PDF Imager-LP

Version 1.8.3

株式会社トラスト・ソフトウェア・システム
２０１７年３月

目次

1.0 はじめに

PDF Imager-LP は、PDF（Portable Document Format）文書を画像に変換する機能をアプリケーションに追加するライブラリです。

PDFに指定されたカラーでの画像化だけではなく、グレースケール/二値画像やディザ変換した画像を生成できます。また、PDFで指定されたフォントを別のフォントに代替して変換できます。

2.0 利用環境およびPDFのバージョンと画像フォーマット

PDF Imager-LP は、以下の環境で利用できます。

利用環境	Windows Vista、7、8、10
開発環境	C/C++、C#、VB. NET
画像フォーマット	PNG（Portable Network Graphics）、JPEG（Joint Photographic Experts Group）、TIFF（Tagged Image File Format）形式 TIFF 形式では、非圧縮、Deflate 圧縮、JPEG 圧縮、LZW 圧縮および、CCITT FAX G3 または G4 圧縮を指定できます。

PDFのバージョン PDF 1.4 から PDF 1.7 を変換します。

PDF Imager-LP は、パスワードで暗号化されたPDF文書を画像に変換できます。（暗号化されたPDF文書を画像に変換するには、パスワードが必要です。）

3.0 インストール

PDF Imager-LP には、以下のフォルダーおよびファイルが含まれます。

`doc`	「PDF Imager-LP 説明書」および「使用許諾契約書」
`include`	C/C++で使用するためのヘッダファイル、他
`lib`	ライブラリ群
`sample`	C/C++、C#/VB.NET（Visual Studio 2008プロジェクト）サンプル コード

開発環境に応じて適切なフォルダーでご利用ください。

なお、PDF Imager-LP を使用するためには、適切なライセンスキーが必要です。

3.1 ファイルの概要

PDF Imager-LP に含まれるファイルの概要です。

`lib/PdfImagerLP.dll`	PDF画像変換のためのライブラリです。
`lib/PdfImagerLP.lib`	C/C++開発環境の場合にリンクして使用します。
`lib/PdfImagerLPNET.dll`	PDF画像変換機能を C#（または VB.NET）で利用するためのラッパーDLL です。
`include/ImagerLP.h`	C/C++用のヘッダファイルです。プロジェクトに含めて使用します。
`sample/init.xml`	代替フォントを指定する初期化ファイルの例です。

3.2 C/C++ インターフェース

ネイティブC/C++開発環境では、ヘッダファイル（`ImagerLP.h`）を利用できるようにし、ライブラリ（`PdfImagerLP.lib`）をリンクしてください。

3.3 .NET インターフェース

PDF画像変換ライブラリ(PdfImagerLP.dll)は、.NETアセンブリではありません。C#またはVB.NETから利用するための.NETアセンブリDLL(PdfImagerLPNET.dll)を参照して画像変換します。開発時においてはPdfImagerLPNET.dllを「参照設定」に追加する必要があります。

これらのDLLは、コンパイル・実行時において適切に参照できるようにしてください。

4.0 関数 – C/C++、.NETの開発環境

C/C++、および C#/VB.NET で利用する関数です。C#/VB.NET で利用する場合は、接頭辞の「Mlp」を除いた関数名に読み替えてください。

4.1 初期化

PDF Imager-LP ライブラリの使用に先立って初期化します。

なお、ライブラリは使用後に MlpUninitialize()を使って開放しなければなりません。

```
int MlpInitialize(
    TCHAR     *license
);
```

引数

license　　PDF Imager-LP を使用するためのライセンス文字列

戻り値

ライブラリ初期化に成功すると0(ゼロ)が戻ります。それ以外の場合は、エラーコードです。

4.2 ライセンス情報の表示または取得

PDF Imager-LP の初期化の後に、そのライセンスキーの内容を文字列で表示または、取得します。

MlpShowLicenseInfoは結果をコンソールに出力し、MlpLicenseInfoStrは結果を引数で与えられた領域に格納します。

```
void MlpShowLicenseInfo();

int MlpLicenseInfoStr(
    TCHAR     *licenseStrPtr
);
```

引数

licenseStrPtr　　ライセンスキーの内容(文字列)を格納する領域

戻り値

MlpShowLicenseInfo は、戻り値がありません。

MlpLicenseInfoStr は、成功するとライセンスを説明する文字列が戻ります。

4.3 初期化ファイル(または初期化データ)を指定する

PDF Imager-LP は、PDF文書で指定されたフォントの代替などを初期化ファイルまたは初期化データで指定できます。初期化ファイル(初期化データ)はXML形式です。

初期化ファイル(初期化データ)でのフォントの代替などの詳細は、「5.0 初期化ファイル(初期化データ)について」を参照してください。

```
int MlpLoadInitFile(
    TCHAR     *initFileName
);
```

```
int MlpLoadInitData(
    TCHAR     *data
    int       len
);
```

引数

initFileName	初期化ファイルのパス名
data	初期化データ（XML 形式データ）
len	初期化データのバイト数

戻り値

初期化ファイルを読み取ると0（ゼロ）が戻ります。それ以外の場合は、エラーコードです。

4.4 PDFファイルを開く

画像に変換するPDFファイルを開きます。
パスワードは、PDFファイルがパスワードで暗号化されている場合のみ使用され（他の場合は無視され）ます。

```
int MlpOpenDoc(
    const TCHAR *fileName,
    const TCHAR *ownerPassword,
    const TCHAR *userPassword
);
```

引数

fileName	PDF のファイルパス
ownerPassword	所有者パスワード
userPassword	ユーザーパスワード

戻り値

成功すると0（ゼロ）が戻り、失敗した場合はエラーコードが戻ります。

4.5 PDF文書の情報

PDF Imager-LP は、開いたPDF文書の情報を取得できます。

4.5.1 PDF文書のページ数取得

現在開いているPDF文書の総ページ数を取得します。

```
int MlpPageCount();
```

引数

ありません

戻り値

成功すると、PDF文書の総ページ数（0（ゼロ）の場合もあります）が戻り、失敗した場合はエラーコードが戻ります。なお、エラーコードは負数です。

4.5.2 PDF文書のパーミッション（許可）フラグ

PDF文書には、その文書を開いたユーザーに変更や印刷の可否を指定するフラグがあります。PDF Imager-LP はこの値を現在開いているPDF文書からパーミッション（許可）フラグ値として取得します。

```
int MlpGetDocumentInfo(MLP_PERMISSION_FLAG, int flag);
```

引数

flag	パーミッション（許可）フラグ値

戻り値

成功すると、0（ゼロ）が戻り、失敗した場合はエラーコードが戻ります。

パーミッションフラグの詳細は、「PDF Reference」 3.5.2 Standard Security Handler を参照してください。
パーミッションフラグ値は、PDF文書の「Standard Security Handler」ディクショナリ内のPキーに指定された整数値です。

4.6 画像に変換する関数群

JPEG形式の画像およびPNG形式の画像の場合は、PDF文書の1つのページを1つの画像に変換します。TIFF形式画像の場合は、PDF文書の1つのページを1つの画像に変換することや、PDF文書の複数のページを1つの画像ファイルに変換することができます。以下は、画像変換用の関数名および変換できる画像形式です。

関数名	変換できる画像形式
MlpCreatePict	PNG、JPEG、TIFF
MlpPageToPict	PNG、JPEG、TIFF
MlpConvertToTiff	TIFF
MlpCreateTiffRange	TIFF
MlpCreateTiffMulti	TIFF
MlpRangeToTiff	TIFF
MlpMultiPageToTiff	TIFF
MlpCreateMem	ピクセルデータ

4.6.1 指定された1ページを画像に変換（解像度と画像品質を指定しない）

PDF文書の指定されたページを画像に変換します。
解像度と画像品質は、省略時解釈値（デフォルト値）または画像の変換に先立って指定された値が使用されます。画像の解像度や品質を指定する場合は、MlpSetPictureResolution、MlpSetPictureQuality、MlpSetPicture関数などを利用してください。
ページ番号の指定では、最初のページを1とします。負数のページ番号は、PDF文書の最終ページとみなされます。
変換する画像ファイルの拡張子で画像の形式が自動で判断されます。

```
int MlpCreatePict(
    int         pageNumber,
    TCHAR       *outputFilePath
);
```

引数

pageNumber	画像に変換するPDF文書のページ番号（先頭のページ番号は1です。）
outputFilePath	画像のファイルパス 画像形式はファイルの拡張子によって以下のように自動で選択されます。 拡張子が“png”の場合、PNG形式の画像 拡張子が“jpeg”または“jpg”の場合、JPEG形式の画像 拡張子が“tiff”または“tif”の場合、TIFF形式の画像

戻り値

成功すると0（ゼロ）が戻り、それ以外の場合はエラーコードが戻ります。

4.6.2 指定された1ページを画像に変換（解像度と画像品質を指定する）

PDF文書の指定されたページを画像に変換します。

画像に変換する際に、指定された解像度および画像品質を使用します。ページ番号は、先頭のページを1とします。

ページ番号に負数を指定しますと、PDF文書の最終ページが画像に変換されます。

変換する画像ファイルの拡張子で画像の形式が判断されます。

```
int MlpPageToPict(
    int         pageNumber,
    int         dpi,
    int         quality,
    TCHAR       *outputFilePath
);
```

引数

pageNumber	画像に変換するPDF文書のページ番号（先頭のページ番号は、1です。）
dpi	生成される画像の 1 インチあたりの画素数（生成される画像の解像度）を MIN_RESOLUTION 以上かつ MAX_RESOLUTION 以下で指定します。範囲外の値を指定すると MIN_RESOLUTION または MAX_RESOLUTION に正規化されます。
quality	JPEG圧縮の画像品質を MIN_QUALITY 以上かつ MAX_QUALITY までの数値で指定します。範囲外の値を指定すると MIN_QUALITY または MAX_QUALITY に正規化されます。 PNG形式画像を指定した場合は無視されます。
outputFilePath	画像のファイルパス 画像形式はファイルの拡張子によって以下のように自動で選択されます。 拡張子が“png”の場合、PNG形式の画像 拡張子が“jpeg”または“jpg”の場合、JPEG形式の画像 拡張子が“tiff”または“tif”の場合、TIFF形式の画像

戻り値

成功すると0（ゼロ）が戻り、それ以外の場合はエラーコードが戻ります。

4.6.3 すべてのページをTIFF形式画像に変換

PDF文書のすべてのページを1つのTIFF形式画像ファイルに変換します。

```
int MlpConvertToTiff (
    TCHAR       *outputFilePath
);
```

引数

outputFilePath	画像のファイルパス TIFF形式の画像が作成されます。必要に応じて拡張子「.tiff」が付加されます。

戻り値

成功すると、0（ゼロ）が戻り、失敗するとエラーコードが戻ります。

4.6.4 ページ範囲を指定してTIFF形式画像に変換

指定された範囲のPDFページをひとつのTIFF画像に変換します。（指定された範囲内のページをすべて含む1つのTIFF画像に変換されます）。ただし、PDF文書にないページを指定すると失敗します。

成功すると、0（ゼロ）が戻り、失敗するとエラーコードが戻ります。
ページ番号0を指定すると最初のページ（ページ番号1）、ページ番号が負数の場合は最終ページとみなされます。したがって、開始ページ=1 終了ページ=-1 と指定すると、最初のページから最後のページまですべてのページをTIFF画像に変換します。
ページは逆順（大きいページ番号から小さいページ番号の順）でも指定できます。

```
int MlpCreateTiffRange(
    int         startPageNumber,
    int         endPageNumber,
    TCHAR       *outputFilePath
);
```

引数

startPageNumber	画像に変換する最初のPDF文書のページ番号（先頭のページ番号は、1です。）
endPageNumber	画像に変換する最後のPDF文書のページ番号
outputFilePath	画像のファイルパス TIFF形式の画像のみが作成されます。必要に応じて拡張子「.tiff」が付加されます。

戻り値
成功すると、0（ゼロ）が戻り、失敗するとエラーコードが戻ります。

4.6.5 ページのリストを指定してTIFF形式画像に変換

リストで指定されたPDF文書のページをひとつのTIFF画像に変換します。
画像に変換するページを区切り文字（スペース、タブまたはコンマ）で区切ってページを指定すると、指定した順に画像に変換された複数ページで構成されたTIFF画像が生成されます。このとき、PDF文書に無いページを指定するとその指定は無視されます。成功すると、0（ゼロ）が戻り、失敗するとエラーコードが戻ります。
ページの指定の詳細は、「6.0 文字列でのページ指定方法」を参照してください。

```
int MlpCreateTiffMulti(
    const TCHAR*pageList,
    TCHAR       *outputFilePath
);
```

引数

pageList	画像に変換するページのリスト。画像に変換するページを区切り文字で区切って1っ以上のページを指定します。PDF文書にないページは無視されます。 ページ番号 0 は最初のページ、ページ番号-1 は最後のページに変換されます。 詳細は、「6.0 文字列でのページ指定方法」を参照してください。
outputFilePath	画像のファイルパス TIFF形式の画像のみが作成されます。必要に応じて拡張子「.tiff」が付加されます。

戻り値
成功すると、0（ゼロ）が戻り、失敗するとエラーコードが戻ります。

4.6.6 ページ範囲を指定してTIFF形式画像に変換（解像度を指定する）

指定された範囲のPDFページをひとつのTIFF画像に変換します。指定された範囲内のページをすべて含む1つのTIFF画像に変換されます。ただし、PDF文書にないページを指定すると失敗します。
ページ番号0を指定すると最初のページ（ページ番号1）、ページ番号が負数の場合は最終ページとみなされます。したがって、開始ページ=1 終了ページ=-1 と指定すると、最初のページから最後のページまですべてのページをTIFF画像に変換します。
ページは逆順（大きいページから小さいページ番号の順）でも指定できます。

```
int MlpRangeToTiff(
    int         startPageNumber,
    int         endPageNumber,
    int         dpi,
    TCHAR       *outputFilePath
);
```

引数

startPageNumber	画像に変換する最初のPDF文書のページ番号（最初のページは、1です。）
endPageNumber	画像に変換する最後のPDF文書のページ番号（最後のページを、-1 と指定できます。）
dpi	生成される画像の 1 インチあたりの画素数（生成される画像の解像度）を MIN_RESOLUTION 以上かつ MAX_RESOLUTION 以下で指定します。範囲外の値を指定すると MIN_RESOLUTION または MAX_RESOLUTION に正規化されます。
outputFilePath	画像のファイルパス TIFF形式の画像のみが作成されます。必要に応じて拡張子「.tiff」が付加されます。

戻り値

成功すると、0（ゼロ）が戻り、失敗するとエラーコードが戻ります。

4.6.7 ページのリストを指定してTIFF形式画像に変換（解像度を指定する）

リストで指定されたPDFページをひとつのTIFF画像に変換します。

画像に変換するページを区切り文字（スペース、タブまたはコンマ）で区切ってページを指定すると、指定した順に画像に変換された複数ページで構成されたTIFF画像が生成されます。このとき、PDF文書に無いページを指定するとその指定は無視されます。

ページの指定の詳細は、「6.0 文字列でのページ指定方法」を参照してください。

```
int MlpMultiPageToTiff(
    const TCHAR*pageList,
    int         dpi,
    TCHAR       *outputFilePath
);
```

引数

pageList	画像に変換するページのリスト。画像に変換するページを区切り文字で区切って1っ以上のページを指定します。PDF文書にないページは無視されます。 ページ番号 0 は最初のページ、ページ番号-1 は最後のページに変換されます。 詳細は、「6.0 文字列でのページ指定方法」を参照してください。
dpi	生成される画像の 1 インチあたりの画素数（生成される画像の解像度）を MIN_RESOLUTION 以上かつ MAX_RESOLUTION 以下で指定します。範囲外の値を指定すると MIN_RESOLUTION または MAX_RESOLUTION に正規化されます。
outputFilePath	画像のファイルパス TIFF形式の画像のみが作成されます。必要に応じて拡張子「.tiff」が付加されます。圧縮設定は指定できません。

戻り値

成功すると、0（ゼロ）が戻り、失敗するとエラーコードが戻ります。

4.6.8 指定されたページをピクセルデータに変換

指定されたページをピクセルデータに変換し、そのピクセルデータを取得します。

変換されたピクセルデータにアルファチャネルは含みません。各ピクセルは RGB（Red、Green、Blue；カラーの場合）値がこの順番でまたは、グレースケール（白黒の場合）値が格納されます。

変換されたピクセルデータは、MlpGetPictFromMem を使って取得することもできます。

```
int MlpCreatePictBufPage(
    int             *pageNumber,
    unsigned TCHAR **buf,
    int             *width,
    int             *height,
    int             *nOfColors
);
```

引数

pageNumber	ピクセルデータに変換するページの番号。
buf	ピクセルデータ（一次元配列） このアドレスは一時的ですので、記録しておいて後で使うことはできません。
width	画像に変換されたピクセルの幅
height	画像に変換されたピクセルの高さ
nOfColors	画像に変換されたピクセルの1画素あたりの色数

戻り値

成功すると、0（ゼロ）が戻り、失敗するとエラーコードが戻ります。

4.6.9 連続する複数ページをピクセル画像に変換

複数のページを連続して画像に変換します。

ページは、開始のページ番号と終了ページの番号を指定します。変換されたピクセルデータは、ライブラリ内部に格納されます。ピクセルデータは、インデックス番号（指定されたページ順に0から割り振られます）とページ番号で管理されます。格納されたピクセルデータは、MlpPictMemWIndex または MlpPictMemWPageNum を使って取得します。

MlpCreatePictBufPageRange は、ページ画像への変換を並列に処理するため、画像変換用のメモリーを多く必要とします。メモリーの使用量を少なくする場合は、並列に処理しないメソッドを利用するまたは並列処理を抑制してください。

```
int MlpCreatePictBufPageRange(
    int             *firstPageNumber,
    int             *lastPageNumber
);
```

引数

firstPageNumber	ピクセルデータに変換を開始するページの番号。
lastPageNumber	ピクセルデータに変換を終了するページの番号。

戻り値

成功すると、0（ゼロ）が戻り、失敗するとエラーコードが戻ります。

4.6.10 不連続な複数ページをピクセル画像に変換

連続していない複数のページを画像に変換します。

ページは、配列で指定します。変換されたピクセルデータは、ライブラリ内部に格納されます。ピクセルデータは、インデックス番号（指定されたページ順に0から割り振られます）とページ番号で管理されます。格納されたピクセルデータは、MlpPictMemWIndex または MlpPictMemWPageNum を使って取得します。

MlpCreatePictBufPageArray は、ページ画像への変換を並列に処理するため、画像変換用のメモリー

を多く必要とします。メモリーの使用量を少なくする場合は、並列に処理しないメソッドを利用するまたは並列処理を抑制してください。

```
int MlpCreatePictBufPageArray(
     int            *pageNumberArray,
     int            length
);
```

引数

pageNumberArray	ピクセルデータに変換するページの番号が格納された配列
length	配列に格納されたページの総数

戻り値

成功すると、0（ゼロ）が戻り、失敗するとエラーコードが戻ります。

4.6.11 指定されたページをライブラリ内部データに変換

指定されたページをライブラリ内部の画像データに変換します。

生成された画像は MlpGetPictFromaMem で取り出したり、MlpMemPictToPanel や MlpMemPictToPanelBBox でパネルに描画したりできます。

```
int MlpCreatePictMem(
     int            *pageNumber,
);
```

引数

pageNumber	ピクセルデータに変換するページの番号。

戻り値

成功すると、0（ゼロ）が戻り、失敗するとエラーコードが戻ります。

4.6.12 ライブラリ内部の画像データを取得

ライブラリ内部に生成された画像データを取得します。この関数で取得できる画像データは、単一のページを画像に変換したデータです。複数のページを画像に変換した場合（MlpCreatePictBufPageRange を使った場合）の画像を取り出す場合は、MlpPictMemWIndex または MlpPictMemWPageNum を使用します。

変換されたピクセルデータにアルファチャネルは含みません。各ピクセルは RGB（Red、Green、Blue；カラーの場合）値がこの順番でまたは、グレースケール（白黒の場合）値が格納されます。

```
int MlpGetPictFromMem(
     unsigned TCHAR **pixels,
     int            *width,
     int            *height,
     int            *nOfColors
);
```

引数

pixels	ピクセルデータ（一次元配列） C/C++開発環境で利用する場合、このアドレスは一時的ですので後で使うことはできません。
width	画像に変換されたピクセルの幅
height	画像に変換されたピクセルの高さ
nOfColors	画像に変換されたピクセルの1画素あたりの色数

戻り値

成功すると、0（ゼロ）が戻り、失敗するとエラーコードが戻ります。

4.6.13 ライブラリ内部の画像データをインデックスで取得

MlpCreatePictBufPageRange で作成されたピクセルデータをインデックスで取得します。

```
int MlpGetPictFromMemIndex(
     int             index,
     unsigned TCHAR **pixels,
     int             *width,
     int             *height,
     int             *nOfColors,
     int             *pageNumber
);
```

引数

index	PDF文書のページを連続して画像に変換した場合にそれぞれのページに自動で割り振られた 0 から始まる番号
pixels	ピクセルデータ(一次元配列) C/C++開発環境で利用する場合、このアドレスは一時的ですので後で使うことはできません。
width	画像に変換されたピクセルの幅
height	画像に変換されたピクセルの高さ
nOfColors	画像に変換されたピクセルの1画素あたりの色数
pageNumber	PDF文書のページ番号

戻り値

成功すると、0(ゼロ)が戻り、失敗するとエラーコードが戻ります。

4.6.14 ライブラリ内部の画像データをページ番号で取得

MlpCreatePictBufPageRange で作成されたピクセルデータをPDF文書のページ番号で取得します。

```
int MlpGetPictFromMemPage(
     int             pageNumber,
     unsigned TCHAR **pixels,
     int             *width,
     int             *height,
     int             *nOfColors,
);
```

引数

pageNumber	PDF文書のページ番号
pixels	ピクセルデータ(一次元配列) C/C++開発環境で利用する場合、このアドレスは一時的ですので後で使うことはできません。
width	画像に変換されたピクセルの幅
height	画像に変換されたピクセルの高さ
nOfColors	画像に変換されたピクセルの1画素あたりの色数

戻り値

成功すると、0(ゼロ)が戻り、失敗するとエラーコードが戻ります。

4.6.15 ライブラリ内部の画像データを削除

ライブラリ内部の画像データを削除し、そのメモリー領域を開放します。

この画像データ領域は、現在のPDF文書を閉じる際またはライブラリの使用終了で自動的に削除されるためメモリー領域に問題がない場合は開放しなくともかまいません。

```
int MlpFreePictMem();
```

引数
　ありません
戻り値
　成功すると、0（ゼロ）が戻り、失敗するとエラーコードが戻ります。

4.7 出力画像の色指定をする

PDF文書がカラーで構成されている場合でも、グレースケール、白黒ディザ、白黒（2階調）画像に変換します。
色の指定を省略すると、RGBカラーの画像が生成されます。
TIFF画像に変換する際は、CCITT Group3または Group4圧縮 Fax ファイルを生成できます。

4.7.1 RGBカラー画像を指定

PDF文書をRGBカラーの画像に変換するよう設定します。
この設定後には、RGBカラー画像が生成されます。PDF文書がRGBカラーでなかった場合でも、画像データはRGBカラー（1画素あたり24ビットで構成された画像データ）に変換されます。
RGBカラー画像指定は、既定値です。

```
int MlpSetPictureRGB();

int MlpSetPicture(MLP_PICTURE_RGB);
```

引数
　MlpSetPictureRGB には、ありません。
　MlpSetPicture では、MLP_PICTURE_RGB を指定します。
戻り値
　成功すると、0（ゼロ）が戻り、失敗するとエラーコードが戻ります。

4.7.2 グレースケール画像を指定

PDF文書をグレースケールの画像に変換するよう設定します。

```
int MlpSetPictureGray();

int MlpSetPicture(MLP_PICTURE_GRAY);
```

引数
　MlpSetPictureGray には、ありません。
　MlpSetPicture では、MLP_PICTURE_GRAY を指定します。
戻り値
　成功すると、0（ゼロ）が戻り、失敗するとエラーコードが戻ります。

4.7.3 白黒ディザ画像を指定

PDF文書を白黒ディザ（Floyd-Steinberg ディザリング）の画像に変換するよう設定します。

```
int MlpSetPictureDither();
```

```
int MlpSetPicture(MLP_PICTURE_DITHER);
```

引数
MlpSetPictureDither には、ありません。
MlpSetPicture では、MLP_PICTURE_DITHER を指定します。
戻り値
成功すると、0（ゼロ）が戻り、失敗するとエラーコードが戻ります。

4.7.4 白黒2階調画像を指定

PDF文書を白黒2階調の画像に変換するよう設定します。

```
int MlpSetPictureBW();

int MlpSetPicture(MLP_PICTURE_BW);
```

引数
MlpSetPictureBW には、ありません。
MlpSetPicture では、MLP_PICTURE_BW を指定します。
戻り値
成功すると、0（ゼロ）が戻り、失敗するとエラーコードが戻ります。

4.7.5 CCITT Group3 圧縮Faxファイルを指定

PDF文書を CCITT Group3 圧縮のFaxファイルに変換するよう設定します。

```
int MlpSetPictureFaxG3();

int MlpSetPicture(MLP_PICTURE_FAXG3);
```

引数
MlpSetPictureBW には、ありません。
MlpSetPicture では、MLP_PICTURE_FAXG3 を指定します。
戻り値
成功すると、0（ゼロ）が戻り、失敗するとエラーコードが戻ります。

4.7.6 CCITT Group4 圧縮Faxファイルを指定

PDF文書を CCITT Group4 圧縮のFaxファイルに変換するよう設定します。

```
int MlpSetPictureFaxG4();

int MlpSetPicture(MLP_PICTURE_FAXG4);
```

引数
MlpSetPictureBW には、ありません。
MlpSetPicture では、MLP_PICTURE_FAXG4 を指定します。
戻り値
成功すると、0（ゼロ）が戻り、失敗するとエラーコードが戻ります。

4.8 画像のサイズ

PDF文書では、1 ポイントを 1/72 インチとしています。PDF Imager-LP は、この 1 ポイントを指定された画像

解像度に変換して画像を生成します。そのため、大きな値の画像解像度を指定すると大きなピクセルサイズの画像が生成され、小さな画像解像度では小さなピクセルサイズの画像が生成されます。

キャンバスとは、画像を配置する領域です。生成された画像は、このキャンバスに貼り付けられます。キャンバスの領域が生成された画像よりも小さい場合は画像の一部が切り取られ、逆に大きな領域を指定すると余白の付加された画像になります。

生成された画像は、画像の解像度から算出された画像の論理サイズを持っています。論理サイズを指定のサイズに設定できます、がこの指定によって画像のピクセルサイズは変わりません。論理サイズの既定値は、PDF文書に記載されたページの大きさです。

4.8.1 ページの大きさ

PDF文書では、ページごとにその大きさを持っています。さらに、このページの大きさ（詳細は、『PDF Reference』を参照してください。）を表すために5種類の大きさ（MediaBox、CropBox、BleedBox、TrimBox、ArtBox）が用意されています。PDF Imager-LP は、ページの大きさを決定する際の境界を指定できます。

```
int MlpSetPicture(
    int       opt
);
```

引数

opt	利用する大きさの種類を指定します。	
	MLP_PICTURE_USE_MEDIABOX	MediaBox で指定された大きさを使います。（既定値）
	MLP_PICTURE_USE_CROPBOX	CropBox で指定された大きさを使います。 PDF文書に指定されていない場合は、MediaBox を使います。
	MLP_PICTURE_USE_BLEEDBOX	BleedBox で指定された大きさを使います。 PDF文書に指定されていない場合は、CropBox を使います。
	MLP_PICTURE_USE_TRIMBOX	TrimBox で指定された大きさを使います。 PDF文書に指定されていない場合は、CropBox を使います。
	MLP_PICTURE_USE_ARTBOX	ArtBox で指定された大きさを使います。 PDF文書に指定されていない場合は、CropBox を使います。

4.8.2 画像解像度の設定

PDF文書を画像に変換する際の 1 ポイントを指定のピクセル数で構成するよう指定します。このときに指定する数値を画像解像度（または、解像度）といいます。既定の解像度は、150DPI（Dot Per Inch）です。

解像度は、X方向、Y方向で同じ値が設定されます。設定する値が大きい場合は、処理に必要とするメモリーが多くなり、処理時間も増大します。

指定を省略すると、150DPI で画像が生成されます。

```
int MlpSetPictureResolution(
    int       dpi
);

int MlpSetPicture(
    MLP_RESOLUTION_DPI,
    int       dpi
);
```

引数

dpi　　解像度を DPI(Dot per Inch)単位で指定します。
解像度は、MIN_RESOLUTION 以上かつ MAX_RESOLUTION 以下を指定します。範囲外の値を指定すると、MIN_RESOLUTION または MAX_RESOLUTION に正規化されます。

MlpSetPicture では、MLP_RESOLUTION_DPI とともに解像度を指定します。

戻り値

成功すると、0(ゼロ)が戻り、失敗するとエラーコードが戻ります。

4.8.3 画像のピクセルサイズを取得

指定された解像度で生成される画像のピクセルサイズを取得します。

```
int MlpGetPicturePixel(
    int        pageNum,
    int        *x,
    int        *y
);
```

引数

pageNum　　サイズを取得するページの番号
x　　画像の幅(ピクセル単位)
y　　画像の高さ(ピクセル単位)

戻り値

成功すると、0(ゼロ)が戻り、失敗するとエラーコードが戻ります。

なお、PDF文書では、ページごとにその物理的な大きさが"MeiaBox"(PDF Reference 3.6.2 Page Tree 参照)として記載されています。PDF文書での解像度 72DPI を指定して所得する画像のピクセルサイズは、この"Media Box"のサイズです。

4.8.4 画像幅のピクセルサイズを取得

指定された解像度で生成される画像幅のピクセルサイズを取得します。

```
int MlpGetPicturePixelX(
    int        pageNum
);
```

引数

pageNum　　サイズを取得するページの番号

戻り値

成功すると画像の幅がピクセル数で戻り、失敗するとエラーコード(負数)が戻ります。

なお、PDF文書では、ページごとにその物理的な大きさが"MeiaBox"(PDF Reference 3.6.2 Page Tree 参照)として記載されています。PDF文書での解像度 72DPI を指定して所得する画像のピクセルサイズは、この"Media Box"のサイズです。

4.8.5 画像高さのピクセルサイズを取得

指定された解像度で生成される画像高さのピクセルサイズを取得します。

```
int MlpGetPicturePixelY(
    int        pageNum
);
```

引数

pageNum	サイズを取得するページの番号

戻り値

成功すると画像の高さがピクセル数で戻り、失敗するとエラーコード（負数）が戻ります。

なお、PDF文書では、ページごとにその物理的な大きさが”MeiaBox”（PDF Reference 3.6.2 Page Tree 参照）として記載されています。PDF文書での解像度 72DPI を指定して所得する画像のピクセルサイズは、この”Media Box”のサイズです。

4.8.6 画像のピクセルサイズ指定

生成される画像の大きさをピクセル単位で指定します。ただし、指定できるのは幅（X方向）または、高さ（Y方向）のみです。

```
int MlpSetPicturePixelX(
    int         x
);

int MlpSetPicturePixelY(
    int         y
);
```

引数

x	画像の幅をピクセル単位で指定します。 0（ゼロ）を指定すると、既定の大きさになります。 高さ（Y方向）は、自動で計算されます。
y	画像の高さをピクセル単位で指定します。 0（ゼロ）を指定すると、既定の大きさになります。 幅（X方向）は、自動で計算されます。

戻り値

成功すると、0（ゼロ）が戻り、失敗するとエラーコードが戻ります。

4.8.7 キャンバスサイズの設定

PDF文書から変換された画像を配置する仮想の領域（キャンバス）のサイズを指定します。

キャンバスサイズを変更しても画像の解像度は変化しません。そのため、変換された画像よりも小さな領域を指定すると画像が切り取られ、大きな領域を指定すると画像に余白が付加されます。

生成された画像は、キャンバスの左上に配置され、余白は白色で塗りつぶされます。

指定を省略すると、生成される画像のピクセルサイズとキャンバスのサイズは同じになります。

```
int MlpSetCanvasSize(
    int         x,
    int         y
);
```

引数

x	キャンバスの幅をピクセル単位で指定します。 0（ゼロ）を指定すると、変換された画像の幅と同じになります。
y	キャンバスの高さをピクセル単位で指定します。 0（ゼロ）を指定すると、変換された画像の高さと同じになります。

戻り値

成功すると、0（ゼロ）が戻り、失敗するとエラーコードが戻ります。

4.8.8 画像の論理サイズ指定

PDF Imager-LP で生成する画像データ（TIFF 形式および JPEG 形式画像）は、そのピクセルサイズとは別に、論理的な大きさを持っています。論理的な大きさは、画像データのプロパティに含まれる解像度と画像のピクセルサイズから算出されます。PDF Imager-LP では、この算出された画像の大きさを画像の論理サイズといいます。

次の関数では、この論理サイズを指定します。ただし、記載される解像度の丸め誤差によって、生成された画像の論理サイズにも誤差が生じます。そのため、画像の形式にもよりますが、画像への変換時に指定した論理サイズと、生成された画像の論理サイズに差異が生じます。

```
int MlpSetPictureSize(
    int         x,
    int         y,
    int         unit
);
```

引数

x	論理サイズの幅を指定します。	
y	論理サイズの高さを指定します。	
unit	指定した論理サイズの単位を指定します。 以下の値を指定します。	
	MLP_PICTURE_UNIT_PERCENT	値をパーセントで指定
	MLP_PICTURE_UNIT_INCH	値をインチで指定
	MLP_PICTURE_UNIT_MM	値をミリメートルで指定

論理サイズは、幅もしくは高さ いずれかのみ設定できます。両方を指定した場合は、幅が指定されます。高さを指定する場合は、幅に0（ゼロ）を指定します。

戻り値

成功すると、0（ゼロ）が戻り、失敗するとエラーコードが戻ります。

4.8.9 ページの境界ボックスのサイズを取得

PDF文書では、そのページのサイズを表すのに、Media box、Crop box、Bleed box、Trim boxおよびArt box を使用します。なお、Media box は必須で指定されますが、他は省略される場合があります。

次の関数では、この境界ボックスのサイズを所得します。

MlpGetBoundary 関数を使うと、各境界ボックスの値を既定値で補完した値を取得しますが、MlpGetRawBoundary 関数では PDF 文書から読み取った値をそのまま取得します。

```
int MlpGetBoundary(
    int         pageNum,
    int         kind,
    float       *userUnit,
    float       *left,
    float       *bottom,
    float       *right,
    float       *top
);

int MlpGetRawBoundary(
    int         pageNum,
    int         kind,
    float       userUnit,
    float       *left,
    float       *bottom,
    float       *right,
```

```
    float      *top
);
```

引数

pageNum	値を取得するページの番号を指定します。
kind	取得する境界ボックスの種類を指定します。 以下の値を指定します。 MEDIA_BOX　Media boxのサイズを取得するよう指定 CROP_BOX　Crop box のサイズを取得するよう指定 BLEED_BOX　Bleed box のサイズを取得するよう指定 TRIM_BOX　Trim boxのサイズを取得するよう指定 ART_BOX　Art box のサイズを取得するよう指定
userInit	UserUnit を取得します。(既定値は 1.0)
left	左辺のx座標位置を取得
bottom	下辺のy座標位置を取得
right	右辺のx座標位置を取得
top	上辺のy座標位置を取得

戻り値

成功すると、0(ゼロ)が戻り、失敗するとエラーコードが戻ります。

4.8.10 ページの回転角を取得

PDF 文書では、ページが回転されている場合があります。
以下の関数で、その回転角を取得します。
なお、回転角は0°、90°、180°、270°のいずれかです。

```
int MlpGetPageRotation(
    int        pageNum,
    int        *rotate
);
```

引数

pageNum	値を取得するページの番号を指定します。
rotate	回転の角度を取得します。

戻り値

成功すると、0(ゼロ)が戻り、失敗するとエラーコードが戻ります。

4.9 TIFF形式画像の圧縮指定

TIFF画像を生成する場合の圧縮方法を指定します。

4.9.1 圧縮しない

TIFF形式画像を生成する場合に圧縮をしません。

```
int MlpTiFFCompress(
    MLP_TIFF _COMPRESS_NONE
);

int MlpSetPicture(
    MLP_COMPRESS_NONE
);
```

引数

MLP_TIFF_COMPRESS_NONE を指定します。(MlpTiFFCompress の場合)
MLP_COMPRESS_NONE を指定します。(MlpSetPicture の場合)

戻り値

成功すると、0(ゼロ)が戻り、失敗するとエラーコードが戻ります。

4.9.2 JPEG圧縮指定

TIFF形式画像を生成する場合にJPEG圧縮をします。
3番目の指定形式では、JPEG 圧縮の品質を同時に指定します。ここで指定する品質は、画像形式にJPEGを選択した場合にも有効になります。

```
int MlpTiffCompress(
    MLP_TIFF_COMPRESS_JPEG
);

int MlpSetPicture(
    MLP_COMPRESS_JPEG
);

int MlpSetPicture(
    MLP_COMPRESS_JPEG,
    int         quality
);
```

引数

MLP_TIFF_COMPRESS_JPEG を指定します。(MlpTiffCompress の場合)
MLP_COMPRESS_JPEG を指定します。(MlpSetPicture の場合)
quality　JPEG 圧縮画像の品質を MIN_QUALITY 以上かつ MAX_QUALITY 以下で指定します。
範囲外の値を指定した場合は、MIN_QUALITY または MAX_QUALITY に正規化されます。

戻り値

成功すると、0(ゼロ)が戻り、失敗するとエラーコードが戻ります。

4.9.3 Deflate圧縮指定

TIFF形式画像を生成する場合にDeflate圧縮をします。

```
int MlpTiffCompress(
    MLP_TIFF_COMPRESS_DEFLATE
);

int MlpSetPicture(
    MLP_COMPRESS_DEFLATE
);
```

引数

MLP_TIFF_COMPRESS_DEFLATE を指定します。(MlpTiffCompress の場合)
MLP_COMPRESS_DEFLATE を指定します。(MlpSetPicture の場合)

戻り値

成功すると、0(ゼロ)が戻り、失敗するとエラーコードが戻ります。

4.9.4 LZW圧縮指定

TIFF形式画像を生成する場合にLZW圧縮をします。

```
int MlpTiffCompress(
    MLP_TIFF_COMPRESS_LZW
);

int MlpSetPicture(
    MLP_COMPRESS_LZW
);
```

引数

MLP_TIFF_COMPRESS_LZW を指定します。(MlpTiffCompress の場合)
MLP_COMPRESS_LZW を指定します。(MlpSetPicture の場合)

戻り値

成功すると、0(ゼロ)が戻り、失敗するとエラーコードが戻ります。

4.10 JPEG圧縮の品質指定

JPEG圧縮が指定された場合の画像品質を指定します。画像データの圧縮が指定されない場合やJPEG圧縮以外の圧縮が指定された場合は、この指定は無視されます。

```
int MlpSetPictureQuality(
    int         quality
);

int MlpSetPicture(
    MLP_JPEG_QUALITY,
    int         quality
);
```

引数

quality	JPEG圧縮の際の品質をMIN_QUALITY 以上かつMAX_QUALITY 以下の値で指定します。範囲外の値を指定すると、MIN_QUALITY または MAX_QUALITY に正規化されます。

戻り値

成功すると、0(ゼロ)が戻り、失敗するとエラーコードが戻ります。

4.11 変換の詳細を指定する

4.11.1 塗りつぶしパスのアンチアイリアスを抑制する

PDF Image-LP は、パスや文字描画の際にその境界を滑らかにするため、アンチエイリアス処理を施しています。そのため、小さなパスを組み合わせて大きなパス画像にする場合などで、その下地となる画像や文字が透けてしまう場合があります。このような場合は、塗りつぶしパスのアンチエイリアスを抑制して下地が透けて見えるのを防ぐことができます。

なお既定では、塗りつぶしパスのアンチエイリアスは実施します。

```
int MlpSetFillPathAADisable();

int MlpSetPicture(MLP_FILL_PATH_AA, 0);
```

戻り値
成功すると、0（ゼロ）が戻り、失敗するとエラーコードが戻ります。

4.11.2 塗りつぶしパスのアンチアイリアスを実施する

塗りつぶしパスのアンチエイリアスを抑制すると、それ以降の変換で抑制されますので、以下でその抑制を中止します。

```
int MlpSetFillPathAAEnable();

int MlpSetPicture(MLP_FILL_PATH_AA, 1);
```

戻り値
成功すると、0（ゼロ）が戻り、失敗するとエラーコードが戻ります。

4.11.3 文字コード0（ゼロ）の表示

PDF Image-LPは、表示文字のコードが0（ゼロ）の場合には、未定義グリフ [1]の替わりにスペースが表示されます。
この動作を以下の手順で変更およびその設定状態を取得できます。

```
int MlpSetCid0GlyphMode(                    /*設定*/
    int mode);
int MlpGetCid0GlyphMode();                  /*取得*/

int MlpSetPicture(                          /*設定*/
    MLP_CID__GLYPH,
    int        mode
);
int MlpGetLibraryInfo(                      /*取得*/
    MLP_CID__GLYPH,
    int        mode
);
```

引数

mode	以下のグリフ表示モードを指定します。	
	SHOW_NOTDEF_GLYPH	未定義グリフを表示
	REPLACE_TO_BLANK_GLYPH	空白（Space）グリフを表示（既定の動作です。）
	IGNORE_CID_0_GLYPH	CID=0 の文字表示を無視します。

戻り値
設定の場合は、成功すると0（ゼロ）が戻り、失敗するとエラーコードが戻ります。
取得の場合は、成功すると設定されている値が戻り、失敗するとエラーコードが戻ります。

4.11.4 極細い線の描画

PDF Imager-LP はPDF文書を指定された解像度で画像に変換するため、解像度に比較して極端に細い線を期待したとおりに変換できないことがあります。このような場合に、その細い線をより太い線で代替描画することを指定します。
以下の関数で、描画する最小の太さ（線幅）を指定します。これにより、指定された線幅より小さな線幅は最小線幅に代替して描画されます。

[1] フォントなどの文字コードに対するグリフ（字形）が未定義の場合に表示されるグリフ

```
int MlpSetPicture(
    MLP_STROKEPATH_MINIMUM_LINE_WIDTH,
    float      width
);
```

引数

width	最小の線幅をポイント(1 ポイント=1/72 インチ)単位で指定します。 0(ゼロ)または負数を指定すると、最小線幅の指定がなくなります。

ご注意ください。
この関数で変更できる線は、PDFコマンドのパスストロークによって描画される線です。
パスストロークの詳細は、「PDF Reference」を参照してください。

4.12 代替フォント指定

PDF Imager-LP はPDF文書に指定されたフォントが埋め込まれていない場合、そのフォントを別のフォントに替えて画像に変換することができます。

4.12.1 既定フォントの指定

PDF文書に指定されたフォントや代替指定されたフォントが検索できない場合のフォントを指定します。
この指定がない場合は、「MS ゴシック」または「MS 明朝」フォントを使って画像変換されます。

```
int MlpSetDefaultFont(
    TCHAR      *defaultFont
);
```

引数

defaultFont	既定で使用するフォントの名前

4.12.2 代替フォントの指定

PDF文書に指定されたフォントでそれが埋め込まれていない場合は、そのフォントに替えて使用するフォントを指定できます。

```
int MlpSetAlternateFont(
    TCHAR      *originalFont,
    TCHAR      *alternateFont,
    int        always
);

int MlpSetAlternateFont2(
    TCHAR      *originalFont,
    TCHAR      *alternateFont
    TCHAR      *fontWeight
    TCHAR      *fontStyle
    int        always,
    int        replace
);
```

引数

originalFont	PDF文書に指定されたフォントの名前
alternateFont	代替するフォントの名前
fontWeight	代替するフォントの太さ

	“normal”、“bold”または、“100”、“200”、“300”、“400”、“500”、“600”、“700”、“800”、“900”のいずれかを指定します。なお、数値の300以下は“normal”と同じで、400以上は“bold”と同じになります。
fontStyle	“normal”、“italic”または、“oblique”のいずれかを指定します。なお、“italic”と“oblique”は同じです。
always	1(真)を指定すると、originalFont フォントがシステムにインストールされている場合でも alternateFont で代替されます。0(偽)を指定すると、originalFont フォントがシステムにインストールされている場合は代替されません。
replace	1(真)を指定すると、originalFont フォントがPDF文書に埋め込まれている場合でも alternateFont で代替されます。0(偽)を指定すると、originalFont フォントがにPDF文書に埋め込まれている場合は代替されません。

4.13 OCG(レイヤー)

PDF文書は、レイヤーを持つことができます。 PDF Imager-LP は、OCGを初期状態で画像に変換することや各レイヤーの表示・非表示を指定して画像に変換することができます。

4.13.1 OCGの総数取得

PDF文書に記載されたレイヤーの総数を取得します。

```
int MlpGetOcgCount();
```

引数

ありません

戻り値

成功すると、0(ゼロ)以上の正数が戻ります。負数は失敗で、エラーコードです。

4.13.2 OCGの状態取得

OCGのID、名前、階層、表示・非表示の状態を取得します。取得には、MlpOcgData 型(構造体またはクラス)を使います。

```
#define MAX_OCG_NAME_LEN 100
typedef struct MlpOcgDataRec_s {
    int    id;
    TCHAR  name[MAX_OCG_NAME_LEN+1];
    int    level;
    int    visible;
} MlpOcgDataRec_t;
typedef MlpOcgDataRec_t MlpOcgData;
```

メンバー(またはプロパティ)

id	各OCGにつけられたユニークな認識番号
name	PDF文書で示されたname
level	OCGの階層(ゼロがトップレベル)、-1の場合はそのOCGは階層に含まれません。
visible	0:OCGが非表示に設定されている場合、0以外:表示に設定されている場合

```
int MlpGetOcgData(MlpOcgData **data);
```

引数

data	OCGのID、名前、階層、表示・非表示の状態を示す構造体（またはクラス）

戻り値

成功すると、0（ゼロ）以上の正数が戻ります。負数は失敗で、エラーコードです。

4.13.3 OCGの表示・非表示指定

OCGの表示・非表示を指定します。設定には、MlpOcgData 型（構造体またはクラス）を使います。詳細は、「4.12.2 OCGの状態」を参照してください。

```
int MlpetOcgState(
    MlpOcgData    *data
);
```

引数

data	OCGのID、名前、階層、表示・非表示の状態を示す構造体（またはクラス） ただし、id および visible 以外は無視されます。

戻り値

成功すると、0（ゼロ）以上の正数が戻ります。負数は失敗で、エラーコードです。

4.14 コールバック関数

サイズの大きなPDF文書または複雑なPDF文書を画像に変換する場合や、解像度の高い画像を生成する際は、コンピュータの能力によってその変換に時間を要する場合があります。そのような場合にコールバック関数を指定することで、その進捗状況を知ることができます。なお、コールバック関数は、ライブラリと同じスレッドで実行されますので注意してください。

以下の関数で、コールバック関数を登録します。

```
void MlpSetCallbackFunc(
    int         mode,
    int         opt,
    void        *fnc
);
```

引数

mode	進捗を知らせるモードを指定します。指定できるモードは、「4.11.1 ページ解析終了のコールバック」および「4.11.2 画像生成進捗のコールバック」を参照してください。
opt	関数をコールする際のオプション（モードごとに意味が違います）
fnc	コールバック関数（モードごとに形式が違います）

戻り値

成功すると、0（ゼロ）が戻り、失敗するとエラーコードが戻ります。

4.14.1 ページ解析終了（画像変換開始）のコールバック設定

指定されたページを画像に変換する場合はまずそのページを解析し、その後に画像に変換します。

以下のモードとコールバック関数を指定すると、ページの解析後にその関数がコールされます。

mode	MLP_RENDERING_CB
opt	無視されます。整数の0（ゼロ）を指定してください。

fnc　　`void __stdcall rendering_cb(int pageNum)`を指定します。
`pageNum`には、画像に変換されるページ番号が渡ります。

4.14.2 画像生成進捗のコールバック設定

指定されたライン数（ラインは、画像の横幅に相当します）を処理するごとに関数がコールされます。
なお、変換される画像形式ごとに処理が違いますので、関数がコールされるタイミングは必ずしも一定ではありません。

mode　　`MLP_PROGRESS_CB`
opt　　ライン数を指定します。このライン数処理終了後に関数がコールされます。
fnc　　`void __stdcall progress_cb(int lines, int totalLines)`を指定します。
`lines`には処理したライン数が渡り、`totalLines`には生成される画像の全ライン数が渡ります。

4.14.3 パスストローク描画時のコールバック設定

PDF Imager-LP においてPDF文書のパスストロークを描画する場合は、指定された解像度とアンチエイリアスの関係から期待したとおりの画像に変換されない場合があります。このような場合に、PDF Imager-LP が生成した画像に替えてユーザーアプリケーションで作成した画像を貼り付けることができます。
指定された関数は PDF Imager-LP がパスストロークを画像に変換した後で、ページ画像（生成する画像）へ描画する前にコールされます。
なお、パスストロークの描画コマンドを画像に変換する処理は、ユーザーアプリケーションで行います。
パスストロークや描画コマンドなどの詳細は、「PDF Reference」を参照してください。

mode　　`MLP_DRAW_STROKE_PATH_CB`
opt　　無視されます。整数の0（ゼロ）を指定してください。
fnc　　`int __stdcall draw_stroke_path_cb(StrokePathAttr *pa, unsigned char *pixels)`を指定します。
`pa`には、以下のパスストローク属性が渡ります。

```
typedef struct StrokePathAttr_s {
    float   lineWidth;            //パスの線幅
    int     lineCap;              //線の始点・終点形状
    int     lineJoin;             //線接続点の形状
    float   miterLimit;           //とがった線続点の形状
    int     dashLen;              //点線定義配列の長さ
    float   dashPhase;            //最初の点線の長さ
    float   dashData[32];         //点線定義配列
    float   colorfv[5];           //線描画色
    char    *pathString;          //パスストローク コマンド文字列
    int     x, y, width, height;  //変換された画像の位置とサイズ
    int     colors;               //描画色の色数
} StrokePathAttr;
```

戻り値　　コールバック関数からの戻り値によって貼り付けられる画像が変わります。
0（ゼロ）　画像を貼り付けません。
1　　ユーザーアプリケーションによって変更された`pixel`値のパス画像を描画します。
2　　`PDF` Imager-LP が生成した画像を貼り付けます。

その他　　線描画色はカラーの場合 RGBA（Red, Green, Blue, Alpha 値）の順で格納されグレースケール（白黒2値も含みます）の場合はグレースケール値,Alpah 値の順で格納されています。
ピクセルデータは、画像の色数が3の場合は3バイトでひとつの画素を表し RGB（Red, Green, Blue）の順で格納されています。画像の色数が1の場合は、1バイトでひとつの画素を表し、グ

レースケール値です。色数が4色および2色の場合は、格画素値にAlpha値が追加されます。
さらに、画素データは行（横方向スキャン）がライン（縦方向）数隙間無く格納されたデータです。
画像の原点は、左上です。

ご注意：
このコールバック関数を利用すると、生成される画像の解像度によってはピクセルデータのコピーなどに大きな時間を要する場合があります。
以下の関数を使うと、コールバックを抑制したり実行したりできます。

```
int MlpSetPicture(MLP_STROKEPATH_ENABLE_CALLBACK, int flag);
```

flagを1（真）にすると、コールバック関数が実行され、0（偽）にすると実行されません。
コールバックをC#開発環境で利用する場合は、既定でコールバックが抑制されています。

4.14.4 タイル画像定義時のコールバック設定

タイルやタイルを敷き詰めた画像をユーザーアプリケーションで作成したものと替えるために、タイルの画像定義時にユーザー関数をコールバックします。
ただし、コールバックされるのはタイルが画像だけで構成されている場合に限ります。

mode　MLP_DEFINE_IMAGE_TILE_CB
opt　無視されます。整数の0（ゼロ）を指定してください。
fnc　void __stdcall define_image_tile_cb(ImageTileAttr *ta, unsigned char *pixels)を指定します。
pa には、以下のタイル属性が渡ります。

```
typedef struct ImageTileAttr_s {
    int     width, height, colors;   //タイルのサイズ、画像の色数
    float   xstep, ystep;            //タイル敷詰めの縦横ステップ
    Matrix  ctm;
} ImageTileAttr;
```

なお、MatrixはPDF文書に定義されたタイルをユーザー指定空間に写像する行列式の係数です。
戻り値　ありません。

その他　ピクセルデータは、画像の色数が3の場合は3バイトでひとつの画素を表しRGB（Red, Green, Blue）の順格納されています。画像の色数が1の場合は、1バイトでひとつの画素を表し、グレースケール値です。色数が4色および2色の場合は、格画素値にAlpha値が追加されます。
さらに、画素データは行（横方向スキャン）がライン（縦方向）数隙間無く格納されたデータです。
画像の原点は、左上です。

ご注意：
このコールバック関数を利用すると、生成される画像の解像度によってはピクセルデータのコピーなどに大きな時間を要する場合があります。
以下の関数を使うと、コールバックを抑制したり実行したりできます。

```
int MlpSetPicture(MLP_TILE_IMAGE_ENABLE_CALLBACK, int flag);
```

flagを1（真）にすると、コールバック関数が実行され、0（偽）にすると実行されません。
コールバックをC#開発環境で利用する場合は、既定でコールバックが抑制されます。

4.14.5 タイルマスク画像定義時のコールバック設定

タイルやタイルを敷き詰めた画像をユーザーアプリケーションで作成したものと替えるために、タイルのマスク画像定義時にユーザー関数をコールバックします。

ただし、コールバックされるのはタイルが画像だけで構成されている場合に限ります。

mode　MLP_DEFINE_MASK_IMAGE_TILE_CB
opt　無視されます。整数の0(ゼロ)を指定してください。
fnc　void __stdcall define_image_image_tile_cb(ImageTileAttr *ta, unsigned char *pixels)を指定します。
pa には、以下のタイル属性が渡ります。

```
typedef struct ImageTileAttr_s {
    int     width, height, colors; //タイルのサイズ、画像の色数
    float   xstep, ystep;          //タイル敷詰めの縦横ステップ
    Matrix  ctm;
} ImageTileAttr;
```

なお、MatrixはPDF文書に定義されたタイルをユーザー指定空間に写像する行列式の係数です。
戻り値　ありません。

その他　マスクのピクセルデータは、画像の色数が1となります。1バイトでひとつの画素を表し、グレースケール値です。Alpha 値は含まれません。
さらに、画素データは行(横方向スキャン)がライン(縦方向)数隙間無く格納されたデータです。
画像の原点は、左上です。

ご注意:
このコールバック関数を利用すると、生成される画像の解像度によってはピクセルデータのコピーなどに大きな時間を要する場合があります。
以下の関数を使うと、コールバックを抑制したり実行したりできます。

```
int MlpSetPicture(MLP_TILE_MASK_IMG_ENABLE_CALLBACK, int flag);
```

flag を 1(真)にすると、コールバック関数が実行され、0(偽)にすると実行されません。
コールバックを C#開発環境で利用する場合は、既定でコールバックが抑制されます。

4.14.6 タイル処理時のコールバック設定

ユーザーアプリケーションではタイル画像やタイルのマスク画像を元にして、それを敷き詰めた画像を作成できます。PDF Imager-LP は、ユーザーアプリケーションで作成した画像をページ画像(PDF文書を変換した画像)に貼り付けられるようにユーザー関数をコールバックします。
ただし、コールバックされるのはタイルが画像だけで構成されている場合に限ります。

mode　MLP_DRAW_TILED_IMAGE_CB
opt　無視されます。整数の0(ゼロ)を指定してください。
fnc　int __stdcall define_image_image_tile_cb(ImageTileAttr *ta1, unsigned char *pixels1, ImageTileAttr *ta2, unsigned char *pixels2)を指定します。
ta1,ta2 には、以下のタイル属性が渡ります。

```
typedef struct ImageTileAttr_s {
    int     width, height, colors;  //画像のサイズ、画像の色数
    float   xstep, ystep;
    Matrix  ctm;
} ImageTileAttr;
```

なお、ta1,pixel1はタイルの画像で、ta2,pixel2はタイルを敷き詰めた画像です。
また、xstep、ystaep、ctmの値は0(ゼロ)となり意味がありません。
戻り値　コールバック関数からの戻り値によって貼り付けられる画像が変わります。
0(ゼロ)　画像を貼り付けません。

1　　ユーザーアプリケーションによって変更された pixel2 値のタイルを敷き詰画像をページ画像（PDF文書を変換した画像）に描画します。
2　　ユーザーアプリケーションによって変更されたタイル画像 pixel1 を使い、それを敷き詰めた画像をページ画像（PDF文書を変換した画像）に描画します。
3　　PDF Imager-LP が生成した、タイルを敷き詰めた画像をページ画像に貼り付けます。

その他　　ピクセルデータは、画像の色数が3の場合は3バイトでひとつの画素を表し RGB（Red, Green, Blue）の順格納されています。画像の色数が1の場合は、1バイトでひとつの画像をあらわし、グレースケール値です。色数が4色および2色の場合は、格画素値に Alpha 値が追加されます。さらに、画素データは行（横方向スキャン）がライン（縦方向）数隙間無く格納されたデータです。
画像の原点は、左上です。

ご注意：
このコールバック関数を利用すると、生成される画像の解像度によってはピクセルデータのコピーなどに大きな時間を要する場合があります。
以下の関数を使うと、コールバックを抑制したり実行したりできます。

```
int MlpSetPicture(MLP_TILING_ENABLE_CALLBACK, int flag);
```

flag を 1（真）にすると、コールバック関数が実行され、0（偽）にすると実行されません。
コールバックを C#開発環境で利用する場合は、既定でコールバックが抑制されます。

4.15 コールバックの抑制

C/C++開発環境では、コールバック関数を定義することで、パスストロークやタイル処理時にコールバックされるようになります。しかし C#開発環境では、コールバック関数が既定で登録されていてその動作が抑制されています。そのため、C#開発環境でこれらのコールバックを利用する場合は、その動作が抑制されないように指定しなければなりません。

4.15.1 パスストローク描画時コールバックの抑制

パスストローク描画時のコールバックを抑制します。
以下の関数を使いコールバックを抑制したり実行したりします。

```
int MlpSetPicture(
    MLP_STROKEPATH_ENABLE_CALLBACK,
    int        flag
);
```

引数

flag　　1（真）を指定するとコールバック関数が実行され、0（偽）にするとコールバックは抑制されます。

戻り値

成功すると、0（ゼロ）が戻り、失敗するとエラーコードが戻ります。

4.15.2 タイル画像定義時コールバックの抑制

タイル画像が定義された際のコールバックを抑制します。
以下の関数を使いコールバックを抑制したり実行したりします。

```
int MlpSetPicture(
    MLP_TILE_IMAGE_ENABLE_CALLBACK,
    int        flag
```

```
);
```

引数

flag	1（真）を指定するとコールバック関数が実行され、0（偽）にするとコールバックは抑制されます。

戻り値

成功すると、0（ゼロ）が戻り、失敗するとエラーコードが戻ります。

4.15.3 タイルマスク画像定義時コールバックの抑制

タイルマスク画像が定義された際のコールバックを抑制します。
以下の関数を使いコールバックを抑制したり実行したりします。

```
int MlpSetPicture(
    MLP_TILE_MASK_IMG_ENABLE_CALLBACK,
    int         flag
);
```

引数

flag	1（真）を指定するとコールバック関数が実行され、0（偽）にするとコールバックは抑制されます。

戻り値

成功すると、0（ゼロ）が戻り、失敗するとエラーコードが戻ります。

4.15.4 タイル処理時コールバックの抑制

タイル処理時のコールバックを抑制します。
以下の関数を使いコールバックを抑制したり実行したりします。

```
int MlpSetPicture(
    MLP_TILING_ENABLE_CALLBACK,
    int         flag
);
```

引数

flag	1（真）を指定するとコールバック関数が実行され、0（偽）にするとコールバックは抑制されます。

戻り値

成功すると、0（ゼロ）が戻り、失敗するとエラーコードが戻ります。

4.16 画像への変換処理用のメモリーサイズ

PDF文書の画像への変換では、その画像を格納するメモリー領域をあらかじめ確保します。そのため、画像の解像度を高くした場合や、大きなサイズのPDF文書の場合は、多くのメモリーを必要とします。

この確保されるメモリーのサイズを以下によって変更できます。PDF Image-LP では、指定されたメモリーサイズを超えない範囲で画像用のメモリーを確保します。既定の画像用メモリーサイズは200MB です。

```
int MlpSetPictureBufSize(
    int         mb
);
```

引数

mb	MB 単位でメモリーサイズを MIN_PICT_BUF_SIZE 以上かつ

MAX_PICT_BUF_SIZE 以下で指定します。
範囲外の値を指定すると、MIN_PICT_BUF_SIZE または MAX_PICT_BUF_SIZE に正規化されます。

戻り値

成功すると、0（ゼロ）が戻り、失敗するとエラーコードが戻ります。

4.17 変換された画像をパネルに描画

PDF Imager-LP は、その内部に特別な画像を生成できます。この画像を PDF Imager-LP では「パネル」と呼びます。

パネルにはPDF文書を変換したページ画像を1つ以上貼り付けることができます。さらに、ページ画像は別のPDF文書を変換した画像でも貼り付けられます。作成されたパネルは指定された画像形式で取得できます。

一般に以下の手順で複数のページ画像をパネルに貼り付けます。

1. ライブラリを初期化（MlpInitialize）
2. パネルを生成（MlpCreatePanel）
3. PDF文書を開く（MlpOpenDoc）
4. 変換する画像の色空間や解像度などを指定
   （MlpSetPictureRGB、MlpSetPictureResolutionなど）
5. PDF文書のページを指定して内部の画像データに変換（MlpCreatePictMem）
6. 3、4、5の手順を必要に応じて繰り返します。
7. パネルのデータを画像形式で取得（MlpPanelToFile）
8. PDF文書を閉じる（MlpCloseDoc）
9. ライブラリを終了（MlpUninitialize）

4.17.1 パネル作成

複数のページ画像を貼り付けられるパネルを生成します。生成されたパネルは不透明白色でクリアされています。

パネルはライブラリ内部に1つだけ作成されます。既に作成されている場合は、そのパネルを削除して新たなパネルが作成されます。作成されたパネルは、MlpFreePictMem で削除できますが

```
int MlpCreatePanel(
    int         width,
    int         height,
    int         colorspace
);
```

引数

width	パネルの横方向のピクセル数
height	パネルの縦方向のピクセル数
colorspace	色空間（MLP_PICTURE_RGB または MLP_PICTURE_GRAY）

戻り値

成功すると、0（ゼロ）が戻り、失敗するとエラーコードが戻ります。

4.17.2 パネルをクリア

複数のページ画像を貼り付けられるパネルを不透明白色でクリアします。

```
int MlpClearPanel();
```

引数

ありません

戻り値

成功すると、0（ゼロ）が戻り、失敗するとエラーコードが戻ります。

4.17.3 パネルにページ画像を貼り付ける

メモリー上のページ画像をパネルに貼り付けます。

メモリーにはあらかじめ MlpCreatePictMemを使ってページ画像を作成しておかなければなりません（メモリー上に画像がない場合などは貼り付けができません）。

メモリー上の画像はパネルの色空間と同じ色空間に自動で変換されます。ページ画像をパネルに貼り付ける際に画像のサイズや解像度は変更されません。ページ画像にアルファチャネルはありませんので、後から描画したページ画像でそれまでのページ画像に上書きされます。ページ画像の背景は不透明白色ですのでご注意ください。

パネル画像やページ画像の原点（x=0、y=0）は、左上です。

```
int MlpMemPictToPanel(
    int         xPos,
    int         yPos,
    float       alpha
);
```

引数

xPos	ページ画像を貼り付けるパネルの X 座標
yPos	ページ画像を貼り付けるパネルの Y 座標
alpha	貼り付けるページ画像の不透明度（0.0 から 1.0 の間で指定）

戻り値

成功すると、0（ゼロ）が戻り、失敗するとエラーコードが戻ります。

4.17.4 パネルにページ画像の境界を指定して貼り付ける

メモリー上のページ画像の境界を指定してパネルに貼り付けます。境界を指定すると、画像を切り取ることなどができます。

指定された領域の内部がパネルに貼り付けられます。指定された領域にページ画像がない部分はパネルに描画されません。

メモリーにはあらかじめ MlpCreatePictMemを使ってページ画像を作成しておかなければなりません（メモリー上に画像がない場合などは貼り付けができません）。

メモリー上の画像はパネルの色空間と同じ色空間に自動で変換されます。ページ画像をパネルに貼り付ける際に画像のサイズや解像度は変更されません。ページ画像にアルファチャネルはありませんので、後から描画したページ画像でそれまでのページ画像に上書きされます。ページ画像の背景は不透明白色ですのでご注意ください。

パネル画像やページ画像の原点（x=0、y=0）は、左上です。

```
int MlpMemPictToPanelBBox(
    int         xPos,
    int         yPos,
    float       alpha,
    int         bboxX,
    int         bboxY,
    int         bboxWidth,
    int         bboxHeight
);
```

引数

xPos	ページ画像を貼り付けるパネルの X 座標
yPos	ページ画像を貼り付けるパネルの Y 座標

alpha	貼り付けるページ画像の不透明度（0.0 から 1.0 の間で指定）
bboxX	貼り付けるページ画像から切り取る領域の X 座標（ページ画像上の座標）
bboxY	貼り付けるページ画像から切り取る領域の Y 座標（ページ画像上の座標）
bboxWidth	貼り付けるページ画像から切り取る領域の幅（ピクセル値）
bboxHeight	貼り付けるページ画像から切り取る領域の高さ（ピクセル値）

戻り値

成功すると、0（ゼロ）が戻り、失敗するとエラーコードが戻ります。

4.17.5 パネルに矩形を描画

パネルに矩形を描画します。
ページ画像を切り取った場合など、その領域を明示したい場合などに利用します。
パネル画像の原点（x=0、y=0）は、左上です。矩形は不透明な指定色で描画されます。

```
int MlpPaintRectToPanel(
    int             xPos,
    int             yPos,
    int             width,
    int             height,
    int             lineWidth,
    unsigned char   color[]
);
```

引数

xPos	矩形を描画するパネルの X 座標
yPos	矩形を描画するパネルの Y 座標
width	描画する矩形の幅（ピクセル値）
height	描画する矩形の高さ（ピクセル値）
lineWidth	矩形の線幅（ピクセル値）
color	矩形線の色配列（カラーの場合はRGBの3色、グレースケールの場合は1色） 各色は0～255の値で指定

戻り値

成功すると、0（ゼロ）が戻り、失敗するとエラーコードが戻ります。

4.17.6 パネルのデータを画像で取得

パネルのデータを画像に変換しファイルで取得します。

```
int MlpPanelToFile(
    TCHAR     *fileName
);
```

引数

fileName	画像ファイル名 画像形式はファイルの拡張子によって以下のように自動で選択されます。 拡張子が“png”の場合、PNG形式の画像 拡張子が“jpeg”または“jpg”の場合、JPEG形式の画像 拡張子が“tiff”または“tif”の場合、TIFF形式の画像

戻り値

成功すると、0（ゼロ）が戻り、失敗するとエラーコードが戻ります。

4.17.7 パネル削除

作成されたパネルを削除し、そのメモリー領域を開放します。

パネル領域は、ライブラリの使用終了で自動的に削除されるためメモリー領域に問題がない場合は開放しなくともかまいません。

```
int MlpFreePanel();
```

引数

ありません

戻り値

成功すると、0（ゼロ）が戻り、失敗するとエラーコードが戻ります。

4.18 画像変換の並列処理

PDF Imager-LP は、複数ページを画像に変換する場合に、それぞれをスレッドで並列に処理します。また、各ページを画像に変換する場合でも、そのページをより小さな領域に分割して並列処理します。

PDF Imager-LP のこのような並列処理は既定の動作です。ただし、この動作は抑制することやスレッドの最大数を変更できます。

4.18.1 複数ページの並列処理の抑制

複数ページを画像変換する場合にスレッドによる並列処理を抑制します。

```
int MlpSetPicture(
    MLP_THREAD_EACH_PAGE,
    int         flag
);
```

引数

flag	0（ゼロ）を指定すると、並列処理は抑制されます。それ以外の場合は抑制されません。

戻り値

成功すると、0（ゼロ）が戻り、失敗するとエラーコードが戻ります。

4.18.2 複数ページの並列処理の最大スレッド数

複数ページを画像変換する場合の最大スレッド数（並列処理数）を変更します。

```
int MlpSetPicture(
    MLP_THREAD_EACH_PAGE_COUNT,
    int         num
);
```

引数

num	実行されるスレッドの総数を指定します。省略した場合は、使用しているコンピュータの論理 CPU 数と同じ数が使用されます。

戻り値

成功すると、0（ゼロ）が戻り、失敗するとエラーコードが戻ります。

4.18.3 単一ページの並列処理の抑制

各ページを画像変換する場合の並列処理を抑制します。

```
int MlpSetPicture(
    MLP_THREAD_SPLIT_PAGE,
    int         flag
);
```

引数

flag	0（ゼロ）を指定すると、並列処理は抑制されます。それ以外の場合は抑制されません。

戻り値

成功すると、0（ゼロ）が戻り、失敗するとエラーコードが戻ります。

4.18.4 単一ページの並列処理の最大スレッド数

各ページを画像変換する場合の並列処理の最大スレッド数（並列処理数）を変更します。

```
int MlpSetPicture(
    MLP_THREAD_SPLIT_PAGE_COUNT,
    int         num
);
```

引数

num	実行されるスレッドの総数を指定します。省略した場合は、使用しているコンピュータの論理 CPU 数と同じ数が使用されます。

戻り値

成功すると、0（ゼロ）が戻り、失敗するとエラーコードが戻ります。

4.19 画像変換の結果

PDF Imager-LP はPDF文書にフォントが埋め込まれていない場合に代替フォントを使って文字を画像に変換します。この際に、PDF文書で指定された文字コードのグリフ（字形）が代替されたフォントに存在しない場合があります。この場合は、一般にグリフが存在しない場合の特殊な字形（未定義グリフ）が表示されます。

このように文字が「未定義グリフ」に変換されたことを、画像生成後に知ることができます。

以下の関数を利用して、未定義グリフに変換された文字数を取得できます。

```
int MlpGetDocumentInfo(
    MLP_GLYPH_NOT_FOUND_COUNT,
    int         *count
);
```

引数

count	未定義グリフに変換された文字の総数。

ご注意ください。

この関数で取得できる総数は、「正しい文字（グリフ）に変換されなかった文字の総数」ではありません。画像への変換時に指定されたフォントに、対照のPDF文書で指定されたグリフ（字形）が存在しておらず、未定義グリフに変換された文字の総数です。

4.20 PDF文書処理の終了

PDF文書の画像変換処理を終了します。

```
void MlpCloseDoc();
```

引数
　　ありません。
戻り値
　　ありません。

4.21 ライブラリの使用終了

ライブラリの使用を終了します。

```
void MlpUninitialize();
```

引数
　　ありません。
戻り値
　　ありません。

5.0 初期化ファイル（初期化データ）について

PDF Imager-LP は初期化ファイル（または初期化データ）を使うことでPDF文書に指定されたフォント（埋め込み・非埋め込みフォントにかかわらず）を指定したフォントに置き換えて変換することができます。

初期化ファイル（または初期化データ）はXML形式です。

初期化ファイル（または初期化データ）の指定手順は、「4.3 初期化ファイル（初期化データ）を指定する」を参照してください。

5.1 非埋め込みフォントの代替指定

PDF文書で使用しているフォントがその文書に埋め込まれていない場合は、システムにインストールされている同じ名前のフォントを検索して利用します。もし、必要なフォントがシステムで検索できない場合は、「MS ゴシック」または「MS 明朝」フォントで代替します。

属性値 `always=true` を指定すると、PDF文書で指定されたフォントがシステムにインストールされている場合でも代替指定されたフォントが使用されます。なお、`always` の既定値は `false`（偽）です。省略すると、システムにインストールされたフォントが使用されます。

代替フォントは関数で指定することもできます。「4.12 代替フォント指定」を参照してください。

この指定ファイルまたはデータの指定は、「4.3 初期ファイル（初期化データ）を指定する」を参照してください。

フォントの指定は、以下のようにXML形式で指定します。

```
<?xml version="1.0" encoding="shift_jis" ?>
<init>
  <fonts default="Font0">
    <font org="Font11" alt="Font12" />
    <font org="Font21" alt="Font22" />
    <font always="true" org="Font31" alt="Font32" />
  </fonts>
</init>
```

属性

`org`	PDF文書に指定されたフォント名指定
`alt`	代替するフォント名指定
`always`	値に `true`（真）を指定すると、`org` で指定されたフォントは `alt` で指定されたフ

ォントに代替されます。
false（偽）を指定すると、org で指定されたフォントがシステムにインストールされていない場合にのみ alt で指定されたフォントに代替されます。
既定値は、false

default　指定のフォントがシステムにインストールされていない場合に既定で利用されるフォント名
既定値は「MS ゴシック」および「MS 明朝」

値

Font0　PDF文書で指定されたフォントや代替フォントが検索できない場合に利用されるフォント名

Font11、Font21　PDF文書で指定されたフォントのPostScript名
属性「org」で指定します。

Font12、Font22　代替するシステムにインストールされたフォント名
属性「alt」で指定します。

Font31 は Font32 に常に代替されます。ただし、Font32 がシステムにインストールされている場合に限ります。

代替フォントは<init>および<fonts>の子要素として指定します。

この初期化ファイルはShift_JIS文字コードで記述してください。それ以外の文字コードは利用できません。

5.2 埋め込みフォントの代替指定

PDF Imager-LP はPDF文書にフォントが埋め込まれている場合でもフォントを代替指定できます。
代替フォントは関数で指定することもできます。「4.12 代替フォント指定」を参照してください。

この指定ファイルまたはデータの指定は、「4.3 初期ファイル（初期化データ）を指定する」を参照してください。

フォントの指定は、以下のようにXML形式で指定します。

```
<?xml version="1.0" encoding="shift_jis" ?>
<init>
  <fonts default="Font0">
    <font replace="true" org="Font11" alt="Font12" />
    <font replace="true" org="Font21" alt="Font22" />
    <font always="true" replace="true" org="Font31" alt="Font32" />
  </fonts>
</init>
```

属性

org　PDF文書に指定されたフォント名指定

alt　代替するフォント名指定

replace　値に true（真）を指定すると、org で指定されたPDF文書の埋め込みフォントが利用されなくなります。

always　値に false（偽）を指定すると、org で指定されたフォントがシステムにインストールされているフォントに代替使用されます。システムにインストールされていない場合は alt に指定されたフォントが使用されます。
true（真）を指定すると、org で指定されたフォントに替わって alt で指定されたフォントが使用されます。
既定値は、false です。

replace　値に true（真）を指定すると、org フォントがPDF文書に埋め込まれている場合でも always が真であれば alt フォントに代替され、always が偽であれば

org フォントがシステムにインストールされていない場合に alt フォントに代替されます。
false（偽）を指定すると、org で指定されたフォントがPDF文書に埋め込まれている場合代替されません。
既定値は、false

default　指定のフォントがシステムにインストールされていない場合に既定で利用されるフォント名です。既定値は「MS ゴシック」および「MS 明朝」です。

値

Font0　PDF文書で指定されたフォントや代替フォントが検索できない場合に利用されるフォント名

Font11、Font21　PDF文書で指定されたフォントのPostScript名
属性「org」で指定します。

Font12、Font22　代替するシステムにインストールされたフォント名
属性「alt」で指定します。

Font31 がPDF文書に埋め込まれている場合でも Font32 に代替されます。なお、always を false（偽）とすると Font31 がシステムにインストールされている場合にはインストールされているフォントが使用されます。

代替フォントは<init>および<fonts>の子要素として指定します。

この初期化ファイルはShift_JIS文字コードで記述してください。それ以外の文字コードは利用できません。

5.3 代替フォントの太さとスタイル

初期化ファイルまたは初期化データでフォントを代替する場合は、指定するフォントの太さ（Font-Weight）やスタイル（Font-Style）を指定できます。
代替フォントは関数で指定することもできます。「4.12 代替フォント指定」を参照してください。

この指定ファイルまたはデータの指定は、「4.3 初期ファイル（初期化データ）を指定する」を参照してください。

フォントの指定は、以下のようにXML形式で指定します。

```
<?xml version="1.0" encoding="shift_jis" ?>
<init>
  <fonts>
    <font org="Font11" alt="Font12" weight="bold" />
    <font org="Font21" alt="Font22" weight="normal" style="italic" />
  </fonts>
</init>
```

属性

weight　代替フォントの太さ指定
値は、normal、bold または 100、200、300、400、500、600、700、800、900 のいずれかを指定できます。
なお、300以下の数値を指定すると normal、400以上の数値を指定すると bold と同じ太さになります。
既定値は、normal です。この属性は省略できます。

style　代替フォントのスタイル指定
値は、normal、italic、oblique のいずれかを指定できます。
なお、italic と oblique は同じ斜体です。
こ既定値は、normal です。の属性は省略できます。

この初期化ファイルはShift_JIS文字コードで記述してください。それ以外の文字コードは利用できません。

5.4 複数行フォームの文字サイズ

PDF Imager-LP は初期化指定されたフォームの文字列も画像に変換します。

なお、一般にPDF文書では複数行が指定されたフォームの文字サイズが指定されない場合があります。その場合のために、そのフォントサイズを初期ファイルまたはデータであらかじめ指定します。

この指定ファイルまたはデータの指定は。「4.3 初期ファイル（フォントの代替）」を参照してください。

フォントサイズの指定は、以下のように XML 形式で指定します。

```
<?xml version="1.0" encoding="shift_jis" ?>
<init>
  <annot>
    <widget>
      <text>
        <multiline use-last-font-size="true" font-size="5" />
    </widget>
  </annot>
</init>
```

属性値 `use-last-font-size` 直前のフォームで使用したフォントサイズを優先して使います。
属性値 `font-size` 複数行フォームへの表示でのフォントサイズ指定。
`use-last-font-size` の指定が無い場合は常に指定されたサイズで表示されます。
複数行フォームのフォントサイズは上記の階層どおりに指定します。

この初期化ファイルはShift_JIS文字コードで記述してください。それ以外の文字コードは利用できません。

6.0 文字列でのページ指定方法

複数のページで構成されたTIFF画像に変換する場合、ページを文字列で指定できます（「4.8 複数ページのTIFF画像に変換（その2）」参照）。ページの指定方法は、以下のとおりです。

複数のページを指定する場合の区切り文字は、スペース、タブまたはコンマのいずれかです。
PDF文書に存在しないページ番号は、無視されます。
最初のページをページ番号「0」と指定できます。
最後のページをページ番号「-1」と指定できます。
連続したページをマイナス文字（"-"）で区切って初めのページ番号、終わりのページ番号の順に昇順で指定します。降順では指定できません。

指定の例：

`"0,3,5,7"`	ページ番号 `1,3,5,7` をこの順で画像に変換します。`0` は、ページ番号「1」に変換されます。
`"2,5,3,9,-1"`	ページ番号 `2,5,3,9` 及び最終のページをこの順で画像に変換します。5ページで構成された画像が作成されます。
`"2,2,3,3"`	ページ番号 `2` を2回そしてページ番号 `3` を2回この順で画像に変換します。4ページで構成された画像が作成されます。
`"1,3-7,9"`	ページ番号 `1,3,4,5,6,7,9` をこの順で画像に変換します。
`"7-4,10"`	ページ番号 `7,6,5,4,10` をこの順で画像に変換します。
`"1-3,6-5"`	ページ番号 `1,2,3,6,5` をこの順で画像に変換します。

上記の例を組み合わせてページを指定できます。
いずれの場合でも、PDF文書に存在しないページ番号は無視されます（単独で現れる「`0`」と「`-1`」を除きます）。

7.0 定数 一覧

定数の一覧を以下に記します。

定数	値	内容
MIN_RESOLUTION	50	最小の解像度
MAX_RESOLUTION	10000	最大の解像度
DEFAULT_RESOLUTION	150	既定の解像度
MIN_QUALITY	10	JPEG圧縮が指定された際の品質の最小値
MAX_QUALITY	90	JPEG圧縮が指定された際の品質の最大値
MIN_PICT_BUF_SIZE	50	画像変換用メモリーサイズの最小値
MAX_PICT_BUF_SIZE	1024	画像変換用メモリーサイズの最大値

8.0 エラーコード 一覧

エラーコードを以下に記します。

MLP_ALREADY_INITIALIZED	既に初期化されています。 MlpUninitialize()で終了する、もしくはそのまま処理を続けます。
MLP_NOT_INITIALIZED	初期化できない、もしくは、初期化していない場合。 MlpInitialize()で再度初期化してください。
MLP_INIT_FILE_OPEN_ERROR	初期化ファイルを読めません。 初期化ファイルのパスや名前を正しく指定してください。
MLP_LICENSE_ERROR	不正なライセンスキーもしくは、評価用ライセンスキーの期限切れです。 正規のライセンスキーもしくは、新しいライセンスキーを使用します。
MLP_ALREADY_OPENED	既にPDF文書をオープンしています。 MlpCloseDoc()を使ってPDF文書をクローズしてから再度オープンします。
MLP_FILE_OPEN_ERROR	指定のPDF文書をオープンできません。 PDFファイルのパスや名前を正しく指定してください
MLP_FILE_IS_NOT_PDF	PDF文書として解析できません。 正しいPDF文書を指定してください。
MLP_FILE_NOT_DECRYPTED	PDF文書が暗号化されていますが、指定のパスワードでは復号できません。 暗号化の際に利用したパスワードを指定してください。
MLP_FILE_NOT_OPENED	PDF文書がオープンされていません。 MlpOpenDoc()を使ってPDF文書をオープンしてください。
MLP_PDF_PARSE_ERROR	PDF文書の解析中にエラーとなりました。 指定のPDF文書は画像に変換できません。
MLP_PDF_HAS_NOT_PAGE	指定のPDF文書にはページがありません。 ページのあるPDF文書を指定してください。
MLP_INVALID_PAGE_NUMBER	指定したページの番号は無効です。 ページ番号は、1以上でPDF文書のページ総数を超えてはいけません。
MLP_INVALID_RESOLUTION	指定された解像度は無効です。 解像度は、50以上2000以下を指定してください。
MLP_INVALID_QUALITY	指定されたJPEG品質は無効です。 JPEG品質は、10以上100以下を指定してください。
MLP_NO_OUTPUT_FILE	出力ファイルが指定されていません。または、指定の出力ファイルの形式（拡張子）が無効です。 正しい形式の出力ファイル名を指定してください。

MLP_TOO_LARGE_PIXEL	作成しようとしている画像が大きすぎます。 解像度下げる、もしくはPDF文書のページサイズを小さくしてください。
MLP_DRAW_ERROR	画像作成用のメモリー領域を確保できません。または、画像の書き出しに失敗しました。
MLP_MEMORY_ERROR	メモリー領域の確保に失敗しました。 致命的なエラーです。
MLP_INVALID_ARG_VALUE	関数の引数が不正です。 正しい値を指定してください。
MLP_INVALID_PICT_TYPE	変換される画像の形式が不正です。 利用できる画像形式を指定してください。
MLP_INVALID_CMD	指定されたコマンドが不正です。 正しいコマンドを指定してください。
MLP_NO_DATA	データがありません。
MLP_FAIL_TO_GET_PAGE	ページの取得に失敗しました
MLP_INVALID_DATA	無効なデータ
MLP_NO_LICENSE_KEY	ライセンスキーが指定されていません
MLP_UNUSABLE_LICENSE	このバージョンでは使えないライセンスです
MLP_EXPIRED_LICENSE	失効したライセンスです
MLP_ILLIGUL_OS_LICENSE	このOSでは使えないライセンスです
MLP_FONT_NOT_FOUND	フォントが見つかりません
MLP_ALT_FONT_NOT_FOUND	代替フォントが見つかりません
MLP_FONT_FILE_NOT_OPENED	フォントファイルを開けません
MLP_FONT_NOT_LOADED	フォントをロードできません
MLP_GLYPH_NOT_FOUND	グリフを検索できません