PDF Stamper

Version 1.10.1

株式会社トラスト・ソフトウェア・システム
2024 年 5 月

目次

1.0 はじめに 1
2.0 利用環境および PDF のバージョン 1
3.0 インストール 1
3.1 ファイルの概要 1
3.2 C/C++ インターフェース 2
3.3 .NET インターフェース 2
3.4 評価用ライセンスについて 2
4.0 スタンプ付加概要 2
5.0 関数 – C/C++、.NET の開発環境 2
5.1 初期化 2
5.2 ライセンスの追加 3
5.3 ライセンス情報の表示および取得 3
5.4 初期化ファイル（または初期化データ）を指定する 3
5.5 既存のＰＤＦファイルをオープン 4
5.6 新規にＰＤＦ文書を作成してオープン 5
5.7 ＰＤＦ文書の情報 5
5.7.1 PDF文書のページ数取得 5
5.7.2 PDF文書の許可（パーミッション）フラグ 5
5.7.3 PDF文書の暗号化レビジョン 6
5.8 指定の単一ページを画像形式ファイルに変換 6
5.9 出力画像の色空間を指定する 7
5.9.1 RGB カラー画像を指定 7
5.9.2 CMYK カラー画像を指定 7
5.9.3 グレースケール画像を指定 7
5.9.4 白黒ディザ画像を指定 7
5.9.5 白黒2階調画像を指定 8
5.10 出力画像の透明背景指定 8
5.11 カラープロファイル 8
5.12 画像のサイズ 9
5.12.1 ページサイズの種類 9
5.12.2 画像のピクセルサイズを取得 10
5.12.3 ページの境界ボックスの位置を取得 10
5.12.4 ページの回転角を取得 11
5.13 スタンプ設定 11
5.13.1 スタンプ設定データ 11

5.13.2 スタンプ設定ファイル 12
5.13.3 スタンプ設定を削除 12
5.14 スタンプ付加指定 12
5.14.1 全てのスタンプを付加 12
5.14.2 データで指定されたスタンプを付加 13
5.14.3 ファイルで指定されたスタンプを付加 13
5.14.4 初期設定データで指定された名前付きスタンプを付加 13
5.16 現在のＰＤＦデータをＰＤＦファイルや画像形式ファイルに出力 14
5.16.1 PDFファイルに出力 14
5.16.2 画像形式データとしてファイルに出力 14
5.16.3 SVG画像形式データとしてファイルに出力 14
5.17 現在のＰＤＦ文書を画像データに変換して取得 15
5.17.1 指定されたページを画像に変換 15
5.17.2 変換された画像データを取得 15
5.17.3 変換された画像データを取得（.NET 開発環境専用） 16
5.18 フォントフォルダーの指定 16
5.18.1 フォントファイルが格納されたフォルダーを指定 16
5.18.2 フォントファイルのフォント名リスト作成 17
5.18.3 現在利用できるフォント名のリスト取得 17
5.19 文書のクローズ（ＰＤＦ文書処理の終了） 17
5.20 ライブラリの終了 17
6.0 **スタンプ設定データ** 18
6.1 初期化データで指定 18
6.2 スタンプ設定データで指定 18
7.0 **スタンプ設定データ詳細** 19
7.1 スタンプ指定ルート要素 19
7.2 スタンプ 19
7.3 文字列 21
7.3 画像 23
7.4 矩形 24
7.5 円形 25
7.6 枠線 26
7.7 複数行の文字列を描画する 29
8.0 **著作権** 30

1.0 はじめに

PDF Stamper はPDF（Portable Document Format）文書にスタンプ付加機能をアプリケーションに追加するライブラリです。

通常スタンプは注釈（Annotation）として追加されますが、ページコンテンツに追加することもできます。スタンプとして追加できるのは文字列、画像および図形（矩形または円形のパス）です。

PDF Stamper によって生成されたPDF文書は更新された（Updated PDF）文書ではなく新規に生成されます。Web用に最適化されたPDF文書（Linearized PDF）から生成されるPDF文書も新規に生成された通常の（Linearized PDFではない）文書となります。

2.0 利用環境および PDF のバージョン

PDF Stamper は以下の環境で利用できます。

利用環境	Windows 8、8.1、10
開発環境	C/C++、C#

PDFのバージョン PDF 1.4 から PDF 1.7 およびPDF 2.0（その一部）にスタンプを付加できます。

PDF Stamper はパスワードで暗号化されたPDF文書にもスタンプを付加できますが、スタンプが付加されたPDF文書を暗号化PDF文書として出力しません。そのため、PDF文書に解像度や印刷の可能・不可能などの制限項目は削除されます。なお、暗号化されたPDF文書にスタンプを付加する場合はパスワードが必要です。

PDF文書に付加できるテキスト文字列のフォントは「True Type 形式」または「Compact Font 形式」のOpenTypeフォントファイルで、画像の形式は、PNG形式、JPEG形式、TIFF形式、BMP形式です。

3.0 インストール

PDF Stamper 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 Stamper 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#（Visual Studio 2017プロジェクト）サンプル コードなど

開発環境に応じて適切なフォルダーにコピーして使用してください。
PDF Stamper を使用するためにはライセンスキーが必要です。

3.1 ファイルの概要

PDF Stamper に含まれるファイルの概要です。

lib/x86/PdfStructure.dll	PDF文書を解析・変更するためのライブラリ（32ビット環境用）
doc/PdfStamper 説明書.pdf	「PDF Stamper 説明」（本書）
doc/使用許諾契約書 第7版.pdf	「PDF Stamper 使用許諾契約書」
lib/x64/PdfStructure.dll	PDF文書を解析・変更するためのライブラリ（64ビット環境用）
lib/x86/PdfStructure.lib	C/C++開発環境用のリンクライブラリ（32ビット環境用）
lib/x64/PdfStructure.lib	C/C++開発環境用のリンクライブラリ（64ビット環境用）
lib/PdfStamperNET.dll	PDFスタンプ付加機能を対象とした C#用ラッパーDLL
lib/PdfStructureNET.dll	PdfStructure.dll の全機能用ラッパーDLL 通常は使用しません。
include/PdfStamper.h	C/C++用でPDFスタンプ付加機能を対象としたヘッダファイル

include/PdfStructure.h	C/C++用で PdfStructure.dll の全機能を対象としたヘッダファイル 通常は使用しません。
sample/C	C 言語のサンプルコード
sample/C#	C#言語のサンプルコード
sample/init.xml	PDF Stamper の初期設定を変更する初期化ファイルの例
sample/電子印鑑.xml	PDF文書に追加する電子印鑑を定義するXML文書の例
sample/スタンプ.xml	PDF文書に追加する画像や文字列を定義するXML文書の例

ご注意ください:
ネイティブ DLL（PdfStructure.dll）は32ビット環境用および64ビット環境用に最適化されています。必ず開発・利用環境に合ったDLLを使用してください。また、開発時のプラットフォームターゲットは利用環境に必ず合致させてください。

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（PDFStamper.h）を使い、ライブラリ（PdfStructure.lib）をリンクしてください。実行時には PdfStructure.dll を参照できるようにしてください。

3.3 .NET インターフェース

PDFスタンプ追加ライブラリ（PdfStructure.dll）は. NETアセンブリではありません。C#からスタンプ付加機能を使うための. NETアセンブリDLL（PdfStamperNET.dll）を参照して開発します。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。このDLLが参照する PdfStructure.dll はプラットフォーム（x64またはWin32）に最適化されていますので、利用環境に合った DLL を使ってください。

開発時にはターゲットプラットフォームを「x68」もしくは「x64」に指定します。

開発時には以下の名前空間を追加します。

```
using PDFTools.PDFStructure;
```

以下の手順でインスタンスを生成してください。

```
Stamper stamp = new Stamper();
```

3.4 評価用ライセンスについて

評価用ライセンス（ライセンスキー）を使って生成されたPDF文書や画像には透かしが挿入される場合があります。 正規ライセンス（ライセンスキー）では透かしは表示されません。

4.0 スタンプ付加概要

5.0 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/C++開発環境で利用する場合は接頭辞「Mlp」を付加した関数名に読み替えてください。

5.1 初期化

PDF Stamper 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 Uninitialize メソッドを使って開放しなければなりません。

```
int Initialize(
    string      licenseKey
);
```

引数

licenseKey　　PDF Stamper 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でその値はエラーコードです。

5.2 ライセンスの追加

PDF Stamper は PdfStructure.dll ライブラリの一部の機能を使ってスタンプをPDF文書に付加します。PdfStructure.dll ライブラリは他に指定ページの画像化や、PDF文書からその構成要素を抽出することができます。これらの機能を有効にするためには適切なライセンス（ライセンスキー文字列）が必要です。

適切なライセンスキーを以下のメソッドで入力すると、そのライセンスされた機能を使えるようになります。ライセンスされた機能を使う場合は、PdfStructureNET.dll（C#または VB.NET 開発環境の場合）や PdfStructure.h（C/C++開発環境の場合）を使います。

```
int AddLicense(
    string      licenseKey
);
```

引数

licenseKey　　他の機能を使用するためのライセンス文字列

戻り値

ライセンス追加に成功すると0（ゼロ）が戻ります。それ以外の場合はエラーでその値はエラーコードです。

5.3 ライセンス情報の表示および取得

PDF Stamper 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。
ShowLicenseInfo は結果をコンソールに出力し、LicenseInfoStr は結果を引数に格納します。

```
void ShowLicenseInfo();

string GetLicenseInfoStr();
```

引数

ありません。

戻り値

ShowLicenseInfo は戻り値がありません。
LicenseInfoStr は成功するとライセンス情報文字列が戻ります。失敗すると null 文字が戻ります。

5.4 初期化ファイル（または初期化データ）を指定する

PDF Stamper 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「6.0 スタンプ設定データ」を参照してください。

```
int LoadInitFile(
    string      initFileName
);

int LoadInitData(       //.NET 開発環境で使用可能
    string      data
);

int LoadInitData(
    string      data
    int         length
);
```

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）
length	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が戻ります。

ご注意

PDF Stamper はスタンプ付加の前にPDF文書を解析します。この解析データなどの情報はキャッシュされます。このキャッシュよって初期化の値を変更するタイミングによっては期待する結果を得られない場合があります。

初期化のデータを替えた場合は文書のクローズやライブラリの終了処理が必要な場合があります。

5.5 既存のPDFファイルをオープン

スタンプを付加するPDFファイル（またはデータ）をオープンします。

PDF Stamper はパスワードで暗号化されたPDFファイル（またはデータ）をオープンしスタンプを付加できます。しかし、PDF Stamper はスタンプが追加されたPDF文書を暗号化しませんので元のPDF文書の制限項目は削除されます。

引数に指定されたパスワードは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し、暗号化されていない場合は無視されます。

```
int OpenDoc(
    string      fileName,
    string      password,
);

int OpenMemDoc(
    byte[]      pdfData,
    uint        length
    string      password,
);
```

引数

fileName	PDF のファイルパス
pdfData	バイト配列の PDF データ
length	PDF データのサイズ（バイト長）
password	暗号化に使用したパスワード

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。

5.6 新規にPDF文書を作成してオープン

PDF Stamper はPDF文書を新規に作成してオープンできます。

作成されるPDF文書は1ページで構成されその内容は白紙です。電子印鑑の素材（カスタムスタンプ）を作成する場合などに利用できます。

```
int OpenNewDoc(         /*.NET 開発環境のみ*/
    float       width,
    float       height,
    int         rotate
);

int MlpOpenNewDoc(      /*C/C++開発環境のみ*/
    float       width,
    float       height,
    int         rotate,
    void        *fill
);
```

引数

width	作成されるページの幅　単位は[ポイント]（1ポイント=1/72 インチ）
height	作成されるページの幅　単位は[ポイント]（1ポイント=1/72 インチ）
rotate	作成されるページの回転角　0、90、180、270を指定します。
fill	NULL を指定します。

5.7 PDF文書の情報

PDF Stamper は開いたPDF文書から情報を取得できます。

5.7.1 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 PageCount();
```

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。

5.7.2 PDF文書の許可（パーミッション）フラグ

PDF文書にはその文書を開いた際に変更や印刷の可否を指定するフラグがあります。PDF Stamper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。このフラグを評価するには、このPDF文書の暗号化レビジョンも必要です。暗号化レビジョンを取得するのは「5.5.3PDF文書の暗号化レビジョン」を参照してください

```
int GetDocumentInfo(DocumentOpt.PERMISSION_FLAG, out int flag);
```

引数

第一引数　　C/C++開発環境の場合は MLP_PERMISSION_FLAG を指定
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。

5.7.3 PDF文書の暗号化レビジョン

PDF文書が暗号化されている場合は印刷の許可などのフラグ（パーミッションフラグ）を確認しなければならない場合があります。以下ではこのフラグの評価に必要な暗号化レビジョンを取得します。（パーミッションフラグの取得は「5.5.2 PDF文書のパーミッションフラグ」を参照）

```
int GetDocumentInfo(Document.ENCRYPT_REVISION, out int rev);
```

引数
第一引数　　C/C++開発環境の場合は MLP_ENCRYPT_REVISION を指定
rev　　暗号化のレビジョン番号

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。

暗号化レビジョンの詳細は「PDF Reference」 3.5.2 Standard Security Handler を参照してください。
暗号化レビジョンは、PDF文書の「Standard Security Handler」ディクショナリ内のRキーに指定された値です。

5.8 指定の単一ページを画像形式ファイルに変換

PDF Stamperはスタンプ付加の前後の指定された単一ページを画像に変換できます。
ただし、変換された画像の解像度常に72DPI（Dot per Inch）で非可逆圧縮の品質は75%に固定され、それらの変更はできません。
ページ番号指定は最初のページを1とします。なお、ページ番号0はPDF文書の先頭ページ、負数のページ番号は最終ページとみなします。
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 CreatePict(
    int         pageNumber,
    string      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
拡張子が“bmp”の場合、BMP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。

5.9 出力画像の色空間を指定する

PDF文書のページを画像に変換する際に色空間をグレースケール、白黒ディザ、白黒（2階調）画像に変換します。PDF文書の色空間は変更されません。
規定ではRGBカラーの画像に変換します。

5.9.1 RGB カラー画像を指定

PDF文書のページをRGBカラーの画像に変換するよう設定します。
RGBカラー画像指定は既定値です。

```
int SetPictureRGB();
```

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。

5.9.2 CMYK カラー画像を指定

PDF文書のページをRGBカラーの画像に変換するよう設定します。
PDF文書で指定された色はすべてCMYKカラーに変換されますが、指定された画像形式がCMYK色空間をサポートしていない場合はRGB色空間になります。

```
int SetPictureCMYK();
```

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。

5.9.3 グレースケール画像を指定

PDF文書のページをグレースケールの画像に変換するよう設定します。

```
int SetPictureGray();
```

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。

5.9.4 白黒ディザ画像を指定

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

```
int SetPictureDither();
```

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。

5.9.5 白黒2階調画像を指定

PDF文書のページを白黒2階調の画像に変換するよう設定します。

```
int SetPictureBW();
```

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。

5.10 出力画像の透明背景指定

PDF文書のページを画像に変換する際にその背景を透明に設定します。

画像に変換した電子印鑑をワープロソフトなどで文書に貼り付ける際に、電子印鑑の背景部分が透明なため押印したような表現ができます。

```
int SetPicture(PictureOpt.TRANSPARENT_PICT
    PictureOpt   opt,
    int          flag
);
```

引数

opt	TRANSPARENT_PICT を指定します。 （C/C++開発環境の場合は MLP_TRANSPARENT_PICT）
flag	1（True）を指定するとそれ以降に作成される画像の背景が透明になります。 0（False）を指定すると背景が不透明になります。既定値は0（False）です。

注意

透明な背景は画像形式をPNG形式またはTIFF形式、色空間をRGB、CMYKまたはグレースケールを指定した場合に限り作成されます。

5.11 カラープロファイル

PDF Stamper はPDF文書のページを画像に変換する際にカラープロファイルを使った色空間の変換を行います。

規定ではライブラリ内部のカラープロファイルを使って色空間を変換しますが、カラープロファイルを指定して変換することができます。UnuseCMS()で簡易計算による色空間変換処理にすることができます。

```
int SetRGBProfile(              /*RGBカラープロファイル指定*/
    string    rgbpfn
);
int SetCMYKProfile(             /*CMYKカラープロファイル指定*/
    string    cmykpfn
);
```

```
int UnuseCMS();                    /*カラープロファイルの使用を解除*/
int UseCMS();                      /*カラープロファイルを使った色変換*/
```

引数

rgbpfn	RGBカラープロファイルのファイル名
cmykpfn	CMYKカラープロファイルのファイル名

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。

注意：カラープロファイルはシステムフォルダーまたはカレントフォルダーから検索されます。システムフォルダー内のカラープロファイルが優先して利用されます。

5.12 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 Stamper はこのサイズで（画像解像度を72DPIとして）画像に変換します。

5.12.1 ページサイズの種類

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 Stamper はページの大きさを取得する際や画像に変換する際の境界を指定できます。

```
int SetPicture(
    PictureOpt   opt
);
```

引数

opt	利用する大きさの種類を指定します。	
	PICTURE_USE_MEDIABOX	MediaBox で指定された大きさを使います。（既定値） （C/C++では MLP_PICTURE_USE_MEDIABOX）
	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。 （C/C++では MLP_PICTURE_USE_CROPBOX）
	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。 （C/C++では MLP_PICTURE_USE_BLEEBOX）
	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。 （C/C++では MLP_PICTURE_USE_TRIMBOX）
	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。 （C/C++では MLP_PICTURE_USE_ARTBOX）

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。

5.12.2 画像のピクセルサイズを取得

生成される画像のピクセルサイズを取得します。

```
int GetPicturePixel(
    int       pageNum,
    out int   width,
    out int   height
);
```

引数

pageNum	サイズを取得するページの番号
width	画像の幅（ピクセル単位）
height	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（既定の場合）。

5.12.3 ページの境界ボックスの値を取得

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 は必ずPDF文書に指定されますが、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各値（単位[ポイント]；1ポイント=1/72 インチ）を所得します。境界ボックスの幅は `right - left` で計算でき、高さは `top - bottom` で計算できます。

`GetPageBoundary` 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、`GetPageRawBoundary` 関数は PDF 文書から読み取った値をそのまま取得します。

境界ボックスの詳細は「PDF Reference version1.7」10.10.1 Page Boundaries を参照してください。

```
int GetPageBoundary(
    int         pageNum,
    BoundaryBox kind,
    out float   userUnit,
    out float   left,
    out float   bottom,
    out float   right,
    out float   top
);

int GetPageRawBoundary(
    int         pageNum,
    BoundaryBox kind,
    out float   userUnit,
    out float   left,
    out float   bottom,
    out float   right,
    out 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	ページ左辺のX座標位置を取得
bottom	ページ下辺のY座標位置を取得
right	ページ右辺のX座標位置を取得
top	ページ上辺の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。

注意:

各座標位置は userUnit 倍されることなく取得します。userUnit 値が宣言されたPDF文書を取り扱う場合にはご注意ください。

5.12.4 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 GetPageRotation(
    int         pageNum,
    out 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。

5.13 スタンプ設定

PDF文書に付加するスタンプはXML形式設定ファイルで指定します。
設定ファイルの詳細は「6.0 スタンプ設定データ」を参照してください。

5.13.1 スタンプ設定データ

スタンプ設定データ(XML形式データ)を指定します。
この設定データを使ってスタンプを付加するには PaintStamp メソッドなどの実行が必要です。
文字コードは BOM(Byte Order Mark)によって適切に処理されます。

```
int SetStampData(
    string      data
    int         length
```

```
);

int SetStampData(                /*C#開発環境のみ*/
    string    data
);
```

引数

data	スタンプ設定データ
length	data のバイト数 BOM(Byte Order Mark)を含みます。

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。

5.13.2 スタンプ設定ファイル

スタンプ設定ファイル(XML形式データ)を指定します。
この設定ファイルを使ってスタンプを付加するには PaintStamp メソッドなどの実行が必要です。
文字コードは BOM(Byte Order Mark)によって適切に処理されます。

```
int SetStampFile(
    string    filePath
);
```

引数

filePath	スタンプ設定ファイル

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。

5.13.3 スタンプ設定を削除

指定されたスタンプ設定をすべて削除します。

```
int ResetStamp( );
```

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。

5.14 スタンプ付加指定

初期設定ファイル(データ)またはスタンプ設定ファイル(データ)で設定したスタンプを現在のPDF文書に付加します。
貼り付けられるスタンプはPDF文書ページの指定に関わらず正立で貼り付けられます。
設定ファイルの詳細は「6.0 スタンプ設定データ」を参照してください。

5.14.1 全てのスタンプを付加

初期設定データおよびスタンプ設定データで指定された全てのスタンプを現在のPDF文書に付加します。

```
int PaintStamp();
```

引数

ありません。

戻り値

成功すると、0(ゼロ)または正数が戻り、失敗すると負数(エラーコード)が戻ります。

5.14.2 データで指定されたスタンプを付加

スタンプ設定データで指定された全てのスタンプを現在のPDF文書に付加します。
以前に設定されたスタンプに加えて、このメソッドで指定されたスタンプが付加されます。
文字コードは BOM(Byte Order Mark)によって適切に処理されます。

```
int PaintStampData(
    string    data,
    int       length
);
```

引数

data	スタンプ設定データ
length	data のバイト数 BOM(Byte Order Mark)を含みます。

戻り値

成功すると、0(ゼロ)または正数が戻り、失敗すると負数(エラーコード)が戻ります。

5.14.3 ファイルで指定されたスタンプを付加

以前に設定されたスタンプに加えて、引数ファイルのスタンプを現在のPDF文書に付加します。
文字コードは BOM(Byte Order Mark)によって適切に処理されます。

```
int PaintStampFile(
    string    filePath
);
```

引数

filePath	スタンプ設定ファイル

戻り値

成功すると、0(ゼロ)または正数が戻り、失敗すると負数(エラーコード)が戻ります。

5.14.4 初期設定データで指定された名前付きスタンプを付加

初期設定ファイルの名前で指定されたスタンプを現在のPDF文書に付加します。

```
int PaintStampName(
    string    name
);
```

引数

name	付加するスタンプの名前

戻り値

成功すると、正数が戻り、失敗するとエラーコード（負数）が戻ります。

5.16 現在のPDFデータをPDFファイルや画像形式ファイルに出力

現在のPDFデータをPDFファイルや画像に変換した画像形式ファイルとして出力します。

5.16.1 PDFファイルに出力

現在のPDFデータをPDFファイルとしてファイルに出力します。

```
int SavePDF(
    string    filePath
);
```

引数

filePath	出力するPDFファイルのパス名

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。

5.16.2 画像形式データとしてファイルに出力

現在のPDFデータを画像形式ファイルとしてファイルに出力します。

指定される画像ファイル名の拡張子によって生成される画像の形式が変わります。拡張子がPNG、JPG, TIFの場合にそれぞれ、PNG形式画像、JPEG形式画像、TIFF形式画像となります。

なお、画像の解像度や圧縮率などは変更できません。また複数ページの画像を作成することもできません。

```
int CreatePict(
    int       pageNumber,
    string    filePath
);
```

引数

pageNumber	画像に変換するPDF文書のページ番号
filePath	出力する画像ファイルのパス名 ファイル名の拡張子によって作成される

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。

5.16.3 SVG画像形式データとしてファイルに出力

現在のPDFデータをSVG(Scalable Vector Graphics)画像形式ファイルとしてファイルに出力します。

SVG画像として出力できるのは新規作成文書にスタンプを張り付けたPDF文書だけです。

```
int PageToSvg(
    int       pageNumber,
    string    filePath
);
```

引数

pageNumber	SVG画像に変換するPDF文書のページ番号
filePath	出力するSVG画像ファイルのパス名

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。

5.17 現在のPDF文書を画像データに変換して取得

現在オープンしているPDF文書を画像に変換したピクセルデータとして取得します。
ただし、画像化する際に指定できる解像度は72DPI が最大です。

5.17.1 指定されたページを画像に変換

現在のPDFデータの指定されたページをライブラリ内部に画像として変換します。
単一ページのみ画像に変換できます。

```
int CreatePictPageRange(
    int         pageNumber,
    int         pageNumber
);
```

引数

pageNumber	画像に変換するPDFデータのページ番号

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。

5.17.2 変換された画像データを取得

現在のPDFデータの指定されたページをライブラリ内部に画像として変換します。

```
int GetPictFromMemEx(
    uint        csFlag,
    uint        boundary,
    void*       pixels,
    int*        width,
    int*        height,
    int*        stride,
    int*        nOfColors
);
```

引数

csFlag	画像を生成する際のフラグです。必ず"1"を指定します。
boundary	画像の行データの開始アドレスを指定された倍数のアドレスにします。
pixels	ピクセルデータ .NET 開発環境では IntPtr 型で指定します。 C/C++開発環境では unsigned char*型で指定します。
width	画像の行当たりの画素数
height	画像の高さ（行データの総数）
stride	画像の行データのバイト数
nOfColors	画像データの色数

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。

5.17.3 変換された画像データを取得（.NET 開発環境専用）

現在のPDFデータの指定されたページを System.DrawingBitmap に変換可能なデータとして取得します。このメソッドを利用する場合は変換される画像を RGB 画像で作成しなければなりません。

```
int GetPictBufPageEx(
    int          pageNumber,
    uint         csFlag,
    uint         boundary,
    out IntPtr   pixels,
    out int      width,
    out int      height,
    out int      stride,
    out int      nOfColors
);
```

引数

pageNumber	画像に変換するPDFデータのページ番号です。
csFlag	画像を生成する際のフラグです。 1：背景が不透明な（画素当たり24ビット）画像を取得します。 5：背景が透明な（画素当たり32ビット）画像を取得します。
boundary	画像の行データの開始アドレスを指定された倍数のアドレスにします。
pixels	ピクセルデータ .NET 開発環境では IntPtr 型で指定します。 C/C++開発環境では unsigned char*型で指定します。
width	画像の行当たりの画素数
height	画像の高さ（行データの総数）
stride	画像の行データのバイト数
nOfColors	画像データの色数

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。

5.18 フォントフォルダーの指定

PDF Stamper はシステムにインストールされたフォント以外に、インストールされていないフォントファイルを使うことができます。

5.18.1 フォントファイルが格納されたフォルダーを指定

フォントファイルが格納されたフォルダーを指定します。
指定のフォントをシステムフォルダーで検索できなかった場合は、このフォルダーでフォントを検索します。
サブフォルダーも検索します。

```
int UserFontFolder(
    string     folderName
);
```

引数

folderName	フォントファイルが格納されたフォルダー名

戻り値
成功すると、ゼロが戻り、失敗するとエラーコード（負数）が戻ります。

5.18.2 フォントファイルのフォント名リスト作成

現在利用可能なフォント名をリストにします。
システムにインストールされたフォントばかりではなく、フォントが格納されたフォルダー内のフォントも含めてリストにします。リストはライブラリ内に作成され GetFontsList( )でリストとして取得できます。

```
int CreateFontList( );
```

引数
ありません

戻り値
成功すると、ゼロが戻り、失敗するとエラーコード（負数）が戻ります。

5.18.3 現在利用できるフォント名のリスト取得

現在利用可能なフォント名のリストを取得します。
事前に CreateFontsList( )によるリスト作成が必要です。

```
int GetFontsList (
    string[]  fontsList
);
```

引数
fontsList　　フォント名が格納された配列

戻り値
成功すると、ゼロが戻り、失敗するとエラーコード（負数）が戻ります。

5.19 文書のクローズ（PDF文書処理の終了）

オープンしたPDF文書の処理を終了します。

```
void CloseDoc();
```

引数
ありません。

戻り値
ありません。

5.20 ライブラリの終了

ライブラリの使用を終了します。

```
void Uninitialize();
```

引数
ありません。

戻り値
　　ありません。

6.0 スタンプ設定データ

PDF Stamper は初期化データやスタンプ設定データでPDF文書に付加するスタンプ（電子印鑑など）をあらかじめ指定します。各データはファイルで指定することもできます。

6.1 初期化データで指定

初期化データはXML形式データで、<init>をルート要素とします。
初期化データ（またはファイル）の指定手順は「5.3 初期化ファイル（または初期化データ）を指定する」を参照してください。

<init>　　初期化

属性および属性値を以下のとおり指定します。この属性は必須です。
<init>の子要素として<pdfstamp>を指定します。<pdfstamp>要素を初期化データで指定する場合は<pdfstamp>に属性 xmlns の指定を省略できます。

```
xmlns="http://www.trustss.co.jp/2019/Structure/"
```

XML宣言文はその属性と共に無視されますが、データの先頭にある BOM（Byte Order Mark）によって初期化データの文字コードが適切に認識されます。ただし、符号化形式（文字コード）は Shift_JIS、UTF-8、UTF-16 のいずれかです。

以下は初期化データの例です。

```
<?xml version="1.0" ?>
<init xmlns="http://www.trustss.co.jp/2019/Structure/">
  <pdfstamp>
    <stamp>
      <text value="電子印鑑" />
      <image src="image.png" />
    </stamp>
  </pdfstamp>
</init>
```

6.2 スタンプ設定データで指定

スタンプ設定データはXML形式データで、<pdfstamp>をルート要素とします。
スタンプ設定データ（またはファイル）の指定手順は「5.3 初期化ファイル（または初期化データ）を指定する」を参照してください。

<pdfstamp>　　スタンプ設定

属性および属性値を以下のとおり指定します。スタンプ設定データではこの属性を省略できません。

```
xmlns="http://www.trustss.co.jp/2019/PDFStamp/"
```

以下はスタンプ指定データの例です。

```
<?xml version="1.0" ?>
```

```
<pdfstamp xmlns="http://www.trustss.co.jp/2019/PDFStamp/">
  <stamp>
    <text value="電子印鑑" />
    <image src="image.png" />
  <stamp>
</pdfstamp>
```

XML宣言文はその属性と共に無視されますが、データの先頭にある BOM(Byte Order Mark)によって初期化データの文字コートが適切に認識されます。ただし、符号化形式(文字コード)は Shift_JIS、UTF-8、UTF-16 のいずれかです。

7.0 スタンプ設定データ詳細

PDF文書に付加するスタンプは常にXML設定ファイル(またはデータ)で指定します。スタンプ内容の文字列、画像、図形などは<pdfstamp>タグの子要素で指定しその詳細をタグの属性で指定します。

この要素を初期設定ファイル(データ)で指定する場合は、<init>の子要素として、スタンプ設定ファイル(データ)として指定する場合はルート要素として記載します。

各スタンプの詳細は要素の属性名と属性値で指定します。

なお、この章で示す図は機能の説明を目的としたもので、実際の表示と異なります。

7.1 スタンプ指定ルート要素

初期設定ファイル(データ)またはスタンプ設定ファイル(データ)でスタンプを指定する最上位の要素です。この要素がルート要素の場合は以下の属性を指定しなければなりません。

```
<pdfstamp>    スタンプ設定

  xmlns="http://www.trustss.co.jp/2019/PDFStamp"
```

7.2 スタンプ

<stamp>タグで文字列や画像、枠などで構成されたひとつのスタンプを設定します。このタグに文字列、画像、図形(矩形または円形)、枠線を子要素として記載します。

```
<stamp>

  name="文字列"                (省略可)
      各スタンプの識別子
      文字列にはASCII文字のみが使用できます。
      PDF文書にスタンプを付加するスタンプをこの識別子を使って指定できます。

  page="<page_set>"            (必須)
      スタンプを付加するページ
      以下の形式で指定できます。
      <page_set> = <page_range> [","  <page_range>]
      <page_range> = <n> | first | last | not_first | not_last | even | odd
      | all

      ・n: ページ番号(整数値) 先頭のページは1です。
      ・first: 先頭のページ n=1 と同じです。
      ・last: 先頭のページ
      ・not_first: 先頭以外の全てのページ
      ・not_last: 最終ページ以外のすべてのページ
      ・even: 偶数番めのページ
```

・ odd: 奇数番めのページ
・ all: すべてのページ

layer="<place>"　　(省略可)
スタンプを貼り付けるレイヤ
以下を指定できます。
・ annotation: 注釈として付加します。
・ foreground: ページコンテンツに前景として(最前面に)付加します。
・ background: ページコンテンツに背景として(最背面に)付加します。

pos="<a>" | "<x> <y>"　(省略可)
スタンプの貼り付け位置
各値は実数値です。
省略するとPDFページの左下原点位置となります。
・ a: 横方向(x)および縦方向(y)の座標位置
・ x: 横方向の座標位置
・ y: 縦方向の座標位置
既定値(原点)はページの左下です。

alpha="<f>"　　(省略可)
スタンプの非透明度
・ f: 0～1.0の正実数で指定します。既定値は1.0(不透明)

scale="<a>" | "<x> <y>" (省略可)
スタンプが指定の倍率で拡大・縮小されます。
各値は実数値です。
・ a: 横方向(x)および縦方向(y)に同じ倍率が適用されます。
・ x: 横方向の倍率
・ y: 縦方向の倍率

blendmode="<BlendMode>" (省略可)
スタンプ描画時の Blend Mode (layer=annotation の場合のみ指定可)
既定値は"Normal"
詳細は「PDF Reference version1.7」を参照してください。

creator="作成者"　　(省略可)
スタンプの作成者名 (layer=annotation の場合のみ指定可)

subject="タイトル"　　(省略可)
スタンプのタイトル (layer=annotation の場合のみ指定可)

creation="<Y>/<M>/<D> <h>:<m>:<s>"　(省略可)
mod="<Y>/<M>/<D> <h>:<m>:<s>"　(省略可)
スタンプの作成日時および変更日時 (layer=annotation の場合のみ指定可)
各値は整数値です。
省略時は現在日時となります。
・ Y: 西暦年
・ M: 月(1～12)
・ D: 日(1～31)
・ h: 時(0～23)
・ m: 分(0～59)
・ s: 秒(0～59)

unlock="true" | "false"　　（省略可）
スタンプを移動回転などの変更を可能にします。（layer=annotation の場合のみ指定可）
省略時は false となり、スタンプの移動や回転などの変更ができません。

7.3 文字列

<text>タグでスタンプの文字列を描画します。このタグは<stamp>タグの子要素です。
描画された文字列は原点を左下とした境界ボックスを有します。（下図参照）

<text>

value="テキスト文字列"　　（必須）
スタンプの文字列指定
文字列には漢字を含めることができます。

wmode="horizontal" | "vertical"
（省略可）
文字列の横書き（horizontal）・縦書き（vertical）指定
縦書きテキスト（グリフ）を付加する場合は使用するフォントに少なくともGSUBテーブルが必要です。
規定は横書き（"horizontal"）です。

font-name="フォント名"　　（省略可）
文字列のフォント
font-file の指定がある場合は、そのファイル内で指定フォントを検索します。
font-file の指定がない場合（または指定ファイルで検索できない場合）はシステム内で指定のフォントを検索します。
指定が省略された場合や指定フォントが検索できない場合は、システムの規定フォントを使用します。
フォントはアウトラインフォントでなければなりません。指定されたフォントがアウトライングリフを持たない場合は文字を表示しません。

font-file="フォントファイルパス"
（省略可）
文字列のフォントのファイルパス名
font-name で指定されたフォントをこのファイルから検索します。
font-name が省略され font-file が指定された場合はこのファイル内のフォントを使用します。
フォントはアウトラインフォントでなければなりません。指定されたフォントがアウトライングリフを持たない場合は文字を表示しません。

char-space="<f>"　　（省略可）
指定文字列を表示する場合の文字間隔の倍率
既定値は 1.0（フォントの指定値を使用します。）
・ f： 正実数値

scale="<a>" | "<x> <y>"（省略可）
文字列の拡大・縮小率
各値は正の実数値です。
既定値は 1.0（等倍）
・ a： 横方向（x）および縦方向（y）の拡大・縮小率
・ x： 正実数値　横方向の拡大・縮小率

・ y: 正実数値 縦方向の拡大・縮小率

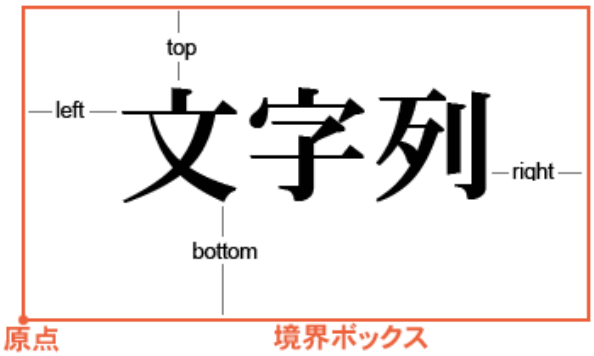
offset="<a>" | "<x> <y>"
(省略可)

文字列の境界ボックス内での移動量(単位[ポイント])
境界ボックスの原点やの大きさは変化しません。
移動によって境界ボックスからはみ出た部分はクリップされ(非表示になり)ます。(下図参照)
各値は実数値です。
省略時は移動しません。
・ a: 横方向(x)および縦方向(y)の移動量
・ x: 横方向の移動量
・ y: 縦方向の移動量

expand="<a>" | "<x> <y>" | "<left> <right> <y>" |
"<left> <right> <top> <bottom>"
(省略可)

文字列境界ボックスの拡張量(単位[ポイント])
原点が移動することなく境界ボックスの大きさが変わります。(下図参照)
各値は実数値です。
省略時は拡張しません。
・ a: 上下(top bottom)および左右(left right)の拡張量
・ x: 上下(top bottom)の拡張量
・ y: 左右(left right)の拡張量
・ l: 左(left)の拡張量
・ r: 右(right)の拡張量
・ t: 上(top)の拡張量
・ b: 下(bottom)の拡張量

境界を拡張すると、右図のように原点が移動することなく文字列が移動し境界ボックスの大きさが変わります。ます。



border-color="<G>" | "<r> <g> <b>" | "<c> <m> <y> <k>"
(省略可)

境界ボックスを指定色で描画
各値は 0～1.0 の正実数値で指定します。
省略された場合は描画しません。
・ G: グレースケール色
・ r: RGB色空間の赤
・ g: RGB色空間の緑
・ b: RGB色空間の青
・ c: CMYK色空間のシアン
・ m: CMYK色空間のマゼンタ
・ y: CMYK色空間のイエロー
・ y: CMYK色空間の墨

background-color="<G>" | "<r> <g> <b>" | "<c> <m> <y> <k>"
(省略可)

境界ボックス内を指定色で塗りつぶします。
各値は 0～1.0 の正実数値で指定します。
省略された場合は描画しません。

色指定は、border-color 属性を参照してください。

border-width="<f>"　　（省略可）
境界線の線幅（単位［ポイント］）
既定値は 1.0
・ f： 実数の線幅値 単位はポイント（1 ポイント=1/72 インチ）です。

visible="true" | "false"　　（省略可）
"false"の場合は指定された文字列はスタンプに追加されません。
既定値は"true"

7.3 画像

<image>タグでスタンプに画像を描画します。このタグは<stamp>タグの子要素です。
描画された画像は原点を左下とした境界ボックスを有します。（右図参照）

<image>

file-name="画像ファイルパス"
（省略可）

size="<a>" | "<w> <h>"（省略可）
画像表示サイズ変更（単位［ポイント］）
各値は正の実数値です。
省略時は画像の実寸（ピクセルサイズ）
・ a： 画像の横方向（w）および縦方向（h）サイズ
・ w： 画像の横サイズ
・ h： 画像の縦サイズ
w または h が0（ゼロ）の場合は画像の縦横比が維持されます。
画像サイズ0（ゼロ）は指定できません。

scale="<a>" | "<w> <h>"（省略可）
画像を表示する際の画像倍率
各値は正の実数値です。
既定値は 1.0（等倍）
・ a： 画像の横方向（w）および縦方向（h）倍率
・ w： 画像の横倍率
・ h： 画像の縦倍率
値0（ゼロ）は指定できません。

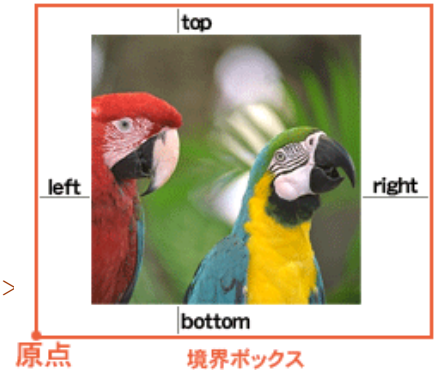
offset="<a>" | "<x> <y>"
（省略可）
画像の境界ボックス内での移動量（単位［ポイント］）
境界ボックスの原点や大きさは変化しません。
移動によって境界ボックスからはみ出た部分はクリップされ（非表示になり）ます。（上図参照）
各値は実数値です。
省略した場合は移動しません。
・ a： 横方向（x）および縦方向（y）の移動量
・ x： 横方向の移動量
・ y： 縦方向の移動量

expand="<a>" | "<x> <y>" | "<left> <right> <top> <bottom>"

（省略可）

画像境界ボックスの拡張量（単位［ポイント］）
原点が移動することなく境界ボックスの大きさが変わります。（下図参照）
各値は実数値です。
省略した場合は拡張しません。

- a: 上下（top bottom）および左右（left right）の拡張量
- x: 上下（top bottom）の拡張量
- y: 左右（left right）の拡張量
- left: 左の拡張量
- right: 右の拡張量
- top: 上の拡張量
- bottom: 下の拡張量




```
border-color="<G>" | "<r> <g> <b>" | "<c>
```

（省略可）

境界ボックスを指定色で描画
各値は 0～1.0 の正実数値で指定します。
省略された場合は描画しません。

- G: グレースケール色
- r: RGB色空間の赤
- g: RGB色空間の緑
- b: RGB色空間の青
- c: CMYK色空間のシアン
- m: CMYK色空間のマゼンタ
- y: CMYK色空間のイエロー
- y: CMYK色空間の墨

```
background-color="<G>" | "<r> <g> <b>" | "<c> <m> <y> <k>"
```

（省略可）

境界ボックス内を指定色で塗りつぶし
各値は 0～1.0 の正実数値で指定します。
省略された場合は描画しません。
色指定は border-color 属性を参照してください。

```
border-width="<f>"
```

（省略可）

境界線の線幅
既定値は 1.0

- f: 正実数の線幅値　単位はポイント（1 ポイント=1/72 インチ）

7.4 矩形

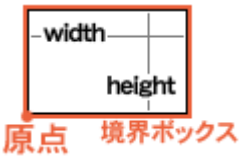
<rectangle>タグでスタンプに矩形を描画します。このタグは<stamp>タグの子要素です。

```
<rectangle>
```

```
size="<a>" | "<width> <height>"
```

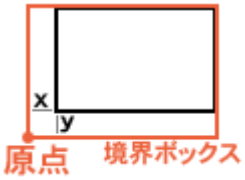
（必須）

矩形の大きさ（単位は［ポイント］）
大きさ0（ゼロ）は指定できません。
指定された矩形は左下を原点とした境界ボックスを有しています。（右図参照）



各値は正の実数値です。
・ a： 矩形の幅（width）と高さ（height）
・ width： 矩形の幅
・ height： 矩形の高さ

pos="<a>" | "<x> <y>"　（省略可）
矩形を移動させて描画
境界ボックスの大きさが変わります。（右図参照）
各値は実数値です。
省略すると移動しません。
・ a： 矩形を横（x）及び縦（y）方向に移動
・ x： 矩形を横方向に移動
・ y： 矩形を縦方向に移動



stroke-color="<G>" | "<r> <g> <b>" | "<c> <m> <y> <k>"
（省略可）
矩形を描画する線の色指定
各色は 0～1.0 の正実数値で指定します。
省略するとPDF文書に色が指定されませんので、PDFの規定色となります。
・ G： グレースケール色
・ r： RGB色空間の赤
・ g： RGB色空間の緑
・ b： RGB色空間の青
・ c： CMYK色空間のシアン
・ m： CMYK色空間のマゼンタ
・ y： CMYK色空間のイエロー
・ y： CMYK色空間の墨

line-width="実数値"　（省略可）
矩形の線幅（単位［ポイント］）

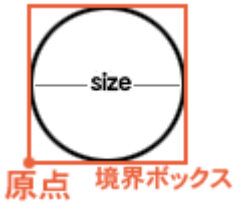
dash="<dash-list>"　（省略可）
点線指定
描画サイズと間隙サイズ指定を10項目まで指定できます。

7.5 円形

<circle>タグでスタンプに円形を描画します。このタグは<stamp>タグの子要素です。

<circle>

size="正実数"　（必須）
円形の直径（単位［ポイント］）
大きさ0（ゼロ）は指定できません。
指定された矩形は左下を原点とした境界ボックスを
有しています。（右図参照）

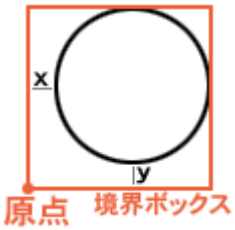


sub="正実数"　（省略可）
デート印などで使用される日付表示枠を円形に追加
日付枠の高さを指定します。（単位［ポイント］）
位置は指定できません。（右図参照）

省略すると日付枠を表示しません。

pos="<a>" | "<x> <y>"　（省略可）

矩形を移動させて描画
各値は実数値です。
境界ボックスの大きさが変わります。（右図参照）
省略すると移動しません。
・ a： 矩形を横（x）及び縦（y）方向に移動
・ x： 矩形を横方向に移動
・ y： 矩形を縦方向に移動



stroke-color="<G>" | "<r> <g> <b>" | "<c> <m> <y> <k>"
（省略可）

矩形を描画する線の色指定
各色は 0～1.0 の正実数で指定します。
省略するとPDF文書に色が指定されませんので、PDFの規定色となります。
・ G： グレースケール色
・ r： RGB色空間の赤
・ g： RGB色空間の緑
・ b： RGB色空間の青
・ c： CMYK色空間のシアン
・ m： CMYK色空間のマゼンタ
・ y： CMYK色空間のイエロー
・ y： CMYK色空間の墨

line-width="正実数"　（省略可）

矩形の線幅（単位［ポイント］）

dash="<dash-list>"　（省略可）

dash-list="正実数" | "正実数 ..."
点線指定（単位［ポイント］）
各値は正実数値です。
描画サイズと間隙サイズ指定を10項目まで指定できます。

7.6 枠線

<border>タグで<stamp>内の全要素を包含する境界ボックスを基にした枠線描画を指定します。このタグは<stamp>要素の子要素です。

描画の基となる境界ボックスは各<stamp>に含まれる要素（<text>、<image>、<rectangle>、<circle>）の境界ボックスを完全に内包する最小サイズのものです。

<border>

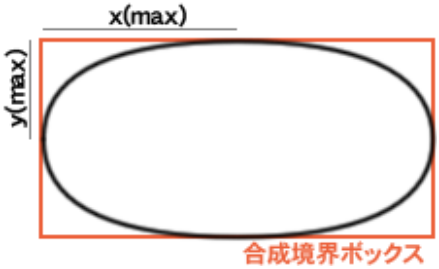
color="<G>" | "<r> <g> <b>" | "<c> <m> <y> <k>"
（省略可）

枠線を描画する線の色指定
各色は 0～1.0 の正実数で指定します。
省略するとPDF文書に色が指定されませんので、PDFの規定色となります。
・ G： グレースケール色
・ r： RGB色空間の赤

・ g： RGB色空間の緑
・ b： RGB色空間の青
・ c： CMYK色空間のシアン
・ m： CMYK色空間のマゼンタ
・ y： CMYK色空間のイエロー
・ y： CMYK色空間の墨

width="正実数"　　（省略可）
枠線の線幅（単位［ポイント］）

corner="<a>" | "<x> <y>"
（省略可）
四隅の丸みをつけます。（単位［ポイント］）
（右図参照）
各値は正実数です。
省略すると丸みをつけません。
・ a： 横方向（x）および縦方向（y）の丸みサイズ
・ x： 横方向の丸みサイズ
・ y： 縦方向の丸みサイズ
各値に max を指定すると最大の丸みを持った（直線部分のない）枠線が描画されます。（下図参照）



dash= "<dash-list>"　　（省略可）
dash-list="正実数" | "正実数 ..."
点線指定（単位［ポイント］）
各値は正実数値です。描画線長、描画間隔の順に指定します。
描画サイズと間隙サイズ指定を10項目まで指定できます。

dash-offset= "実数値"　　（省略可）
点線は合成境界ボックスの下（Bottom）の中心（Center）から時計回りに描画しますが、この値を指定しますと描画開始位置を変更します。（単位［ポイント］）

clip="true" | "false"　（省略可）
枠線の外側へのスタンプ内容の描画を許可（true）または禁止（false）します。
既定では枠外への描画を許可します。（下図参照）

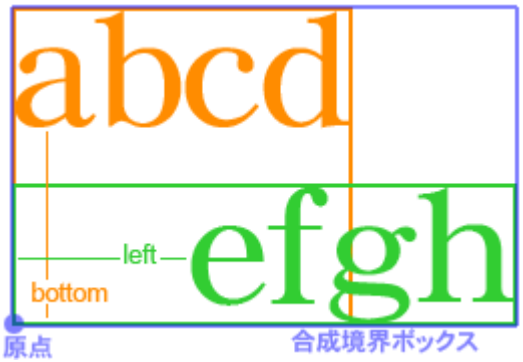
clip="false"

clip="true"

7.7 複数行の文字列を描画する

スタンプに複数行の文字列を含める例を以下に記します。

```
<pdfstamp xmlns="http://www.trustss.co.jp/2019/PDFStamp">
  <stamp>
    <text value="abcd" expand="0 0 0 13" />
    <text value="efgh" expand="20 0 0 0" />
  </stamp>
</pdfstamp>
```



文字列「abcd」の境界ボックスを下側（bottom）に拡張することによって各行が重ならずに描画されます。

「efgh」の境界ボックスは左側（left）に拡張されていますので、インデントされたように描画されます。

8.0 著作権

「PDF Stamper 説明書」および各種ライブラリは株式会社トラスト・ソフトウェア・システムの著作物です。著作者に無断で各種媒体への転載などは固くお断りいたします。