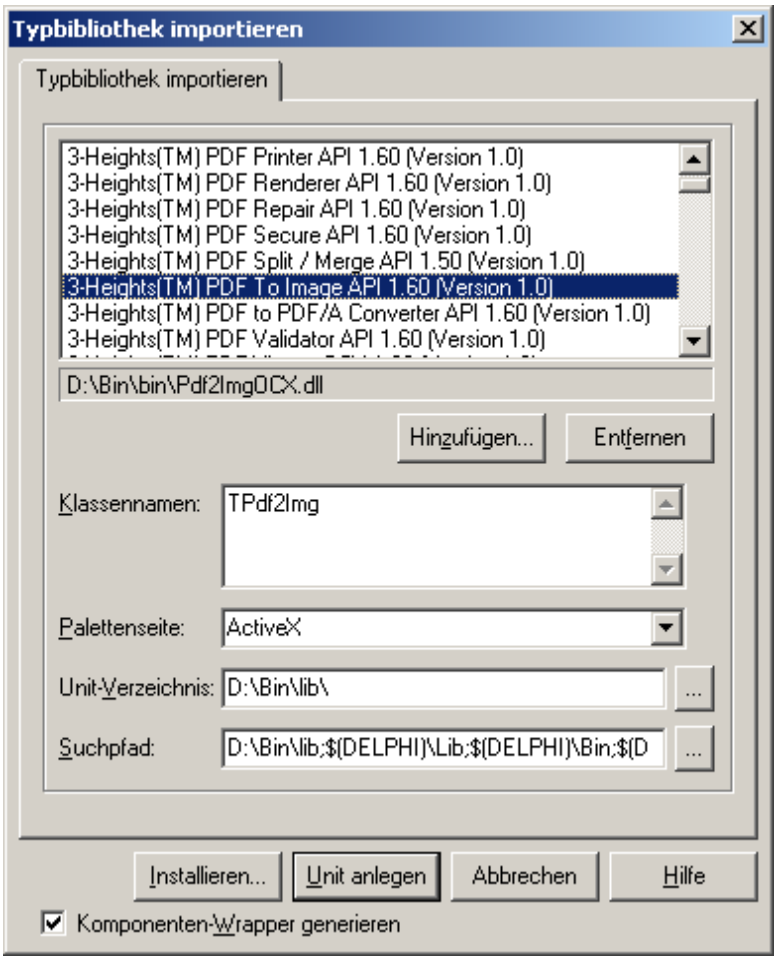
必要なのはこのコードですべてです。オプションとして、解像度や色深度などを変更する場合は、*ユーザー・ガイド*および*プログラマー・リファレンス*を参照してこのコードに追加してください。

5.2 Delphi

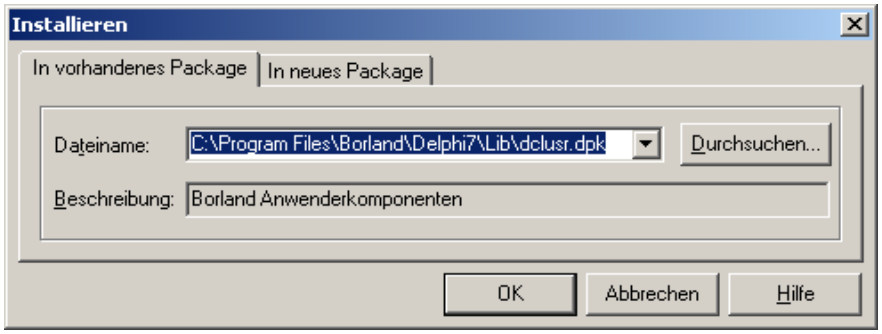
この章では“3-Heights™ PDF to Image Converter API”を Borland Delphi 7 環境で利用する方法を説明します。各スクリーンショットは、Windows 2000（ドイツ語環境）のものです。

1. DLL を登録（2.1 章参照）した後に、Delphi を起動します。

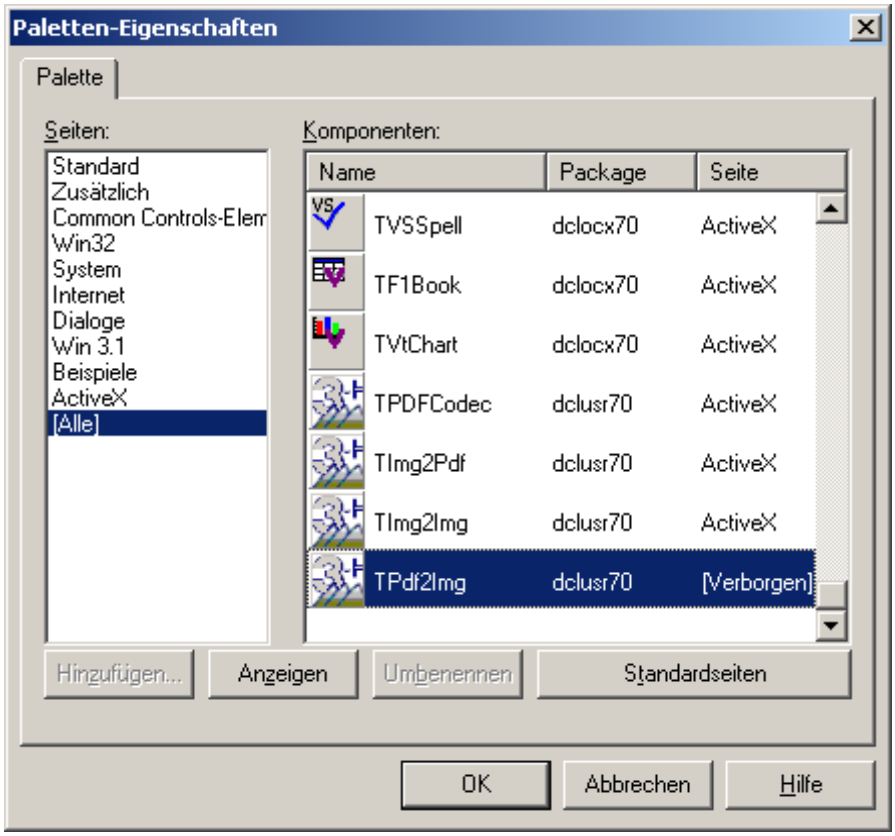
2. *Project* メニューに進み、*Import Type Library* を選択します。DLL が正しく登録されていれば、以下のように表示されてますので、“3-Heights™ PDF to Image Converter API”を選択します。すると、TPdf2Img クラスが表示されます。このとき、クラス名での衝突がある場合はそれに応じてクラス名を調整します。Unit パスを選択します、そのパスのフォルダー（例えば、D:\bin\lib\など）がタイプライブラリのために生成したものであれば、そのパスを検索パスに追加します。



3. “3-Heights™ PDF to Image Converter API”コンポーネントを選択してから、*Install* をクリックして既存のパッケージ$(DELPHI)\Lib\dclusr.dpk にクラスを追加します。



4. コンポーネントが正しく登録されると、それを示すダイアログボックスが表示されます。パッケージ dclusr.dpk を閉じて変更を保存します。"3-Heights™ PDF to Image Converter API"のクラスは ActiveX タブに表示されます。もし、このように表示されない場合は *Components* メニューの *Configure Palette* を選択します。そして、それを[All]から[ActiveX]にドラッグ&ドロップで移動します。



5. これによって、Delphi のサンプル(*samples/Delphi/*フォルダーにあります)を開いたり、新しいアプリケーションを作成したりできるようになります。

5.3 ASP

"3-Heights™ PDF to Image Converter API"の COM(ASP や PHP などで利用されます)でのクラス名は、"*PDF2IMGOCX.Pdf2Img*"です。

以下は、VBScript を使った ASP のサンプルです。

```
<%@ Language=VBScript %>
<%
	option explicit
	dim conv
	set conv = Server.CreateObject("PDF2IMGOCX.Pdf2Img")
	if not conv.CreateImage("C:\temp\output.jpg") then
		Response.Write "<p>"
		Response.Write "Could not create output file." & "<br>"
	else
		Response.Write "<p>"
		Response.Write "Output file created successfully." & "<br>"
		if not conv.Open("C:\PDF-Tools\doc\license.pdf") then
			Response.Write "<p>"
			Response.Write "Could not open input file." & "<br>"
		else
			Response.Write "<p>"
			Response.Write "Input file opened successfully." & "<br>"
			if not conv.RenderPage(1) then
				Response.Write "<p>"
				Response.Write "Could not render page 1." & "<br>"
			else
				Response.Write "<p>"
				Response.Write "Page 1 rendered successfully." & "<br>"
			end if
		end if
	end if
	conv.Close
	conv.CloseImage
%>
```

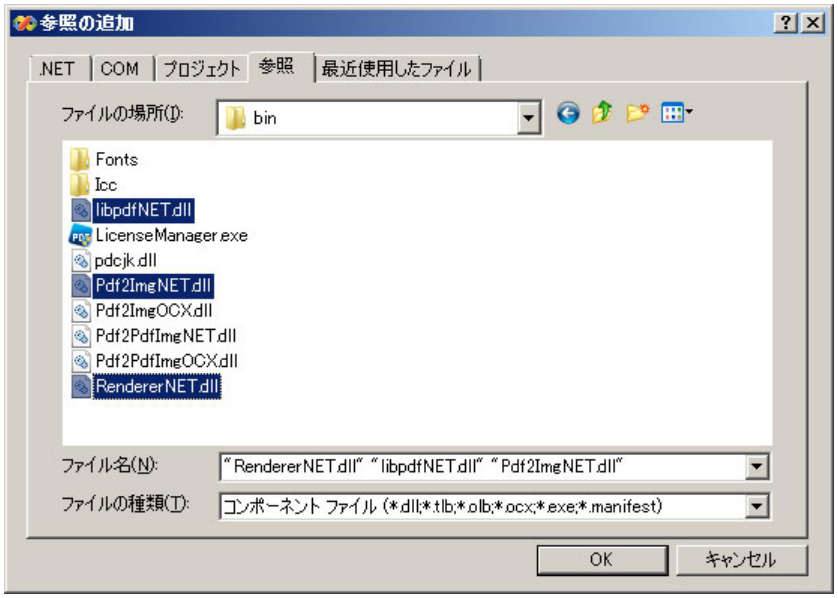
5.4 .NET

Windows 用の“3-Heights™ PDF to Image Converter API”には、.NET 用のサンプルとして MS Visual Studio 2005 用のサンプルが含まれています。このサンプルを参照すると、アプリケーションの作成をすばやく容易に開始できます。

新しいプロジェクトをゼロから作りためには、以下の手順に従ってください。

1. Visual Studio を起動して、C#または Visual Basic .NET プロジェクトを作成します。
2. .NET アセンブリへの参照を追加します。

   「ソリューション エクスプローラ」で生成したプロジェクトを右クリックし、「参照の追加」を選択します。「参照の追加」ダイアログが表示されます。「参照」タブを選択して.NET アセンブリである、*libpdfNET.dll* と *Pdf2ImgNET.dll* および *RendererNET.dll* を以下のようにプロジェクトに追加します。



3. 名前空間をインポート(注意:オプションですが有用です)
4. コードを記述

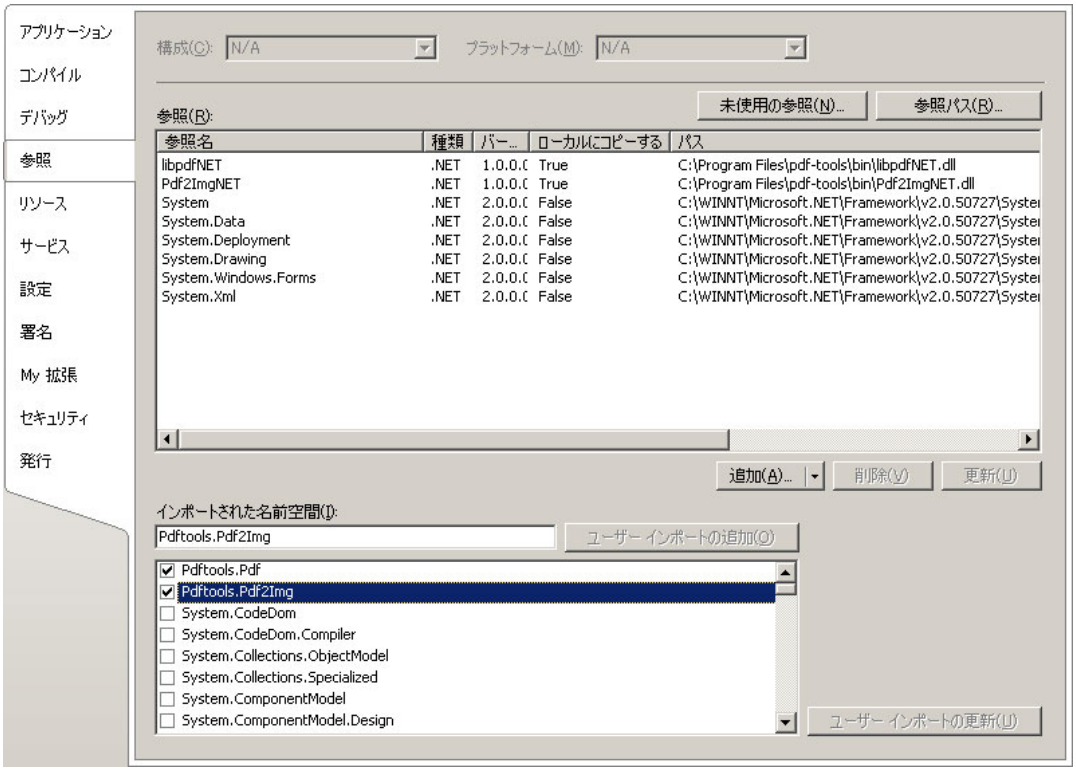
ステップの3と4は C#と Visual Basic では別に説明します。

Visual Basic .NET

3. 作成したプロジェクトのプロパティを表示します。向かって左側に「参照」メニューがあります。先ほど追加した .NET アセンブリは画面の上のほうのウィンドウに表示されます。

   下方のウィンドウには、インポートした2つの名前空間 *Pdftools.Pdf* と *Pdftools.Pdf2Img* があります。

   以下と同様の設定が必要です。



4. これで .NET インターフェースが以下のように利用できるようになりました。

```
Dim conv As New Pdftools.Pdf2Img.Converter
conv.Open(...)
...
```

C# .NET

3. 以下の名前空間を追加します。

```
using Pdftools.Pdf;
using Pdftools.Pdf2Img;
```

4. これで .NET インターフェースが以下のように利用できるようになりました。

```
Converter conv = new Converter();
conv.Open(...);
...
```

トラブルシューティング

.NET インターフェースを利用する場合で最も一般的に生じる問題は、実行時にネイティブ DLL が検出されないことです。通常これは、コンストラクタがはじめてコールされたときに例外（一般に System.TypeInitializationExceptio）がスローされます。

解決のためには、ネイティブ DLL が実行時に検出されるようにします。そのためには、この章の中の「インストール」にある「.NET インターフェース」を参照してください。

6 ユーザーズ・ガイド

6.1 サポートされるコーデック

以下の表は、PDF to Image Converter がサポートするコーデックそれぞれの機能です。

Codec	BitsPerPixel	Gray	Indexed	Quality	Compression
TIFF	1, 2, 4, 8, 24*	Yes	Yes	Yes	Raw, Flate, LZW, JPEG, Group3, Group3_2D, Group4
JPEG	8, 24	Yes	No	Yes	JPEG (Lossy only)
BMP	1, 2, 4, 8, 24*	Yes	Yes	No	Raw
GIF	2-8	Yes	Yes	No	LZW
PNG	1-8, 24	Yes	Yes	No	Flate
JBIG2	1	Yes	No	Yes	JBIG2 (Lossless: Q = 100)**
JPEG2000	8, 24	Yes	Yes	Yes	JPEG2000 (Lossless: Q = 100)**
PBM	1-8, 24	Yes	No	No	Raw
EPS	1, 2, 4, 8, 24*	Yes	No	No	Raw

コーデック表

Codec: The **Co**mpression/**Dec**ompression（圧縮・伸張）タイプ

Bits Per Pixel: サポートする、ピクセルあたりのビット数 1 = モノクロ, 8 = 256 色/グレースケール, 24 =1677 万色

Gray: グレースケースをサポート

Indexed: インデックスカラーをサポート

Quality: 品質パラメータの設定をサポート

Compression: 圧縮タイプをサポート

*) パレット生成において：パレットのエントリー数は、ビット/ピクセル値が8以下である場合に、2の（ビット/ピクセル値）乗となる。これは、3ビット/ピクセルの TIFF、BMP または EPS は生成できるが、そのパレットの大きさは4ビット/ピクセルのものと同じであることを意味している。しかしながら、3ビット/ピクセル画像は4ビット/ピクセルの画像よりも圧縮性に優れている。

**) JBIG2 および JPEG2000 で損失なし画像にするためには、品質パラメータを 100 にします。100 より小さい場合は、非可逆（損失のある）圧縮アルゴリズムが適用されます。

ファイルおよび圧縮のタイプ

多くの画像タイプは、圧縮アルゴリズムが既定されていますが、TIFF 形式は圧縮の形式を指定できます。

JPEG2000(PDF 1.5)および JBIG2(PDF 1.4)形式では、圧縮率は品質パラメータを介して調整できます。なお、これら2つの形式は新しい規格のものであり、古いバージョンの PDF 用のソフトウエアでは機能しない場合があります。

以下では、目的に応じた画像タイプの選定方法を提案しています。

圧縮なし・可逆圧縮

- 白/黒 JBIG2 (Q = 100) または TIFF で G4 圧縮
- グレースケール PNG, JPEG2000 (Q = 100)
- カラー PNG, JPEG2000 (Q = 100)

非可逆圧縮

- 白/黒 JBIG2
- グレースケール JPEG, JPEG2000
- カラー JPEG, JPEG2000

インターネット向け圧縮

- 白/黒 PNG
- カラー(写真) JPEG, PNG
- カラー(CG 画像) GIF, PNG

一般的注意事項：非可逆圧縮アルゴリズムのほうが、損失なし（可逆圧縮）に比して、より多くの CPU の演算能力を必要とします。非可逆圧縮ではファイルサイズは一般に小さくなりますが、しかし生成（圧縮）または読取り(伸張)のコストが大きくなります。

ほとんどのインターネットブラウザでサポートされている画像フォーマットは、JPEG、GIF および PNG です。

6.2 複数ページおよび単一ページ画像を生成するには

複数ページ画像

TIFF 形式は、複数ページをサポートした画像形式のひとつです。

“3-Heights™ PDF to Image Converter API”で PDF 文書を複数ページの TIFF 画像に変換するには、TIFF 画像をクローズせずに変換される PDF 文書のオープン/変換/クローズを繰り返します。

単一ページ画像

“3-Heights™ PDF to Image Converter API”で PDF 単一ページの TIFF 画像を生成するには、PDF 文書の1ページを変換した後に画像をクローズします。さらに続ける場合は、新たな画像ファイルをオープンしてページごとの変換処理を繰り返します。

6.3 ピクセルとポイントを同じにするには

“3-Heights™ PDF to Image Converter API”で生成される画像の既知の解像度は 150dpi ですが、PDF フォーマットは 72dpi を使っています。PDF が使っているポイント値に見合った解像度を指定する場合は、72dpi を使

ってください。一般的にモニターは96dpi なので、粒度の荒い画像を 100%ズームで表示させるときは注意が必要です。

6.4 ファイルサイズを小さくするには

生成される画像のファイルサイズを小さくするにはいくつかの方法があります。まずは、表示の品質を低くすることでサイズが小さくなることを認識しておいてください。

それ以外にに、画像のファイルサイズに影響を与えるものを以下に記します。

- 画像のピクセル数 (画像の高さと幅)
- ピクセルあたりのビット数
- 圧縮のタイプ
- イメージ コンテンツ (ディザリングの影響を受けて)

画像の大きさ

画像の大きさ(総ピクセル数)を少なくすることで、ファイルサイズは小さくなります。例えば、1024x768 ピクセルの画像は、600x480 ピクセルの画像より大きいファイルサイズとなります。

画像の大きさを設定する場合は以下の様にします。

```
SetBitmapDimensions(600, 480)
```

画像の大きさをポイントで設定する場合は以下のようにします。

```
SetPageSize(600, 480)
```

画像の大きさがポイントで設定されている場合、画像のピクセルサイズは解像度に応じて計算されます。

解像度

解像度(1インチあたりのドット数、DPI: Dots per Inch)を指定すると画像の詳細さを指定することができます。そのデフォルトの値は 150dpi で、生成された画像はズーム(拡大表示)しない場合においてシャープに見えます。もっと大きな値であればより詳細な画像が生成されますが、たくさんのピクセルが必要なためファイルサイズは大きくなります。また、解像度を低くするとファイルサイズは小さくなりますが、画像の表示品質は低下します。

以下のように、解像度を 150PDI に替えて 75DPI にしますと、ファイルサイズは約4分の1になります。

```
dpi = 75
```

ピクセルあたりのビット数

ピクセルあたり、1ビット(白黒)や8ビット(グレースケール)を24ビット(1677 万色)に替えると、ファイルサイズを小さくできます。ただし、色深度をサポートしている画像形式は限られますので注意してください。

```
BitsPerPixel = 8
```

フォーマット / 圧縮 タイプ

“3-Heights™ PDF to Image Converter”は、様々な画像形式をサポートしています。ほとんどの形式が圧縮をサポートしています。たとえば、PNG 画像はフラット圧縮され、JPEG 画像は JPEG 圧縮されます。なかでも、TIFF は圧縮のタイプを選択できます。

圧縮のタイプには可逆と非可逆といった根本的に異なる方式があります。

可逆圧縮（損失の無い圧縮）

可逆圧縮では、画像がオリジナルから圧縮された状態への変換でその内容が変更されることはありません。そのため、この変換は可逆的で、圧縮された状態から元の画像を再生することができます。

可逆圧縮は、一般に CG 画像のような画像やスキャンされたテキストに使用されます。

この圧縮タイプは、GIF、PNG、BMP、JPEG2000（品質を100にした場合）、JBIG2（品質を100にした場合）、TIFF（G3、G4、LZW、フラット）で利用できます。

非可逆圧縮（損失のある圧縮）

非可逆圧縮では、画像がよりよく圧縮されるように、その内容を変更します。そのため、非可逆圧縮された画像を元の状態に戻すことはできなくなります。これは、複数のアプリケーションで同じ画像を繰り返し非可逆圧縮すると、そのたびに画像の品質が劣化することを意味します。

非可逆アルゴリズムは通常、視覚的品質を犠牲にしてよりよい圧縮ができるようにします。非可逆圧縮は、一般に写真のような画像に使用されます。

この圧縮タイプは、JPEG、JPEG2000（品質を100未満にした場合）、JBIG2（品質を100未満にした場合）で利用できます。

TIFF 画像フォーマットでは、以下のような様々な圧縮タイプがサポートされています。

CCITT グループ 3、グループ 3-2D

CCITT グループ 3 は、CCITT グループ 4 の前身で、通常は低い圧縮率の単純なアルゴリズムです。

CCITT グループ 4

CCITT グループ 4 は、2値の TIFF 画像（たとえばファクシミリ用）の標準的な圧縮です。

LZW

LZW（の Lempel-Ziv-ウェルチ）圧縮は、画像の可逆圧縮アルゴリズムです。この圧縮アルゴリズムの日本国内での特許は、2004年に失効しています。

JPEG

TIFF は、画像を JPEG（非可逆圧縮アルゴリズム）で圧縮することができます。 JPEG は 8 および 24 ビット画像において、高い圧縮率を担保します。この圧縮は、写真や、文字をほとんど含まない画像に適しています。

ZIP (フラット)

ZIP は可逆圧縮アルゴリズムです。この圧縮は、品質を劣化さないで大きな画像を圧縮する場合などで利用されます。

一般に、フラット圧縮（または ZIP 圧縮）や JPEG 圧縮は、カラーまたはグレースケールの画像で使われます。CCITT グループ3、3–2D、4およびフラット圧縮は白黒(2値の)画像に使われます。

```
Compression = eComprFlate
```

画像コンテンツ、ディザリング

画像がどれだけ効率よく圧縮できるかは、画像のコンテンツそれ自体が影響します。たとえば、真っ白なページの圧縮は、写真の圧縮に比較してはるかに効率よく圧縮できることは容易に想像できます。

ディザリングは、1ビットの白黒のドットがグレーのドットに見えるような視覚効果を、カラー画像にも与えるように画像のピクセルを再配置するアルゴリズムです。"PDF to Image Converter"は、ディザリングを、カラー画像のためにも実装しています。

ディザリングは、このような複雑なピクセルの配置変換をおこなうため、うまく圧縮できずファイルのサイズが大きくなる場合があります。その場合は、ディザリングを無効にすると、ファイルサイズが小さくなります。

ディザリングを無効にするには以下のように設定します。これは一般に、スキャンした文字などの場合に利用されます。

```
Dithering = eDitherNone
```

さらに詳しくは、*ディザリング*の章を参照してください。

6.5 メモリ内で処理するメソッドを利用するには

"3-Heights™ PDF to Image Converter"は、同一の入力ファイル(又は出力ファイル)に何回もオープンとクローズを繰り返す場合があります。このような場合には、PDF や画像のデータをメモリから読み出したり、メモリに格納したりすることで、効率よく高速に処理できるようになります。

また、TIFF形式のように複数のページで構成された画像データを生成する場合は、*RenderPage*を複数回数コールしますので、画像データはメモリ上に作成すると効率がよくなります。

"3-Heights™ PDF to Image Converter"は、以下のようなメソッドでメモリ上のデータを取り扱います。

*CreateImageInMemory*は、画像をメモリ上に作成します。

*GetImage*は、メモリ上に保持されている画像をバイト配列で戻します。その大きさは、使用しているプログラミング言語による長さを戻す演算子によって取得できます。

*OpenMem*は、バイト配列に格納されたPDFデータをオープンします。

文書をメモリ上に生成する

以下は、メモリ上に画像を作成する Visual Basic 6 のサンプルです。このサンプルでは、画像を Variant 型のデータ pdfbytes に格納します。

```
Private Sub ConvertInMemory_Click()
Dim conv As New PDF2IMGOCXLib.Pdf2Img
Dim pdfbytes As Variant
Dim length As Long
      conv.Open "C:\input.pdf"
      conv.CreateImageInMemory ".tif"
      conv.RenderPage 1
      conv.RenderPage 2
      pdfbytes = conv.GetImage
      length = LenB(pdf)
      conv.CloseImage
      conv.Close
End Sub
```

メモリから文書を読み出す

以下の Visual Basic 6 のコードでは、レンダリングするために、メモリ上の PDF 文書を開きます。

(1)で、データベースから PDF データが格納されたバイトデータを読取り(これは説明用のサンプルです)、バイト配列に格納します。

(2)で、そのデータを開きます。

```
Private Sub OpenFromMemory_Click()
' (1) PDF データをメモリに格納する
      Dim conv As New PDF2IMGOCXLib.Pdf2Img
      Dim bChar() As Byte
      Dim lFileLenght As Long
      Open "C:\input.pdf" For Binary As #1
      lFileLenght = LOF(1)
      ReDim bChar(lFileLenght - 1)
      Get #1, , bChar
      Close #1
' (2) メモリに格納された PDF をオープンする
      If Not conv.OpenMem(bChar, "") Then
            MsgBox "couldn't open document"
      End If
End Sub
```

6.6 カラープロファイルを設定するには

“3-Heights”のレンダリングエンジンは、RGB 空間で機能します。色変換では、*Icc* フォルダーに格納されたカラープロファイル（*CMYK.icc* および *sRGB.icm*）が使われます。カラープロファイルは、配布された *Icc* フォルダーにあるリンクからダウンロードできます。

カラープロファイルは、以下の Web サイトからダウンロードすることもできます。

- *www.pdf-tools.com/public/downloads/resources/colorprofiles.zip*
- *www.color.org/srgbprofiles.html*
- *www.adobe.com/support/downloads/iccprofiles/icc_eula_win_dist.html*

カラープロファイルを変更するには、新しいカラープロファイルで上書きします。

レンダリングエンジンが、全くカラープロファイルが利用できない場合、色変換はノイゲバウアー（Neugebauer）アルゴリズムを使用して実施されます。

6.7 色を変更するには - ダーク ブラックについて

PDF で使われている CMYK カラーは、まず RGB カラーに変換されます。この変換は基本的に以下の2つの方法で達成れます。

1. CMYKカラープロファイルが適用されます。推奨のデフォルト カラープロファイルは、“U.S. Web Coated (SWOP) v2” です。違うカラープロファイルを使うと別の色になります。*SetCMYKProfile* および *SetsRGBProfile* を参照してください。RGB と CMYK カラープロファイルは、Windows の*%SystemRoot%\system32\spool\drivers\color*に格納されています。
2. *SetCMSEngine*（引数としてカスタム係数を保持しているテキストファイルを指定します）を使った色変換でノイゲバウアー（Neugebauer）アルゴリズムが適用されます。

   Visual Basic 6 コードスニペット サンプル

```
Dim conv As New PDF2IMGOCXLib.Pdf2Img
' 色管理エンジン設定
conv.SetCMSEngine App.Path & "\CmykToRgb.txt"
```

デフォルトのノイゲバウアー係数では、CMYKの黒(0,0,0,1)から変換されたRGBの黒は純粋な黒ではありません。次の係数は暗い黒を作成します。変更は、5行目に適用されます（元は、~0.2 です、*SetCMSEngine*を参照してください）。濃い黒を受け取るには、kの値をそれよりも低くする必要があります。

```
0.996078, 0.996078, 0.996078 ; White
0.000000, 0.686275, 0.937255 ; C
0.925490, 0.149020, 0.560784 ; M
1.000000, 0.949020, 0.066667 ; Y
0.100000, 0.100000, 0.100000 ; K
0.243137, 0.247059, 0.584314 ; CM
0.000000, 0.658824, 0.349020 ; CY
0.066667, 0.176471, 0.215686 ; CK
0.929412, 0.196078, 0.215686 ; MY
0.215686, 0.101961, 0.141176 ; MK
0.200000, 0.196078, 0.125490 ; YK
0.266667, 0.266667, 0.274510 ; CMY
0.133333, 0.098039, 0.160784 ; CMK
0.074510, 0.180392, 0.133333 ; CYK
0.215686, 0.121569, 0.113725 ; MYK
0.125490, 0.121569, 0.121569 ; CMYK
```

6.8 Isomorphicストレッチを適用するには

PDF のページサイズがピクセルで与えられていて、そのページを正確な寸法の画像に変換したいのに縦横比が PDF と違っている場合は、Isomorphic ストレッチを使うことができます。この変換では、x軸とy軸で異なる解像度を使用します。Y 方向の解像度を仮定すると、X 方向の解像度は以下のサンプルコードのように計算できます。

```
Dim conv As New PDF2IMGOCXLib.Pdf2Img
conv.Open ...
conv.CreateImage ...
For Page = 1 To conv.PageCount
   conv.XDPI = conv.YDPI * conv.PageHeight  / conv.PageWidth
                         * conv.BitmapWidth / conv.BitmapHeight
   conv.RenderPage Page
Next Page
conv.Close
conv.CloseImage
```

6.9 ディザリング

ディザリングは、実際の色として利用できない色をシミュレートするための画像で使用される、一般的な手段です。それは、色深度が低色深度であり、必要な色を（利用可能は）色またはグレーの網掛けを使って（例えば、白と黒のピクセルだけを使って）、それ以外の色をシミュレートするときに最良の結果を得られます。

参考

1. これらの画像によってディザリングの効果が明確になるように、以下に非常に低解像度の画像を用意してあります。高い解像度と色数の多い画像は、高品質の画像です。

2. レンダリングのフィルターや PDF の表示用アプリケーションのズームレベルは、画像のディザリング効果に影響を与える可能性があります。

カラー画像

色空間: RGB (24 bit)

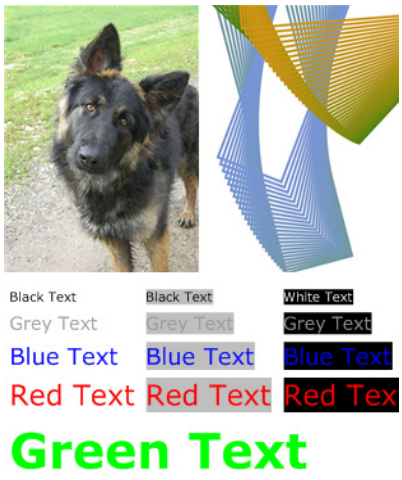
ディザリング: **None**

ファイルサイズ(PNG 形式): 129KB.

24ビット RGB カラーは、約1677万色の色を表現できますので、必要なすべての色を表現でき、どれもシミュレートする必要ありません。そのため、ディザリングを適用する必要はありません。

+最高品質

-大きなファイルサイズ



色空間: 16 colors (4 bit)

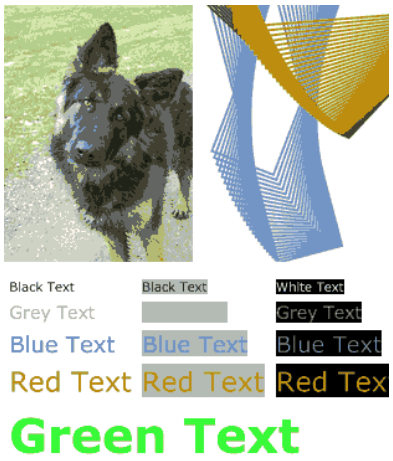
ディザリング: **None**

ファイルサイズ(PNG 形式): 16KB

+小さなファイルサイズ

+色数の少ない画像で良好(CG 画像やテキスト)

-色数の多い画像(写真など)良好ではありません。画像の一部が無地になって詳細が失われます。



色空間: 16 colors (4 bit)

ディザリング: **Floyd-Steinberg**

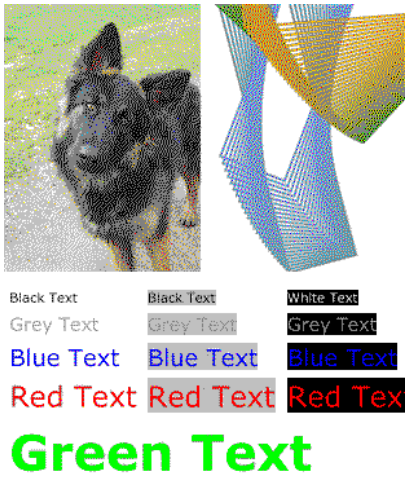
ファイルサイズ(PNG 形式): 18KB

+ レンダリングの詳細が比較的良好

+ 一般的には比較的良好な品質(特に写真画質においてディザリングのない場合に比較して)

- 場合によっては、アーチファクト(目立つピクセル)を生成します。

- ディザリングのない場合よりもファイルサイズが大きくなる場合があります。



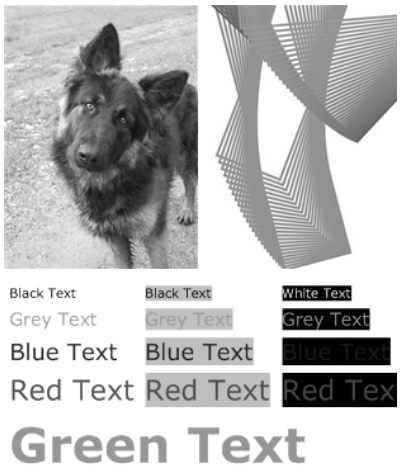
モノクロ画像

色空間: グレースケール (8 bit)

ディザリング: **None**

ファイルサイズ(PNG 形式): 46KB

(参照用の8ビットのグレースケールの元画像です。)



色空間: グレースケール (1 bit)

ディザリング: **None**

ファイルサイズ(PNG 形式): 2.6KB

+ 最小のファイルサイズ

+ コントラストの高い文書(白地に黒文字)で良好

+ アーチファクトを生成しません。

- 詳細は失われます。グレー色は近似されず黒または白に変換されます(画像やその部分においても黒または白に変換されます)。

色空間: グレースケール (1 bit)

ディザリング: **Floyd-Steinberg**

ファイルサイズ(PNG 形式): 9KB

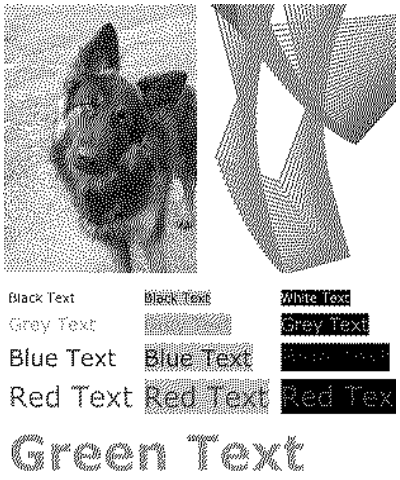
+ 一般的に高品質(特に写真のような画像)

+ ディザリングのグレー色(濃淡)が近似されます。

- ディザリングなしに比して、ファイルサイズが大きくなります。

- アーチファクトを生成します(たとえば、非常に明るいグレー色の紙などが遠く離れた単一の黒いピクセルで表現されます)。

-一般にテキストには不向きです。



色空間: グレースケール (1 bit)

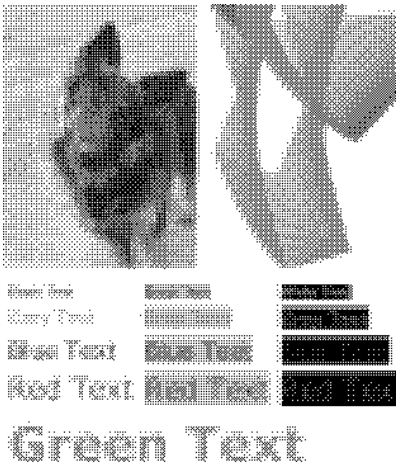
ディザリング: **Halftone**

ファイルサイズ(PNG 形式): 4KB

+ ファイルサイズ 小

+ グレーを近似します。

- テキストや CG 画像には適しません。



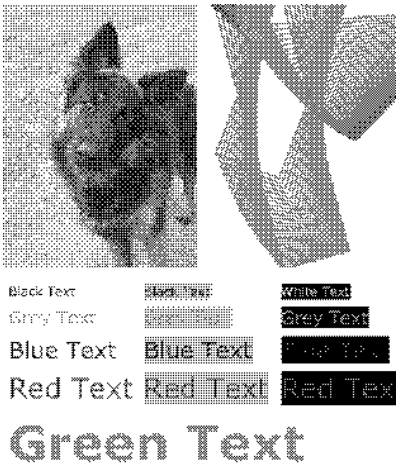
色空間: グレースケール (1 bit)

ディザリング: **Pattern**

ファイルサイズ(PNG 形式): 5KB

+ すべてのコンテンツ(テキスト、写真画像、CG 画像など)に適用可能

- しかし、あまり優秀な結果ではありません。



ガイドライン

上記のサンプルで示したように、ディザリングはコンテンツによって違った結果となります。以下では、コンテンツのタイプによってどのディザリングを選択するのがよいかを提案します。

テキスト、OCR	ディザリングしない

少ない色数でかつ明るい色のない CG 画像	ディザリングしない
色数の多い CG 画像	どのディザリングが老巧な結果となるかテストします。
写真画像	Floyd-Steinberg
混合画像	どのディザリングが老巧な結果となるかテストします。
混合画像で高解像度	300dpi 以上の解像度の場合は、Floyd-Steinberg でほとんどの場合最高の結果となります（白地に真っ黒な文字の場合は例外としてディザリングを適用しません）。

ディザリングは白と黒（1ビット）の画像のような低色深度の画像に対してのみ適用されるべきであることに留意してください。 8 ビット以上（256色または階調）の色深度の画像にディザリングを適用すると、少なからず視覚的な影響が現れます。

7 プログラマーズ・リファレンス

このマニュアルでは、COM インターフェースをのみを記載しています。他のインターフェース（C、Java、 .NET）でも COM の場合と同様の名前や同様の呼び出し方法で利用できます。

7.1 PDF to Image インターフェース

ここに揚げるインターフェースは、*Pdf2ImgOCX.dll* に含まれています。
このインターフェースは、入力を PDF 文書として、ラスター画像（例えば、TIFF 画像）を出力します。

BitmapHeight

```
Property Long BitmapHeight
Accessors: Get
```

ビットマップの高さをピクセル値で返します。

BitmapWidth

```
Property Long BitmapWidth
Accessors: Get
```

ビットマップの幅をピクセル値で返します。

BitsPerPixel

```
Property Integer BitsPerPixel
Accessors: Get, Set
Default: 24
```

色深度を取得または設定します。モノクロ：1、グレースケール：8、RGB1677万色：24、CMYK：32
1ビット/ピクセルを使う場合は、アンチアイリアス（*eOptionHighQuality*）を無効にすることを推奨します。

Center

```
Property Boolean Center
Accessors: Get, Set
Default: False
```

センターモードを取得または設定します。設定値は以下のとおりです。
True（真） ページは、水平垂直共にセンタリングされます
False（偽） ページは、左上に寄せてレンダリングされます

Close

```
Method Boolean Close()
```

現在オープンしている入力ファイルをクローズします。

- 戻り値:
  True: 入力ファイルは正常にクローズされました。
  False: 上記以外

CloseImage

```
Method Boolean CloseImage()
```

開いている画像ファイルをクローズします。ファイルが既にクローズされている場合は、何もしません。

- 戻り値:
  True: 画像ファイルは正常にクローズされました。
  False: 上記以外

ColorSpace

```
Property TPDFColorSpace ColorSpace
Accessors: Get, Set
Default: eColorRGB
```

色空間を取得または設定します。TPDFColorSpace参照
モノクロ画像の場合は、グレースケール色空間を選択しなければなりません。

Compression

```
Property TPDFCompression Compression
Accessors: Get, Set
Default: eComprRaw
```

TIFF 画像の圧縮タイプを取得または設定します。他の画像フォーマットでは、ファイルの拡張子で自動的に定義されます。
サポートされる*TPDFCompression*の値は、対応する列挙値です。

ConvertFile

```
Method Boolean ConvertFile(String PDFFileName, String ImageFileName, String
Password)
```

PDF ファイルのすべてのページを複数ページの TIFF 画像に変換します。

- パラメータ:
  PDFFileName: PDF 文書のファイル名(またはパス名)
  ImageFileName: TIFF 画像のファイル名(またはパス名)

Password (optional): 暗号化されたPDF文書のユーザーまたはオーナーパスワードを指定します。省略された場合は、空の文字列が使用されます。

- 戻り値:

  True: ファイルの生成に成功しました。

  False: PDF ファイルがない、破損している、または、パスワードが無効です。または、画像ファイルがロックされています。

CreateImage

Method Boolean CreateImage(String FileName)

Create a new image file.

- パラメータ:

  FileName: ファイル名（またはパス名） ファイル名は、その拡張子で画像形式を定義します。サポートする拡張子は以下のとおりです。

  - .bmp (Windows Bitmap Format)
  - .gif (Graphics Interchange Format)
  - .jb2 (JBIG2, Bi-level Images)
  - .jpg, jpeg (Joint Photographic Experts Group)
  - .jp2 (JPEG2000)
  - .jpf, jpx (JPEG2000, Part 2 – Coding Extensions)
  - .png (Portable Network Graphics)
  - .tif, .tiff (Tagged Image File Format)

- 戻り値:

  True: ファイルは正常に生成されました。

  False: 上記以外

CreateImageInMemory

Method Boolean CreateImageInMemory(String Extension)

画像イメージをバイトデータでメモリに格納します。*GetImage*参照

- パラメータ:

  Extension: 画像形式を指定する拡張子。指定できる拡張子は、*CreateImage*を参照します。拡張子は“.”で開始されなければなりません。

- 戻り値:

  True: 画像の生成に成功しました。

  False : 上記以外

Dithering

```
Property TPDFDithering Dithering
Accessors: Get, Set
Default: eDitherFloydSteinberg
```

ディザリングのアルゴリズムを取得または設定します。これは、主に低い色深度（白黒画像）の画像で有用です。
サポートする値は、*TPDFDithering* の列挙値です。

DPI

```
Property Single DPI
Accessors: Get, Set
Default: 150
```

画像の解像度を DPI（Dots per Inch）で取得または設定します。

設定: x、y両方の DPI 値を同じ値で設定します。

取得: x と y の積の平方根を返します。

ErrorCode

```
Property TPDFErrorCode ErrorCode
Accessors: Get
```

このプロパティは、直近のエラーコードを取得する場合にアクセスします。*TPDFErrorCode*参照

FaxHSetting, FaxSSetting

```
Method Boolean FaxHSetting()
Method Boolean FaxSSetting()
```

2 つのメソッドは、TIFF の F 値を設定します。値は、*RotateMode* = RotatePortrait、*SetBitmapDimensions*(1728, 0)、XDPI = 204、YDPI = 196 (Fax H)/98(Fax S)、Compression = eComprGroup3 です。

FillOrder

```
Property Integer FillOrder
Accessors: Get, Set
Default: 1
```

ビットフィル順序を取得または設定します。1は、MSB（最上位ビット）が先、2は、LSB（最下位ビット）が先です。

FilterRatio

```
Property Integer FilterRatio
Accessors: Get, Set
Default: 1
```

このプロパティは、画像を高い解像度でレンダリングしてから目的の解像度にダウンサンプリングする手法を有効かつパラメータ化します。この結果として、画像はスムーズに(アンチアイリアス)処理されます。

目標の解像度が低い(72dpi以下)場合に、スーパーサンプリングを適用すると画質が向上します。

この処理では、レシオの2乗に比例してメモリとCPU時間が増大します。そのため2か3を使うべきです。

高すぎる値(元画像の大きさとの組み合わせで)が設定されると無視されます。

FitPage

```
Property Boolean FitPage
Accessors: Get, Set
Default: True
```

ページフィットモードを取得または設定します。Trueに設定するとページは画像(の高さまたは幅)に収まるように大きさが調整されます。Falseに設定するとページは実際の大きさでレンダリングされます。

GetImage

```
Method Variant GetImage()
```

バイト配列(*CreateImageInMemory*で保存した)を戻します。

GetOcg

```
Method Ocg GetOcg(Integer Count)
```

OCG(オプショナル コンテント グループ)インターフェースを戻します。

- パラメータ:

  Count: OCGの数。数は、0から(OCGの数)-1 です。

- 戻り値:

  OCGインターフェース

インターフェース*Ocg*を参照してください。

HasAnnotations

```
Property Boolean HasAnnotations(Long IPageNo)
Accessors: Get
```

選択したページの注釈の有無を戻します。

HasColor

```
Method Boolean HasColor(Long IPageNo)
```

選択したページのカラーの有無を戻します。

HasPopups

```
Property Boolean HasPopups(Long IPageNo)
Accessors: Get
```

選択したページのポップアップの有無を戻します。

ImageQuality

```
Property Integer ImageQuality
Accessors: Get, Set
Default: 75
```

非可逆圧縮の品質を取得または設定します。値は、1から100で設定します。

OcgCount

```
Property Long OcgCount
Accessors: Get
```

文書のOCG（オプショナル コンテント グループ、“レイヤー”として知られています）の数を戻します。*GetOcg*を参照してください。

- 戻り値：

  文書中のOCGの数

Open

```
Method Boolean Open(String FileName, String Password)
```

PDFファイルを開きます。既にPDF文書が開かれていた場合は、先のPDF文書を閉じてから新たに開きます。

- パラメータ：

  FileName: ファイルのファイル名（またはパス名）

  Password (optional): 暗号化されたPDF文書のユーザーまたはオーナーパスワードを指定します。省略された場合は、空の文字列が使用されます。

- 戻り値：

  True: ファイルが正常に開かれました。

  False: ファイルが存在しない、破損している、またはパスワードが無効です。

OpenMem

```
Method Boolean OpenMem(Variant MemBlock, String Password)
```

メモリ上のPDF文書を開きます。既にPDF文書が開かれていた場合は、先のPDF文書を閉じてから新たに開きます。.

- パラメータ:

  MemBlock: PDF文書が格納されたバイト配列を指定します。

  Password (optional): 暗号化されたPDF文書のユーザーまたはオーナーパスワードを指定します。省略された場合は、空の文字列が使用されます。

- 戻り値:

  True: ファイルが正常に開かれました。

  False: ファイルが存在しない、破損している、またはパスワードが無効です。

Options

```
Property TPDFRendererOption Options
Accessors: Get, Set
Default: eOptionBicubic + eOptionHighQuality + eOptionTransparency
```

オプションを取得または設定します。
オプションを追加する場合はOR(論理和)し、削除する場合はAND NOT(否定で論理積)します。
列挙型定数*TPDFRendererOption*を参照してください。

PageCount

```
Property Long PageCount
Accessors: Get
```

PDF文書全体のページ数を取得します。有効なPDF文書が開かれていない場合は、0を戻します。

PageHeight

```
Property Long PageHeight
Accessors: Get
```

ページの高さをポイント単位で取得します。

PageWidth

```
Property Long PageWidth
Accessors: Get
```

ページの幅をポイント単位で取得します。

PreserveAspectRatio

```
Property Boolean PreserveAspectRatio
Accessors: Get, Set
Default: False
```

この値がアップスケーリングまたはダウンスケーリングでTrueに設定されていると、出力画像の縦横比が入力ファイルのそれと同じになり、画像は*SetBitmapDimensions*で設定されたサイズにフィットします。

Quality

推奨されません。*ImageQuality*を使ってください。

RenderingMode

推奨されません。バージョン2. 0またはそれ以降では、1つのレンダリングのみとなりました。

RenderPage

```
Method Boolean RenderPage(Long PageNumber)
```

ラスター画像に選択されたPDF文書をレンダリング（変換）します。

- パラメータ：

  PageNumber: PDF 文書のページ番号、1以上の値を指定します。

- 戻り値：

  True: ページのレンダリングは成功しました。

  False: ページはレンダリングできませんでした（ページ数が範囲外、PDF が開かれていない、画像を生成できない、など）。

RepeatWatermark

```
Property Boolean RepeatWatermark
Accessors: Get, Set
Default: False
```

Trueに設定すると、*SetWatermarkImage*によって指定されたウォーターマーク（透かし）をタイル状に敷き詰めて表示します。

RotateMode

```
Property TPDFRotateMode RotateMode
Accessors: Get, Set
Default: eRotateNone
```

回転モードを取得または設定します。列挙型定数*TPDFRotateMode*を参照してください。

SetBitmapDimensions

Method Void SetBitmapDimensions(Long X, Long Y)

画像のサイズをピクセル単位で設定します。

- パラメータ:
  - ***X:*** X 方向のピクセル単位の画像サイズ
  - ***Y:*** Y 方向のピクセル単位の画像サイズ

SetCMSEngine

Method Boolean SetCMSEngine(String CMSEngine)

カラー マネージメント システム(Color Management System、CMS)を指定します。以下の文字列をサポートしています。

- "None": CMS を適用しません。結果は、可能な最大のコントラストとなります。
- "Neugebauer": ノイゲバウアーアルゴリズムは効率的に CMYK を RGB に変換します。これは、いずれのカラープロファイルも必要としませんが、結果はカラープロファイルを使用した変換に似ています。
- "MSICM": Microsoft ICM エンジンは、デフォルトのエンジンです。
- "lcms": lcms は、ほとんどの Unix プラットフォーム上で使用されます。
- FileName: ファイル名(またはパス名)を指定し、ノイゲバウアーアルゴリズムの設定可能なバージョンが適用されます。係数は、テキストファイルで定義します。ノイゲバウアー係数のデフォルト(Red, Blue, Green ; Color)値は以下のとおりです。

```
0.996078, 0.996078, 0.996078 ; White
0.000000, 0.686275, 0.937255 ; C
0.925490, 0.149020, 0.560784 ; M
1.000000, 0.949020, 0.066667 ; Y
0.215686, 0.203922, 0.207843 ; K
0.243137, 0.247059, 0.584314 ; CM
0.000000, 0.658824, 0.349020 ; CY
0.066667, 0.176471, 0.215686 ; CK
0.929412, 0.196078, 0.215686 ; MY
0.215686, 0.101961, 0.141176 ; MK
0.200000, 0.196078, 0.125490 ; YK
0.266667, 0.266667, 0.274510 ; CMY
0.133333, 0.098039, 0.160784 ; CMK
0.074510, 0.180392, 0.133333 ; CYK
0.215686, 0.121569, 0.113725 ; MYK
0.125490, 0.121569, 0.121569 ; CMYK
```

ノイゲバウアーアルゴリズムは色と対応する重み係数の量に基づいて、混色します。

SetCMYKProfile

```
Method Boolean SetCMYKProfile(String FileName)
```

CMYK プロファイルのパスを指定します。パスを指定しない場合は、*bin/Icc/CMYK.icc* が使われます（利用可能な場合）。カラープロファイルが検索できない場合は、色変換アルゴリズムが適用されます。

- パラメータ：

  FileName: ICC CMYK カラープロファイルのファイルパス

SetPageSize

```
Method Void SetPageSize(Single X, Single Y)
```

ポイント値で画像のサイズを設定します。

- パラメータ：

  X: 画像の X 方向のポイントサイズ

  Y: 画像の Y 方向のポイントサイズ

SetsRGBProfile

```
Method Boolean SetsRGBProfile(String FileName)
```

sRGB プロファイルのパスを指定します。パスを指定しない場合は、*bin/Icc/sRGB.icm* が使われます（利用可能な場合）。カラープロファイルが検索できない場合は、色変換アルゴリズムが適用されます。

- パラメータ：

  FileName: ICCRGB カラープロファイルのファイルパス

- 戻り値：

  True: カラープロファイルの設定に成功しました。

  False: 上記以外

SetWatermarkImage

```
Method Boolean SetWatermarkImage(String FileName, Single Left, Single Bottom)
```

ウォーターマーク（透かし）をファイルから読み込んで、指定位置（ポイント単位）に貼り付けます。*RepeatWatermark* がTrueに設定されていなければウォーターマークは1度のみ配置されます。

XDPI, YDPI

```
Property Single XDPI
Property Single XDPI
Accessors: Get, Set
Default: 150
```

画像解像度の X 軸 Y 軸の DPI（Dots per Inch）を取得または設定します。

7.2 PDF to PDF Image インターフェース

このインターフェースは *Pdf2PdfImgOCX.dll* に含まれています。

このインターフェースは入力に PDF 文書を取り、ラスター化画像 PDF 文書を出力します。ラスター化 PDF 文書とは、ベクター形式のグラフィックスやテキスト文字が含まれずに、各ページがただひとつのラスター画像で構成されたものです。

BitsPerPixel

```
Property Integer BitPerPixel
Accessors: Get, Set
Default: 24
```

色深度を取得または設定します。モノクロ:1、グレースケール:8、RGB1677万色:24、CMYK:32

Center

```
Property Boolean Center
Accessors: Get, Set
Default: False
```

センターモード値を取得または設定します。設定値は以下のとおりです。

True(真) ページは、水平垂直共にセンタリングされます

False(偽) ページは、左上に寄せてレンダリングされます

Close

```
Method Boolean Close()
```

現在オープンしている入力ファイルをクローズします。

- 戻り値:

  True: 入力ファイルは正常にクローズされました。

  False 上記以外

CloseImage

```
Method Boolean CloseImage()
```

開いている画像ファイルをクローズします。ファイルが既にクローズされている場合は、何もしません。

- 戻り値:

  True: 画像ファイルは正常にクローズされました。

  False 上記以外

Compression

```
Property TPDFCompression Compression
Accessors: Get, Set
Default: eComprRaw
```

圧縮のタイプを取得または設定します。サポートされる値は、列挙型定数*TPDFCompression*で示された値です。圧縮タイプの概要やサポートする色深度などは、「*サポートされるコーデック*」を参照してください。

ConvertFile

```
Method Boolean ConvertFile(String PDFFileName, String ImageFileName, String Password)
```

PDF ファイルを画像の PDF ファイルに変換します。

- パラメータ:

  PDFFileName: 入力の PDF 文書のファイル名(またはパス名)

  ImageFileName: 出力の PDF 文書のファイル名(またはパス名)

  Password (optional): 暗号化された入力のPDF文書のユーザーまたはオーナーパスワードを指定します。省略された場合は、空の文字列が使用されます。

- 戻り値:

  True: ファイルの変換に成功しました。

  False: PDF ファイルがない、破損している、または、パスワードが無効です。または、出力の PDF ファイルがロックされています。

CopyLinks

```
Property Boolean CopyLinks
Accessors: Get, Set
Default: True
```

リンクをコピーするオプションを取得または設定します。

CopyOutlines

```
Property Boolean CopyOutlines
Accessors: Get, Set
Default: True
```

アウトライン(しおり)を設定するオプションを取得または設定します。

CopyViewerPreferences

```
Property Boolean CopyViewerPreferences
Accessors: Get, Set
Default: True
```

ビュワー プレファレンス オプション（ページレイアウト、ページモード、開き方）を取得または設定します。

CreateImage

```
Method Boolean CreateImage(String FileName , String UserPw, String OwnerPw,
TPDFPermission PermissionFlags, Long KeyLength)
```

新しい画像の PDF ファイルを生成します。

- パラメータ：

  FileName: 生成する PDF ファイルのファイル名（またはパス名）

  UserPwd (optional): PDF 文書のユーザーパスワードを指定します。0（ゼロ）を指定すると、パスワードなしになります。

  OwnerPwd (optional): PDF 文書のオーナーパスワードを指定します。0（ゼロ）を指定すると、パスワードなしになります。

  PermissionFlags (optional): 許可フラグを指定します。デフォルトでは、権限が付与されません。設定できる権限は、列挙型定数*TPDFPermission*に定義された値です。

  KeyLength (optional): 暗号化の鍵長を指定します。MD5 アルゴリズムの性質上、最大長は128ビットです。最小値は、40で8の倍数を指定しなければなりません。2つの一般的な値は40（標準暗号化）と128（Acrobat5以降）です。Acrobat は40または128のみをサポートしていることに注意してください。“3-Heights™” の他のツールは、上記のすべての鍵長をサポートしています。デフォルトの鍵長は、選択された許可フラグを基に計算されます。

- 戻り値：

  True: ファイルの生成に成功しました。

  False: 上記以外

出力ファイルを暗号化しないためには、許可フラグを-1 に、オーナーパスワードを 0 にします。

高品質での印刷を許可するには、ePermPrint および ePermDigitalPrint を設定する必要があります。

CreateImageInMemory

```
Method Boolean CreateImageInMemory()
```

PDFをメモリに格納します。メソッド*GetPDF*を参照してください。

- 戻り値：

  True: 出力の PDF のメモリ上への生成に成功しました。

  False: 上記以外

Dithering

Property TPDFDithering Dithering
Accessors: Get, Set
Default: *eFloydSteinberg*

ディザリングのアルゴリズムを取得または設定します。
サポートする値は、*TPDFDithering* の列挙値です。

DPI

Property Single DPI
Accessors: Get, Set
Default: 150

画像の解像度を DPI（Dots per Inch）で取得または設定します。

ErrorCode

Property Integer ErrorCode
Accessors: Get

このプロパティは、直近のエラーコードを取得するためにアクセスします。ヘッダーファイル *pdferror.h* を参照してください。

FitPage

Property Boolean FitPage
Accessors: Get, Set
Default: 24

ページフィットモードを取得または設定します。Trueに設定するとページは出力 PDF のサイズ（の高さまたは幅）に収まるように大きさが調整されます。Falseに設定するとページは実際の大きさでレンダリングされます。

GetPDF

Method Variant GetPDF()

バイト配列（*CreateImageInMemory*で保存したもの）を戻します。

GrayScale

Property Boolean GrayScale
Accessors: Get, Set
Default: False

グレースケール モードを取得または設定します。

Open

```
Method Boolean Open(String FileName, String Password)
```

PDFファイルを開きます。既にPDF文書が開かれていた場合は、先のPDF文書を閉じてから新たに開きます。

- パラメータ:

  FileName: ファイルのファイル名(またはパス名)

  Password (optional): 暗号化されたPDF文書のユーザーまたはオーナーパスワードを指定します。省略された場合は、空の文字列が使用されます。

- 戻り値:

  True: ファイルが正常に開かれました。

  False: ファイルが存在しない、破損している、またはパスワードが無効です。

OpenMem

```
Method Boolean OpenMem(Variant MemBlock, String Password)
```

メモリ上のPDF文書を開きます。既にPDF文書が開かれていた場合は、先のPDF文書を閉じてから新たに開きます。

- パラメータ:

  MemBlock: PDF文書が格納されたバイト配列を指定します。

  Password (optional): 暗号化されたPDF文書のユーザーまたはオーナーパスワードを指定します。省略された場合は、空の文字列が使用されます。

- 戻り値:

  True: ファイルが正常に開かれました。

  False: 上記以外

Options

```
Property TPDFRendererOption Options
Accessors: Get, Set
Default: eOptionBicubic + eOptionHighQuality
```

オプションを取得または設定します。
オプションを追加する場合はOR(論理和)し、削除する場合はAND NOT(否定で論理積)します。
列挙型定数*TPDFRendererOption*を参照してください。

PageCount

```
Property Long PageCount
Accessors: Get, Set
Default: undef.
```

PDF文書全体のページ数を取得します。有効なPDF文書が開かれていない場合は、値が定義されません。

- 戻り値:

1-n: PDF 文書のページ数

undef: PDF が開かれていません。

PreserveAspectratio

```
Property Boolean PreserveAspectratio
Accessors: Get, Set
Default: False
```

ページサイズの縦横比を保持します。True が設定されると、1つのページサイズだけが提供されます、False の場合は値が計算されます。

RenderPage

```
Method Boolean RenderPage(Long PageNumber)
```

入力 PDF の指定されたページが出力の PDF ファイルにレンダリングされます。

- パラメータ:

  PageNumber: PDF ファイルのページ番号

- 戻り値:

  True: ページのレンダリングは成功しました。

  False: 上記以外(ページ数が範囲外、PDF が開かれていない、出力ファイルを生成できない、など)。

RepeatWatermark

```
Property Boolean RepeatWatermark
Accessors: Get, Set
Default: False
```

Trueに設定すると、*SetWatermarkImage*によって指定されたウォーターマーク(透かし)をタイル状に敷き詰めて表示します。

RetainText

```
Property Boolean RetainText
Accessors: Get, Set
Default: False
```

True に設定されると、すべてのテキストが出力文書にテキストとしてコピーされ、すべての(テキスト以外の)コンテンツが含まれている背景画像の前面に配置されます。元の PDF で、テキストがそれ以外のオブジェクトより上に配置されていない場合は、修正のためのカバーがなくなってしまう場合があります。

False に設定すると、すべてのコンテンツが画像にレンダリングされます。

RotateMode

```
Property TPDFRotateMode RotateMode
Accessors: Get, Set
Default: eRotateNone
```

回転モードを取得または設定します。設定できる4つの値は、列挙型定数*TPDFRotateMode*で説明されています。
出力 PDF ファイルのすべてのページに、入力ファイルと同じ回転方向を持たせるためには、eRotateAttribute を設定します。

SetBitmapDimensions

```
Method Void SetBitmapDimensions(Long cx, Long cy)
```

画像のサイズをピクセル単位で設定します。

- パラメータ:

  cx: X 方向のピクセル単位の画像サイズ
  cy: Y 方向のピクセル単位の画像サイズ

SetPageSize

```
Method Void SetPageSize (Single X, Single Y)
```

画像のサイズをポイント単位で設定します。

- パラメータ:

  X: 画像の X 方向のポイントサイズ
  Y: 画像の Y 方向のポイントサイズ

SetWatermarkImage

```
Method Boolean SetWatermarkImage(String FileName, Single Left, Single Bottom)
```

ウォーターマーク(透かし)をファイルから読み込んで、指定位置(ポイント単位)に貼り付けます。*RepeatWatermark*がTrueに設定されていなければウォーターマークは1度のみ配置されます。

XDPI, YDPI

```
Property Single XDPI
Property Single YDPI
Accessors: Get, Set
Default: 150
```

画像解像度の X 軸 Y 軸の値を DPI(Dots per Inch)で取得または設定します。

7.3 Ocg インターフェース

OCG（Optional Content Group、“レイヤー”としても知られています）インターフェースは、OCG とそのプロパティを一覧します。

PDF の OCG は、グラフィックスや画像を編集するアプリケーションのそれとは違います。PDF のグラフィックス オブジェクトは OCG に属しません。その代わりに、グラフィックスの視認性（見え方）が OCG に依存した Boolean 関数によって計算されます。たとえば、パスは、OCG の“A”が ON で OCG の“B”が OFF のときだけ表示されます。

OCG の機能は、ISO32000-1 の 8.11.4 章で、または PDF リファレンスの 4.10 章で詳しく説明されています。OCG は PDF1.5 以降でサポートされています。

“PDF to Image Converter”においてOCGインターフェースは、“レイヤー”を一覧し、それらが表示されるか否かを設定するために利用します。OCGオブジェクトを取得するには、“PDF to Image”インターフェースの*OcgCount*と*GetOcg*を使います。

Label

```
Property Boolean Label
Accessors: Get
```

このフラグは、それが OCG またはラベルであるかを示します。ラベルは、階層内の OCG ラベルグループに使われます。これらに可視性を設定してもその効果はありません。

Level

```
Property Long Level
Accessors: Get
```

ユーザー インターフェースで OCG は、ツリーで表現できます。プロパティは、ツリー内の OCG の階層レベルを示しています。レベル0の OCG は、トップレベルの OCG です。レベル-1 は、その OCG が階層内のものではないことを意味していて、ユーザーに提示されるべきではりません。OCG 階層内の親エレメントは、ラベルまたは OCG です。レベル a のラベル b が先の a より高い場合、b は b と同じレベルの次のオブジェクトの親です。OCG b が先の OCG a よりもレベルが高い場合、a は b と同じレベルの次のオブジェクトの親です。階層は、コンテンツ内の OCG の実際のネストを反映していることに注意してください。すべての親の視認性が真に設定されている場合は、OCG の可視性を真に設定するだけで効果があります。

Name

```
Property String Name
Accessors: Get
```

OCG の数を戻します。

Visible

```
Property Boolean Visible
Accessors: Get, Set
```

OCG の視認性を取得または設定します。このプロパティは、コンテンツ オブジェクトの抽出をコントロールします。デフォルト値は、PDF 文書で設定されています。

目に見えないパスはページ上にマークを生成しませんが、それらはまだグラフィックの状態に影響を及ぼすことに注意してください。たとえば、現在の描画位置とクリッピング領域に対する効果は変わりません。したがって、すべてのパスが視認性と関係なく“active”で抽出できます。見えないパスは、パスの末端である“n”のオペレータだけでなく、塗りつぶしまたは線描オペレータを代替します。

例 1

id, OCG, レベル:	*階層*
0, OCG A, 0	- OCG A
1, OCG B, 0	- OCG B
2, OCG B1, 1	-- OCG B1
3, OCG B2, 1	-- OCG B2
4, OCG C, -1	hidden: OCG C

例 2

id, OCG/ラベル, レベル	*階層*
0, OCG A, 0	- OCG A
1, Label B, 1	- Label B
2, OCG B1, 1	-- OCG B1
3, OCG B2, 1	-- OCG B2
4, Label C, 1	- Label C
5, OCG C1, 1	-- OCG C1
6, OCG D, 0	- OCG D

7.4 列挙型定数

注意： 列挙型定数の接頭辞は、インターフェースによって異なります。“TPDF”は COM と C、“PDF”は.NET、Java では接頭辞なしとなります。

TPDFColorSpace

eColorGray	グレー
eColorGrayA	アルファ チャンネル付きグレー
eColorRGB	RGB（赤緑青）
eColorRGBA	アルファ チャンネル付き RGB
eColorCMYK	CMYK（シアン マゼンタ イエロー キー・プレート）
eColorYCbCr	YCbCr
eColorYCbCrK	YCbCrK

eColorPalette	パレット利用の色空間
eColorLAB	CIE L*a*b*
eColorOther	他
eColorCMYK_Konly	CMYK で K チャンネルのみ利用

TPDFCompression

eComprRaw	圧縮なし
eComprJPEG	Joint Photographic Expert Group
eComprFlate	Flate compression
eComprLZW	Lempel-Ziv-Welch
eComprGroup3	CCITT Fax Group 3
eComprGroup3_2D	CCITT Fax Group 3 2D
eComprGroup4	CCITT Fax Group 4
eComprJBIG2	Joint Bi-level Image Experts Group
eComprJPEG2000	JPEG2000
eComprTIFFJPEG	JPEG (6). 古いバージョンの JPEG です。旧来の画像アプリケーションでサポートしていましたが、新しいバージョンのものではサポートしていない場合があります。
eComprUnknown	Unknown compression

すべての画像フォーマットまたは色深度にすべての圧縮がサポートされていりわけではありませんので注意ください。

TPDFDithering

eDitherNone	No dithering
eDitherFloydSteinberg	Floyd-Steinberg (Default)
eDitherHalftone	Half-toning
eDitherPattern	Pattern Dithering
eDitherG3Optimized	Dithering optimized to compress well with Group 3
eDitherG4Optimized	Dithering optimized to compress well with Group 4

TPDFErrorCode

TPDFErrorCode のすべての値は、を“S”、“E”、“W”および“I”を伴った“PDF_”で開始します。これらの文字は、エラーのタイプ（**S**uccess、**E**rror、**W**arnning、**I**nformation）を表します。 これらは、PDF が壊れている（有効でない）ことを表します。Erro の場合は、その PDF ファイルが読取り可能な場合もそうでない場合もあります。Warnning のばあいは、その PDF ファイルは読み取れますが有効ではありません。

完全な PDF Tools のリストは、pdferror.h を参照してください。エラーコードは、以下のとおりです。

PDF_S_SUCCESS	その処理は成功しました。
PDF_E_EVAL	このソフトウエアは、評価版です。www.pdf-tools.com にコンタクトしてください。
PDF_E_FILEOPEN	ファイルをオープンできませんでした。
PDF_E_FILECREATE	ファイルを作成できませんでした。
PDF_E_PASSWORD	パスワードが不正で、認証に失敗しました。

TPDFPermission

TPDFPermission 列挙定数は、付与された許可を表します。許可しないものを設定しないことで、それを禁止できます。

ePermPrint	低解像度印刷
ePermModify	文書の変更
ePermCopy	文書内容のコピーや取り出し
ePermAnnotate	注釈
ePermFillForms	フォームフィールドを埋める
ePermSupportDisabilities	障害者サポート
ePermAssemble	文書アセンブリ
ePermDigitalPrint	高解像度印刷

TPDFRendererOption

レンディングのオプションは、*Options* プロパティを使って設定します。2つ以上のオプションを設定する場合は、それらを論理和（OR）します。

Visual Basic の例:

目的のオプションだけ有効または無効にします。

```
' 高品位を有効にする
.Options = .Options OR eOptionHighQuality
' 高品位を無効にする
.Options = .Options AND NOT eOptionHighQuality
```

C/C++の例:

```
int iOptions = Pdf2ImgGetOptions(pDocument);
// 高品位を有効にする
Pdf2ImgSetOptions(pDocument, iOptions | eOptionHighQuality);
// 高品位を無効にする
Pdf2ImgSetOptions(pDocument, iOptions & ~eOptionHighQuality);
```

以下に“3-Heights™ PDF to Image Converter API”に関連した列挙定数を示します。注意してください、多くの列挙型定数がありますが、これらはこの API の処理そのものには関係ありません。

eOptionBilinear	双線形画像フィルターは画像の品質を向上させるため画像に適用されます。このオプションは、*eOptionBicubic* と同時に指定できません。
eOptionBicubic	バイキュービック画像フィルターは画像の品質を向上させるため画像に適用されます。このオプションは、*eOptionBilinear* と同時に指定できません。最高の品質は、*eOptionHighQuality* と *eOptionBicubic* を組み合わせます。しかし、その場合は、多くの CPU 演算能力を必要とします。
eOptionDisablePatterns	パターンを無効にします。
eOptionHighQuality	文字、パスオブジェクトおよび、フィルタリングオブジェクトのアンチアイリアスを有効又は無効にします。これは、グレースケールおよびカラーの画像で推奨されています。しかし、2値画像にはお勧めできません。
eOptionType1	CFF フォントを Type1 フォントに変換します。
eOptionNoEmbedded	PDF 文書に含まれているフォントを使いません。その代わりに、システムにインストールされているフォントを使います。
eOptionOpenType	埋め込みフォントを Open Type フォントに変換します。
eOptionOutlines	フォントをベクター グラフィックスに変換します。
eOptionOverprint	このオプションを設定すると、PDF のコンテント ストリームの OverPrint (‘OP’ や‘op’) と OverPrint Mode (‘OPM’) 演算子を解釈します。それ以外の場合は、これらを無視します。この OverPrint シミュレーションは、完全にサポートされているわけではありませんので注意してください。分離カラーおよびデバイズ-n 色で期待する結果を生成しません。
eOptionPreInstalled	埋め込みフォントと同じフォントがシステムにインストールされている場合は、それと代替します。
eOptionPrint	このオプションを設定すると、PDF ファイルの外観が、実際に印刷されるか否かにかかわりなく、印刷出力に対応します。
eOptionTrueType	CFF と Type1 フォントを True Type フォントに変換します。このオプションは、*eOptionType1* に優先します。
eOptionUseFastImages	常に高速モードで画像を印刷します。文書の複雑な画像やイメージマスクを精密印刷モードで印刷する際のパフォーマンスの問題を解決するのに役立ちます。
eOptionTransparency	透明性をシミュレートします。

推奨する設定値

グレースケールおよびカラー: *eOptionBicubic* + *eOptionHighQuality* + *eOptionTrueType*

モノクロ: *eOptionTrueType*

TPDFRotateMode

eRotateNone	ページを回転させません、PDF ページの表示の回転属性を考慮しません。

eRotateAttribute	PDF ページの表示時回転属性を設定します。これによって、そのページがPDFビュワーによってレンダリングされる際に回転されます。
eRotatePortrait	縦置き(縦長)に回転されます。
eRotateLandscape	横置き(横長)に回転されます。

8 トラブルシューティング

8.1 テキスト

フォントの置き換え方針

この章では、レンダリングエンジンの正確なフォント処理動作を説明します。この説明はむしろ技術的であり、ソフトウエアを利用するのみであれば理解する必要はありません。

フォントの検索は、以下のステップで順次行われます。各ステップにおいて、もしフォントが検索できた場合は停止され、そうでなければ次が実行されます。

1. フォントが埋め込まれていない、または eOptionPreInstalled が有効になっている場合：
   a. フォント名が設定ファイル“fonts.ini”の[Replace]セクションに記載されている場合は、フォント名は指定された名前に代替され、その名前のフォントをインストールされているフォントのコレクションから検索します。
   b. フォント名が標準フォント[1]の場合は、そのフォントは同等のTrueTypeフォントに代替され、その名前のフォントをインストールされているフォントのコレクションから検索します。
   c. フォント名が設定ファイル“fonts.ini”の[Fonts]セクションに記載されている場合は、フォント名は指定された名前に代替され、その名前のフォントをインストールされているフォントのコレクションから検索します。
   d. フォント名が“Italic”（イタリック）や“Bold”（ボールド）といった装飾指定されている場合は、それらの装飾指定のないフォントをインストールされているフォントのコレクションから検索します。
2. フォント名がインストールされているフォントのコレクションから検索できた場合は、フォント名を以下のように比較します：
   a. PostScript 名
   b. 空白なしの TrueType 名（不足している装飾指定は、“Regular”または“Normal”と解釈されます）
   c. 修正なしの TrueType 名
3. フォントが埋め込まれている場合は、そのフォントがWindowsの互換フォントと変換され一時的にインストールされます。eOptionNoEmbeddedが指定された場合そのフォントのグリフ（字形）は、ビットマップまたはアウトライン[2]のいずれかに変換されます。
4. フォントが埋め込まれていない、かつ Unicode が有効な場合は、最も近いフォントがインストールされたフォントコレクションから検索され指定フォントのメトリックスに調整されます。
5. フォントが埋め込まれている場合は、それがアウトラインに変換されます。
6. 他の場合は、インストールされたフォントコレクションにある最も近いフォントが使われます。

[1] Times-Roman、Helvetica、Courier など

[2] グリフのアウトライン（輪郭）は、元のフォントプログラムへ参照しないベクトル型のグラフィックです。

フォントマッピングファイル“*fonts.ini*”を使用する

“font.ini”は、PDF で使われているフォントを、システムにプレインストールされているフォントにマップするための設定ファイルです。このマッピングファイルは、メインの DLL または実行可能ファイルが格納されたフォルダーの直接のサブフォルダーである“Fonts”という名前のフォルダーに格納されていなければなりません。

マッピングファイルはオプションで、2つのセクション[fonts]と[replace]で構成されています。

2つのセクションは、PDF のフォントをシステムにプリインストールされたフォントにマップするために使われます。これらは、PDF 文書が埋め込まれたフォントプログラムを持っていない場合または、埋め込みフォントが使用できない場合に働きます。

フォントがマッピングされるのは、指定されたフォントのフォントタイプが一致した場合だけです。例えば、PDF に“Symbol”や“ZapfdingBats”がある場合に、マップされるフォントはやはりシンボル フォントです。

[fonts]セクションは、フォント・マッチャが適切なフォントをインストールされたフォントの中から見つけられない場合に考慮されます。このような場合にのみこのセクションを利用しようとします。

[replace]セクションは、強力にフォント・マッチャより先に適用されようとします。これは、正しいプリインストールフォントがシステムで利用できる場合であっても、フォントが置き換えられることを意味します。

Syntax

マッピングファイルは、以下のように記述します。

```
[fonts]
PDF_font_1=installed_font_1, {font_style}
PDF_font_2=installed_font_2, {font_style}
[replace]
PDF_font_n=installed_font_n, {font_style}
```

PDF_font_* は、PDF にあるフォントの名前です。この名前は、以下のいずれかの方法で見つけることができます。

- PDF文書に使われているフォントを一覧するツール（*3-Heights PDF Extract*または*3-Heights PDF Optimization*）を使って探します。見つかったサブセットフォントの名前では無視できる接頭辞を無視します。接頭辞は“＋”（プラスサイン）で区切られた6文字で構成されます。例えば、名前が“KHFOKE+MonotypeCorsiva”であった場合は、“MonotypeCorsiva”をフォント名としてマッピングファイルに記載します。
- 文書を Adobe Acrobat で開きます、「文書テキストを編集  ツール」をつかってフォント名を知りたい文字列をマークします、マークされた文字列を右クリックします。表示されたメニューでプロパティを選択します。

installed_font_* は、インストールされたフォントのファミリー名です。この名前を取得するには、Windows のフォントフォルダーでそのフォントを探し、ダブルクリックして開きます。このときプロパティウィンドウの最初の行にフォントのファミリー名が表示されます（ただし、表示される内容や位置は、OS やそのバージョンに依存します）。フォントファミリーには、フォントスタイルが含まれません、例えば“Arial”はフォントファミリー名ですが、“Arial Itaric”はそうではありません。

font_style は、オプションのスタイルで、コンマで区切ってフォントファミリー名に追加します。このスタイルは、常に“Itaric”、“Bold”や“BoldItaric”のように単語で指定します。

例

```
[fonts]
Ryumin-Light=MS Mincho
GothicBBB-Medium=MS Gothic
```

```
[replace]
ArialIta=Arial,BoldItalic
```

“Deal with Fonts that Are Not Rendered Correctly”の対処方法

テキストが誤って印刷される場合の可能な回避策を以下に記します。回避策は、以下に記された順で試してください。

1. 埋め込みフォントがスプールファイルとプリンターハードウェアで使えなくなるように、*eOptionNoEmbedded* オプションを指定します。この場合は、グリフ（字形）がビットマップ又はアウトラインに変換されます。なお、同時に *eOptionOutline* オプションを指定すると、その変換はアウトラインに制限されます。
2. 埋め込みフォントに替わって、インストールされている同じ名前の（対応する）フォントが使われるように、*eOptionPreInstalled* オプションを指定します。
3. ビットマップをプリレンダリングし、プリレンダリング画像を使うようにします（*eOptionBitmap*）。

8.2 透明画像

3-Heigths™のレンダリングエンジンは、いくつかのブレンドモードと同じ様に隔離および非隔離グループの（一般的ではない）透明機能をサポートしています。

8.3 ページの向き

PDF 文書では、すべてのページに表示の向きの属性があります。この属性は、ビュワーで表示されたときの向きを定義したものです。もしこの設定どおりに向きを指定してレンダリングする場合は、プロパティを *RotateMode* = eRotateAttribute とセットします。