PDFlib GmbH

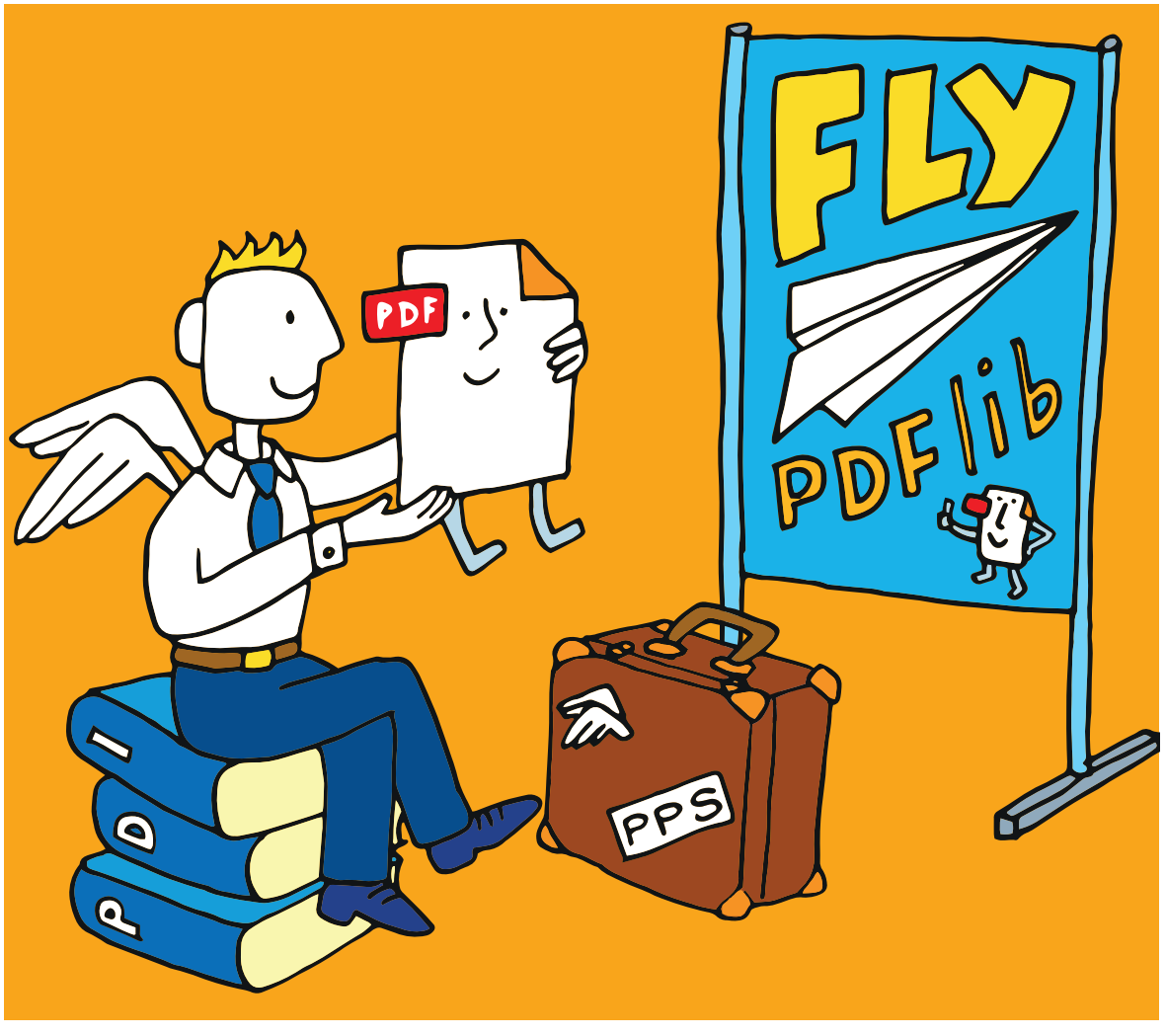
PDFlib, PDFlib+PDI, PPS

A library for generating PDF on the fly

Version 10.0.0

API リファレンス

C・C++・Java・.NET・.NET Core・Objective-C・
Perl・PHP・Python・RPG・Ruby 用



PDFlib GmbH
Franziska-Bilek-Weg 9, 80339 München, Germany
www.pdflib.com
電話 *+49 • 89 • 452 33 84-0*

jp.sales@pdflib.com
jp.support@pdflib.com（ライセンス番号をお知らせください）

目次

1 プログラミングコンセプト

1.1 オプションリスト

オプションリストは、さまざまな API メソッド呼び出しを制御できる、強力ながらも簡便な手段です。多くの API メソッドで利用することができ、メソッド引数を大量に与える必要がありません。略して ***optlist*** ともいいます。任意の数のオプションを記述した文字列です。さまざまなデータ型をサポートしており、リストなど複合データを持たせることもできます。必要とするキーワードと値を文字列連結していくことによって、多くの言語バインディングにおいて、容易に optlist を構築できます。

バインディング

.NET 言語バインディング：C# の ***AppendFormat( )*** StringBuilder メソッドでは中括弧 ***{ }*** を用いて書式項目を表し、おのおの文字列表現へ置換されますので、プログラマーは留意する必要があります。一方、***Append( )*** メソッドでは中括弧に特殊な意味を持たせていません。オプションリストの文法は中括弧を用いますので、***AppendFormat( )*** メソッドを使うか ***Append( )*** メソッドを使うかを適切に選ぶ必要があります。

1.1.1 文法

オプションリスト文法形式定義　オプションリストは以下の規則に従って作成する必要があります。

- オプションリスト内の要素（キーと値）はすべて、以下の区切りキャラクター1 個ないし複数によって区切る必要があります：スペース・タブ・キャリッジリターン・ニューライン・等号「=」。
- 最も外側を囲う中括弧は要素の内容として扱われません。文字列 ***{ }*** は、空の要素を表します。
- 最も外側の中括弧ペアの内側にある区切りキャラクターは、要素を区切らず、要素の一部になります。ですので、区切りキャラクターを内容として含んでいる要素は、中括弧で囲う必要があります。
- 要素が中括弧キャラクター（1 個ないし複数）を含んでいる場合には、各キャラクターの直前にバックスラッシュキャラクター 1 個を付けて保護する必要があります。
- 要素が、中括弧キャラクターの直前にバックスラッシュキャラクター1 個ないし複数を連続して含んでいるときは、そのキャラクター列の中の各バックスラッシュにもう 1 個のバックスラッシュキャラクターを付けて保護する必要があります。
- 要素に、開閉対応しない中括弧があるときは、この中括弧は直前にバックスラッシュキャラクターをつけて保護する必要があります。要素の閉じ中括弧の直前のバックスラッシュにも、直前にバックスラッシュキャラクターをつける必要があります。
- オプションリストにバイナリーゼロ値を含めてはいけません。

オプションは、この PDFlib API リファレンスの説明に従って、リスト値を内容として持てる場合があります。リスト値は、1 個ないし複数の要素を内容として持ちます（これらの要素自体がまたリストである場合もあります）。それらは上記の規則に従って区切られますが、ただしこの場合には等号が区切りキャラクターとして扱われない点だけが異なります。

注 オプション名（すなわちキー）がハイフンキャラクターを含むことはありません。オプションの説明の表の中では、オプション名が長いときにはハイフン区切りして示してある

場合がありますので、この点に注意してください。そのハイフンは、そのオプションをオプションリスト内で与える際にはなくす必要があります。

単純なオプションリスト　多くの場合オプションリストは、1 個ないし複数のキー / 値ペアを内容として持ちます。キーと値の間、およびキー / 値ペアどうしの間は、1 個ないし複数の空白キャラクター（スペース・タブ・キャリッジリターン・ラインフィード）で区切る必要があります。あるいは、キーと値の間は等号「=」で区切ることもできます：

```
key=value
key = value
key value
key1 = value1  key2 = value2
```

可読性を上げるために、キーと値の間には等号を用い、隣り合うキー / 値ペアの間には空白を用いることを推奨します。

オプションリストは左から右へ評価されていきますので、1 つのオプションを同じリストの中で複数回与えることもできます。その場合は、最後に出てきたものがそれより前のものをオーバーライドします。以下の例では、最初のオプション割り当てを次のものがオーバーライドして、オプションリスト処理完了後の ***key*** は値 ***value2*** となります：

```
key=value1 key=value2
```

リスト値　リストは、1 個ないし複数の値の間を区切ったものを内容として持ちます。これらの値は、単純値である場合もありますし、それ自体がさらにリストであることも可能です。リストは中括弧 *{}* で囲われ、リスト内の値の間は空白キャラクターで区切る必要があります。例：

```
dasharray={11 22 33}                    （数値3個を内容として持つリスト）
position={ center bottom }              （キーワード2個を内容として持つリスト）
```

リストは、内容としてリストをネストで持つことも可能です。この場合も、リストどうしの間は空白で区切る必要があります。隣り合う *}* キャラクターと *{* キャラクターとの間には区切りキャラクターを入れる必要がありますが、同じ種類の中括弧どうしの間では省略することもできます：

```
polylinelist={{10 20 30 40} {50 60 70 80}}     （リスト2個を内容として持つリスト）
```

リストの中にリストが 1 個だけあるときも、ネストされたリストの中括弧は省略してはいけません：

```
polylinelist={{10 20 30 40}}                   （ネストされたリスト1個を内容として持つリスト）
```

ネストされたオプションリストとリスト値　オプションのなかには、**オプションリスト**型または**オプションリストのリスト**型の値を持つことができるものがあります。**オプションリスト**型のオプションは、従属する 1 個ないし複数のオプションを内容として持ちます。**オプションリストのリスト**型のオプションは、ネストされた 1 個ないし複数のオプションリストを内容として持ちます。オプションリストをネストして扱う際に重要なことは、適切な数の中括弧を記述することです。以下にいくつかの例を挙げます。

オプション ***metadata*** の値がオプションリストであり、それ自体が 1 個のオプション ***filename*** を内容として持つ：

```
metadata={filename=info.xmp}
```

オプション *fill* の値がオプションリストのリストであり、それがオプションリスト 1 個を内容として持つ：

```
fill={{ area=table fillcolor={rgb 1 0 0} }}
```

オプション *fill* の値がオプションリストのリストであり、それがオプションリスト 2 個を内容として持つ：

```
fill={{ area=rowodd fillcolor={rgb 0 1 0} } { area=roweven fillcolor={rgb 1 0 0} }}
```

リストがオプションリスト 1 個を内容として持ち、その値の中に空白がある：

```
attachments={{filename={foo bar.xml} }}
```

リストが文字列 3 個を内容として持つ：

```
itemnamelist = { {Isaac Newton} {James Clark Maxwell} {Albert Einstein} }
```

リストがキーワード 2 個を内容として持つ：

```
position={left bottom}
```

リストが複数の型を内容として持つ（float とキーワード）：

```
position={10 bottom}
```

リストが長方形 1 個を内容として持つ：

```
boxes={{10 20 30 40}}
```

リストが折れ線 2 個を内容として持ち、その中にパーセント値がある：

```
polygons = {{10 20 40 60 90 120}} {12 87 34 98 34% 67% 34% 7%}}
```

よくある落とし穴　オプションリストの文法について、よくある誤りを以下に挙げます。

中括弧は区切りキャラクターではありませんので、以下は誤りです：

```
key1 {value1}key2 {value2}                    誤り！
```

この場合、エラーメッセージ「*Unknown option 'value2'*」が発生します。同様に、以下も区切りキャラクターがありませんので誤りです：

```
key{value}                                    誤り！
key={{value1}{value2}}                        誤り！
```

中括弧は開閉対応している必要があります。以下は誤りです（括弧で囲わない文字列の文法については後述）：

```
key={open brace {}                            誤り！
```

この場合、エラーメッセージ「*Braces aren't balanced in option list 'key={open brace {}'*」が発生します。文字列の中に単独の中括弧があるときは、その直前にバックスラッシュキャラクターを付加する必要があります：

```
key={closing brace \} and open brace \{}      正しい！
```

文字列値の末尾がバックスラッシュの場合は、もしその直後が閉じ中括弧キャラクターならば、直前にもう 1 個バックスラッシュをつける必要があります：

```
key={\value\}                    誤り！
key={\value\\}                   正しい！
```

1.1.2 さまざまな単純データ型

文字列　文字列は、一般にキーワードのために用いられる、プレーンな ASCII 文字列です（EBCDIC プラットフォーム上では EBCDIC 文字列）。文字列の中に空白キャラクターか「=」キャラクターがあるときは、*{ }* でかこむ必要があります：

```
password={ secret string }       （文字列値の中に空白が3個ある）
contents={length=3mm}            （文字列値の中に等号が1個ある）
```

キャラクター *{* か *}* を文字列に入れたいときは、直前にキャラクター \ を付加する必要があります：

```
password={weird\}string}         （文字列値の中に右中括弧が1個ある）
```

要素の閉じ中括弧の直前にバックスラッシュがあるときは、その直前にバックスラッシュキャラクターをつける必要があります：

```
filename={C:\path\name\\}        （文字列の末尾がバックスラッシュ 1個）
```

空文字列は中括弧のペアで作成できます：

```
{}
```

内容文字列・ハイパーテキスト文字列・名前文字列：これらは各種形式の Unicode 内容を持つことができます。オプション *escapesequence* が設定されていれば、シングルバイトをエスケープシーケンスで表現できます。これらの文字列型と、文字列オプションでのエンコーディングの選択について詳しくは、***PDFlib* チュートリアル**を参照してください。

C・Perl・Ruby で *stringformat=legacy* のとき：オプションリストの先頭が [EBCDIC-]UTF-8 BOM のときは、そのオプションリストの各内容・ハイパーテキスト・名前文字列は、[EBCDIC-]UTF-8 文字列として解釈されます。

括弧で囲われていない文字列　以下の場合には、オプションリスト値内のキャラクター本体が、オプションリスト文法キャラクターと衝突する可能性があります：

- パスワードかファイル名が、開閉対応しない中括弧や、バックスラッシュや、その他特殊キャラクターを含んでいる場合
- JavaScript コードをオプション内で与えると、中括弧 { } の使用によって、問題の原因となります

任意のテキストまたはバイナリーのデータを、オプションリスト文法の構成要素と抵触しないように与えるための単純なしくみを提供するため、括弧で囲わないオプション値を、以下の文法変種の中で、長さの指定子とともに与えることができます：

```
key[n]=value
key[n]={value}
```

ここで 10 進値 *n* は、*value* を表すために必要とされる符号単位の数です。たとえば *stringformat=utf8* の言語バインディングではバイトであり、Java か .NET では UTF-16 符号単位です。

文字列値を囲う中括弧はオプショナルではありますが、強く推奨されます。空白やその他の区切りキャラクターで始まる文字列では中括弧は必須です。文字列値の中の中括弧・区切りキャラクター・バックスラッシュは、何ら特殊な解釈をされることなく、文字通りに取られます。

空白と中括弧キャラクターを含む 7 キャラクターのパスワードを指定する例です。文字列全体を中括弧で囲っており、この中括弧はオプション値の一部分ではありません：

```
password[7]={ ab}c d}
```

オプションリストがネストされている場合に、オプション値を長さカウントとともに与えたときは、その中身のオプションリストも長さカウントを与える必要があります。例：

```
fitannotation[34]={contents[19]={this is a brace '}'}}
```

Unichar　Unichar は 1 個の Unicode 値であり、以下の各種文法が使えます：10 進値≧ 10（例：173）、16 進値の前に *x*・*X*・*ox*・*oX*・U+ のいずれかをつけたもの（*xAD*・*oxAD*・*U+ooAD*）、数値参照・文字参照・グリフ名参照から「&」・「;」修飾を除いたもの（*shy*・*#*xAD・#173）。あるいは、リテラルなキャラクターを与えることもできます。Unichar は、範囲 0 ～ 65535（0 ～ 0xFFFF）でなければなりません。例：

```
replacementchar=?                （リテラル）
replacementchar=63               （10進）
replacementchar=x3F              （16進）
replacementchar=0x3F             （16進）
replacementchar=U+003F           （Unicode記法）
replacementchar=euro             （HTML文字参照）
replacementchar=.question        （標準のグリフ名参照）
replacementchar=.marina          （フォント独自のグリフ名参照）
```

数字 1 文字はリテラルに扱われ、10 進 Unicode 値としては扱われません：

```
replacementchar=3                （U+0033 THREE。U+0003ではありません！）
```

Unichar は、16 進で 0 ～ 0x10FFFF（10 進で 0 ～ 1114111）の範囲内になければなりません。ただし、いくつかのオプションでは範囲 0 ～ 0xFFFF（0 ～ 65535）に制約されます。これはそれぞれのオプションの説明に注記してあります。

Unicode 範囲　Unicode 範囲は、連続的な範囲の Unicode キャラクター群を、その範囲の先頭キャラクターと末尾キャラクターとで指定したものです。Unicode 範囲の先頭値と末尾値とは、空白を入れずに負号「-」で区切る必要があります。例：

```
forcechars={U+03AC-U+03CE}
```

論理値　論理値は値 *true* か *false* を持ちます。論理値オプションの値が省略されたときは、値 *true* であると見なされます。*name=false* のかわりに短縮記法 *noname* を用いることもできます：

```
embedding          （embedding=trueと同等）
noembedding        （embedding=falseと同等）
```

キーワード　キーワード型のオプションは、固定キーワードの定義済みリストから 1 つを持つことができます。例：

```
blendmode=overlay
```

オプションのなかには、数値かキーワードのいずれかの値を持つものがあります。

数値　オプションリストは、いくつかの数値型に対応しています。
整数型は、10 進または 16 進の整数を持つことができます。x・X・0x・0X のいずれかで始まる正の整数は 16 進値を表します：

```
-12345
0
0xFF
```

float は、浮動小数点数または整数を持つことができます。浮動小数点値の小数点としてはピリオドとカンマを用いることができます。指数記法にも対応しています。以下の値はすべて同等です：

```
size = -123.45
size = -123,45
size = -1.2345E2
size = -1.2345e+2
```

パーセント値は、数値の直後に % キャラクターを 1 個つけた数値です。オプションによっては負のパーセント値が許されます：

```
leading=120%
topoffset=-20.5%
```

ハンドル　ハンドルは、フォント・画像・ICC プロファイル・アクションといったさまざまな種類のオブジェクトを特定します。技術的にはこれらは、API メソッドによって以前に返された整数値です。たとえば、画像ハンドルは *PDF_load_image()* によって返されます。ハンドルはつねに不透明な値として取り扱う必要があり、アプリケーション側で直接変更したり作成したりしてはいけません（API メソッドによって返されたハンドルを用いる必要があります）。ハンドルはつねに、それぞれのオブジェクトの種類に対して有効でなければなりません。たとえば、どちらのハンドルも整数型だからといって、画像ハンドルを受け付けるオプションにグラフィックハンドルを与えてはいけません。

1.1.3 文字サイズ・アクションデータ型

文字サイズ　文字サイズは、いくつかの方式で定義することができ、それにより、テキストのサイズを絶対値で指定したり、何らかの外部の実体に対する相対値で指定したり、何らかのフォントプロパティに対する相対値で指定したりすることができます。通常、文字サイズは 0 以外でなければなりませんが、ただしオプションの解説で特記ある場合はこの限りではありません。

多くの場合、文字サイズは float 値 1 個を内容として持ち、これはユーザー座標系における単位に対する比率を表します：

```
fontsize=12
```

または、パーセント値を内容として持つこともでき、これが何に対するパーセント値であるかはコンテキストによって異なります（例：*PDF_fit_textline()* では、はめ込み枠に対する幅）：

```
fontsize=8%
```

あるいは文字サイズは、オプションリストとして指定することもでき、これはキーワードと数値を内容として持つ必要があります。ここでキーワードは、求めるフォントメトリックを表 1.1 に従って記述し、数値は、求めるサイズです。選ばれたテキストメトリックが、与えられた値に一致するよう、PDFlib が適切な文字サイズを算出します：

```
fontsize={capheight 5}
```

表 1.1 文字サイズ型のオプションのサブオプション

オプション	説明
ascender	数値は、アセンダーの高さとして解釈されます。
bodyheight	数値は、ベースラインの間隔の最小値として解釈されます。すなわち、この値を行送りとして用いたときは、隣り合う行のディセンダーとアセンダーがちょうどくっつく形になります。これは、キーワードが与えられないときのデフォルトの動作です。
capheight	数値は、大文字の高さとして解釈されます。
xheight	数値は、小文字の高さとして解釈されます。

アクションリスト　アクションリストは、1 個ないし複数のアクションを指定します。リスト内の各項目は、イベントキーワード（トリガー）1 個と、アクションハンドルのリスト 1 個とから成ります。このアクションハンドルは、*PDF_create_action()* で作成しておく必要があります。アクションは、リスト内に記述された順に実行されます。許されるイベント（例：*docopen*）とアクションの種類（例：JavaScript）は、オプションごとにそのつど記します。

リストが、トリガー 1 個とアクション 3 個を内容として持つ：

```
action={ activate={ 0 1 2 } }
```

リストがトリガー 3 個を持ち、それぞれがアクションを 1 個ずつ持つ：

```
action={ keystroke=0 format=1 validate=2 }
```

1.1.4 色データ型

色空間の概要　パスとテキストキャラクターを塗る色と描線する色を指定できます。色はいくつかの色空間で指定できます（さまざまな色空間と値に関する完全な記述については PDFlib チュートリアルを参照してください）。以下の箇条書きの各項目の頭に、それぞれの色オプションに対応する色空間キーワードを記しています：

- *gray*：0= 黒と 1= 白の間のグレー値。
- *rgb*：RGB の 3 値、すなわち、赤・緑・青を指定する 0 から 1 までの 3 つの値。*(0, 0, 0)*= 黒、*(1, 1, 1)*= 白。広く用いられている範囲 0 ～ 255 の RGB カラー値は、PDFlib が求める範囲 0 ～ 1 へスケールするために 255 で割る必要があります。
  数値の RGB 値のかわりに、RGB 色をその HTML 名または 16 進値を通じて指定することもできます。

- *cmyk*：CMYK の 4 値、すなわち、シアン・マゼンタ・イエロー・黒の値を表す、0= 色なし、1= フルカラーの間の値。*(0, 0, 0, 0)*= 白、*(0, 0, 0, 1)*= 黒。これは RGB 指定とは異なることに留意してください。
- *iccbased*（*PDF_setcolor( )* では不可）・*iccbasedgray/rgb/cmyk*：ICC ベースカラーは ICC プロファイルに基づきます。
- *spotname*：定義済みスポットカラーの名前と濃度値を、0= 色なしから 1= 最高濃度までの範囲で表したもの。
  あるいは、カスタムスポットカラーの名前と、濃度値と、代替表現を上記の他の色空間のいずれか一つで。
- *spot*：定義済みの、または、*PDF_makespotcolor( )* を用いて作成されたカスタムのスポットカラーへのハンドルと濃度値。
- *devicen*：*PDF_create_devicen( )* を用いて作成された DeviceN 色空間へのハンドルと、名前付きインキ群に対する N 個の濃度値。濃度値の範囲は 0 ＝色なしから 1 ＝最大濃度まで。
- *lab*：CIE L*a*b* 黒空間の中のデバイス独立カラーを受け付けます。色は、範囲 0 ～ 100 の輝度値 1 個と、範囲 −128 ～ 127 の 2 個のカラー値 ***a*** と ***b*** で指定されます。***a*** コンポーネントの範囲は緑から赤 / マゼンタまで（負値が緑を、正値がマゼンタを示す）、***b*** コンポーネントの範囲は青から黄まで（負値が青を、正値が黄を示す）です。
- *pattern*：パターンハンドルによって識別されるシェーディングパターン。シェーディングパターンは、複数の色の間のなめらかな遷移を与えるもので、これを作成するには、*PDF_shading( )* を用いて作成したシェーディングハンドルに基づいて、*PDF_shading_pattern( )* を用います。
- *pattern*：パターンハンドルによって識別されるタイリングパターン。タイリングパターンは、任意のテキスト・ベクトルグラフィック群・画像群を含み、塗るべき領域全体に繰り返しタイルされます。タイリングパターンを作成するには *PDF_begin_pattern_ext( )* を用います。

描線・塗り操作のデフォルトカラーは黒です。このデフォルトカラーの色空間は、PDF/X・PDF/A のカラー必要条件に合致するよう自動的に選択されます。

色オプション　色オプションは、3 種類の形式で定義することができます：RGB カラー名、16 進 RGB 値、任意の色空間の色のためのフレキシブルなオプションリストのいずれかを用います。

第一の形式では、SVG 1.1 のすべての有効なカラー名を直接与えて、RGB カラーを、あるいは sRGB ICC プロファイルが選択されている場合には sRGB カラーを、指定することができます。例：

```
strokecolor=pink
```

カラー名は、大文字 / 小文字を区別します。有効な RGB カラー名の一覧は、以下の場所で得られます：

www.w3.org/TR/SVG11/types.html#ColorKeywords

第二の形式では、ハッシュ「#」キャラクターの直後に、任意の 3 個の 16 進数ペア 00 ～ FF を与えて、RGB カラー値を指定することができます。例：

```
strokecolor=#FFC0CB
```

第三の形式は、色空間とカラー値を指定する色オプションリストです。色オプションリストは、色空間キーワード 1 個と、その色空間によって決まる個数の float 値のリスト 1 個を内容として持ちます。表 1.2 に、色空間キーワードを例とともに示します。各メソッドの説明で記しますが、オプションリストによっては、色空間キーワード群の部分集合のみに対応しているものもあります。

表 1.2 オプションリスト内の色データ型のキーワード

キーワード	後続する値	例
gray	グレースケール色空間の float 値 1 個	{ gray 0.5 }
rgb	RGB 色空間の float 値 3 個	{ rgb 1 0 0 }
（キーワードなし）	HTML カラー名または RGB 色の 16 進値	pink #FFC0CB
cmyk	CMYK 色空間の float 値 4 個	{ cmyk 0 1 0 0 }
lab	Lab 色空間の float 値 3 個	{ lab 100 50 30 }
spot	PDF_makespotcolor( ) を用いて作成されたスポットカラーハンドルの後に、濃度値を指定する float 1 個	{ spot <ハンドル> 0.8 }
spotname	（最大 63 バイト。Unicode キャラクター群ならばもっと少なく、形式・エンコーディングに依存）スポットカラー名と、範囲 0 ～ 1 で濃度値を指定する float 1 個	{ spotname {PANTONE 281 U} 0.5 }
spotname	上記の簡単な形の spotname と同様ですが、カラー値を追加することにより、カスタムスポットカラー（すなわち PDFlib が内部的に知らないスポットカラー名）に対する代替色を指定することができます。複数のオプションで同じカスタムスポットカラー名を定義するときは、すべての定義で整合性をとる（すなわち同じ代替色を定義する）必要があります。	{ spotname {PDFlib Blue} 0.5 { lab 100 50 30 } }
devicen	PDF_create_devicen( ) を用いて作成された DeviceN 色空間ハンドルの後に、範囲 0 ～ 1 でインキ群の濃度値を指定した N 個の float 値。	{ devicen <ハンドル> 0.8 0.9 } { devicen <ハンドル> 0 0 0.1 0.2}
iccbased	ICC プロファイルハンドルまたはキーワード srgb と、ICC プロファイルの種別（グレーか RGB か CMYK か）に応じて 1 個か 3 個か 4 個のカラー値。この srgb キーワードは文書スコープ内では使用してはいけません。	{ iccbased <ハンドル> 0.5 } { iccbased <ハンドル> 0 0 0.75 } { iccbased srgb 0 0 0.75 } { iccbased <ハンドル> 0 0 0.3 1 }
iccbasedgray	オプション iccprofilegray を用いて選択された ICC プロファイルを参照する float 値 1 個	{ iccbasedgray 0.5 }
iccbasedrgb	オプション iccprofilergb を用いて選択された ICC プロファイルを参照する float 値 3 個	{ iccbasedrgb 1 0 0 }
iccbasedcmyk	オプション iccprofilecmyk を用いて選択された ICC プロファイルを参照する float 値 4 個	{ iccbasedgray 0 1 0 0 }
pattern	PDF_shading_pattern( ) を用いて作成されたシェーディングパターンハンドル	{ pattern <ハンドル> }
pattern	PDF_begin_pattern_ext( ) を用いて作成されたタイリングパターンハンドル	{ pattern <ハンドル> }
none	色がないことを示します	none

1.1.5 図形データ型

線　線は、float 値 4 個のリストであり、これらは、線分の始点と終点の *x* 座標と *y* 座標を指定します。これらの座標を解釈するための座標系（デフォルト座標系またはユーザー座標系）は、オプションによって異なりますので、個別に示します：

```
line = {10 40 130 90}
```

折れ線　折れ線は、1 個ないし複数の点を内容として持つリストです。それぞれの点は、float 値のペアによって記述されます。リスト内のそれぞれのペアは、点 1 個の *x* 座標と *y* 座標を指定します。これらの点が線分でつながれます。これらの座標を解釈するための座標系（デフォルト座標系またはユーザー座標系）は、オプションによって異なりますので、個別に示します：

```
polyline = {10 20 30 40 50 60}
```

以下のオプションリストは同等です：

```
polyline = {10 20 30r 40r 50r 60r}
polyline = {10 20 40 60 90 120}
```

四辺形は、特殊な種類の折れ線です。これは長方形が回転したものであり、ちょうど 4 個の点を指定する必要があります。

もう 1 つの特殊な種類は多角形です。これは、自動的に線分によって閉じられる折れ線です。

長方形　長方形は、float 値 4 個のリストであり、これらは、長方形の左下隅と右上隅の *x* 座標と *y* 座標を指定します。これらの座標を解釈するための座標系（デフォルト座標系またはユーザー座標系）は、オプションによって異なりますので、個別に示します。オプションのなかには、パーセント値を受け付けるものもあり、これが何に対するパーセント値であるかはコンテキストによって異なります（例：テキストフローのはめ込み枠）。例：

```
cropbox={ 0 0 500 600 }
box={40% 30% 50% 70%}
```

相対座標　早退座標に対応しているオプションもあります。数値の直後に接尾辞 *r* を付加することにより、相対座標を指定することもできます。相対座標が何を基準とするか（たとえばブロックの長方形）はオプションリストの説明で記します。座標リストの中では、相対座標は、直前の *x* 座標か *y* 座標を基準とします。相対座標がリストの先頭にあるときは、原点が基準、すなわち絶対座標となります。

次の 2 つのオプションは同等です：

```
box={12 34 56r 78r}
box={12 34 68 112}
```

円　円は、float 値 4 個のリストとして指定され、1 番目のペアは、中心の *x* 座標と *y* 座標を指定し、2 番目のペアは、円周上の任意の点の *x* 座標と *y* 座標を指定します。これらの座標を解釈するための座標系（デフォルト座標系またはユーザー座標系）は、オプションによって異なりますので、個別に示します：

```
circle={200 325 200 200}
```

曲線リスト　曲線リストは、連結された 2 本以上の 3 次のベジエ曲線分から成ります。1 本のベジエ曲線は、4 個の制御点によって指定されます。1 個目の制御点は曲線の始点であり、4 個目の点は終点です。2 個目の点と 3 個目の点は曲線の形状を制御します。曲線リストでは、1 つの曲線分の最後の点が、次の曲線分の 1 個目の点となります：

```
curve={200 700 240 600 80 580 400 660 400 660 440 620}
```

曲線の描画後は、最後の制御点が新しいカレント点となります。

1.2 メソッドの各種スコープ

PDFlib のアプリケーションは、わかりやすい一定の構造規則に従う必要があります。たとえば、文書を開始する前に終了することなど当然できません。PDFlib は、メソッドが正しい順序で呼び出されるよう、厳格なスコープ機構で強制しています。各スコープの定義を表 1.3 に示します。すべての API メソッドの解説に、各メソッドの許されるスコープを示します。その許されるスコープの外でメソッドを呼ぶと、例外が発生します。*PDF_get_option()* の *scope* キーワードを使うと、カレントのスコープを知ることができます。

表 1.3 メソッドの各種スコープの定義

スコープ名	定義
パス	PDF_moveto( ) ・ PDF_circle( ) ・ PDF_arc( ) ・ PDF_arcn( ) ・ PDF_rect( ) ・ PDF_ellipse( ) ・ PDF_elliptical_arc( ) のいずれかで開始。 165 ページ「7.5 描画とクリッピング」のいずれかのメソッドで終了
ページ	PDF_begin_page_ext( ) と PDF_end_page_ext( ) の間、ただしパススコープの外
テンプレート	PDF_begin_template_ext( ) と PDF_end_template( ) の間、ただしパススコープの外
パターン	PDF_begin_pattern_ext( ) と PDF_end_pattern( ) の間、ただしパススコープの外
フォント	PDF_begin_font( ) と PDF_end_font( ) の間、ただしグリフスコープの外
グリフ	PDF_begin_glyph_ext( ) と PDF_end_glyph( ) の間、ただしパススコープの外
文書	PDF_begin_document( ) と PDF_end_document( ) の間、ただしページ・パス・グリフ・テンプレート・パターン・フォントスコープの外
オブジェクト	PDFlib オブジェクトが存在している間、ただし文書スコープの外。C・RPG 言語バインディングの場合は、PDF_new( ) と PDF_delete( ) の間、ただし文書スコープの外

1.3 ログ記録

ログ記録機能を使うと、API の呼び出しをトレースすることができます。ログファイルの内容は、デバッグ目的にも有用ですし、PDFlib GmbH のサポートから求められることもあります。ログ記録のオプションは、以下の方法で与えることができます：

- *PDF_set_option()* のグローバル *logging* オプションに対するオプションリストとして。例：

```
p.set_option("logging={filename=tracelog remove}")
```

- *PDFLIBLOGGING* という環境変数で。こうすると、いちばん初めに何らかの API メソッドを呼び出した時からログ出力が始められます。

表 1.4 logging オプションのサブオプション

オプション	説明
（空リスト）	ログ出力を有効にします
disable	（論理値）ログ記録の出力を無効にします
enable	（論理値）ログ出力の出力を有効にします
filename	（文字列）ログファイルの名前。特殊名として stdout と stderr も認識されます。CICS ではこのオプションは無視され、ログ出力はつねに stderr へ書き出されます。既存の内容があるときは、出力はそれに追加されます．デフォルト： pdflog　z/OS の場合 PDFlib.log　macOS・IBM System i の場合 \PDFlib.log　Windows の場合 /tmp/PDFlib.log　それ以外の全システムの場合 あるいはログファイルの名前は、環境変数 PDFLIBLOGFILE で与えることもできます。
flush	（論理値）true にすると、出力が終わるたびにログファイルが閉じられ、次の出力でまた開かれるので、出力が確実に 放出されます。これは、ログファイルの中断箇所を見てプログラムのクラッシュを追いたいときに有用ですが、ただし処理はかなり遅くなります。false にすると、ログファイルは 1 回だけ開かれます。デフォルト：false
includepid ***includetid*** ***includeoid***	（論理値。MVS では不可）プロセス ID かスレッド ID かオブジェクト ID をログファイル名に含めます。複数のプロセスが同一のログファイルを使用するときはこれを有効化するべきです。デフォルト：false
remove	（論理値）true にすると、既存のログファイルは、新しい出力が書き出される前に削除されます。デフォルト：false
removeon-success	（論理値）生成されたログファイルを、例外が発生した場合を除き、PDF_delete( ) で削除します。これは、時たま起こる問題や散発的にのみ起こる問題を分析するために有用でしょう。このオプションを、includepid/includetid/includeoid と適宜組み合わせて用いることを推奨します。
stringlimit	（整数）文字列あたりの文字数の制限値か、または 0 で無制限。デフォルト：0

表 1.4 logging オプションのサブオプション

オプション	説明
classes	（オプションリスト）整数型のオプションを入れたオプションリスト。ここで各オプションはログ記録種別を記述し、その値は粒度レベルを記述します。レベル 0 ならそのログ記録種別は無効になり、正の値ならその種別は有効になります。レベルを上げるほど出力は詳しくなっていきます。以下のオプションが用意されています（デフォルト：{api=1 warning=1}）： ***api*** すべての API 呼び出しを、そのメソッド引数と戻り値とともにログ記録します。api=2 にすると、すべての API トレース行の頭にタイムスタンプが生成されるとともに、廃止済のメソッド・オプションはその旨が示されます。 ***filesearch*** SearchPath または PVF によるファイル検索に関連したすべての試みをログ記録します。 ***resource*** Windows レジストリーと UPR 定義によるすべてのリソース検索の試みを、そのリソース検索の結果とともにログ記録します。 ***tagging*** 構造エレメント（タグ付け）操作 ***user*** userlog オプションで与えている、ユーザー指定のログ記録出力。 ***warning*** すべての PDFlib 警告をログ記録します。警告とは、無視または内部対応できるエラー状況です。warning=2 にすると、例外を発生させずにメッセージテキストを中継させて PDF_get_errmsg( ) で取得させるメソッドからのメッセージと、ファイルを開こうとして失敗したすべての試みの原因もログ記録されます。

1.4 例外処理

この節に関連するオプションを表 1.5 に示します。これらのオプションは多くの関数で用いることができ、それぞれのオプションリストの開設でその旨示されています。これは、*PDF_set_option( )* に対するグローバルオプションとしても与えることができます (25 ページ「2.1 グローバルオプション」参照)。

表 1.5 PDF_set_option( ) の例外関連オプション

オプション	説明
errorpolicy	（キーワード）値を返す API メソッドのエラー動作を制御します。errorpolicy グローバルオプションは、さまざまなメソッドの errorpolicy オプションによってオーバーライドすることができ、このオプションに対するデフォルトとしてはたらきます。使えるキーワード（デフォルト：return）： ***return*** 処理エラーが起きた場合にメソッドは返ります。PDF_load_image( ) 等、エラーコードを返せるメソッドは、-1（PHP では 0）を返します。PDF_fit_table( ) 等、結果文字列を返すメソッドは、文字列 _error を返します。アプリケーションの開発者は、戻り値が -1（PHP では 0）または _error でないかを検査してエラー状況を検出する必要があります。エラーが起きたときは、問題の詳しい内容は PDF_get_errmsg( ) で取得できます。新規のアプリケーションにはこの設定を推奨します。 たとえ errorpolicy=return を指定していても、次の場合には例外が発生します：オプションリスト内に文法エラーが見つかった場合。または、PDF 出力が書き出せない場合。 ***exception*** エラーが起きるとメソッドは例外を発生させます。この例外は、クライアントコード内で捕捉する必要があります。その時点までに生成された作りかけの PDF 出力は使えなくなりますので、破棄する必要があります（これは removefragments 文書オプションを用いて自動化することも可能です）。

C++ Java C# ***int get_errnum( )***

Perl PHP ***int get_errnum( )***

C ***int PDF_get_errnum(PDF *p)***

最後に発生した例外か、またはメソッド呼び出し失敗の原因の番号を取得します。

戻り値 もっとも最近のエラー条件のエラーコード。

スコープ PDFlib で例外が発生してから、PDFlib オブジェクトが死ぬまでの間。あるいは、メソッドがエラーコード –1（PHP では 0）を返してから、この節で示しているものを除く任意の関数を呼び出すまでの間に、このメソッドを呼び出すこともできます。

バインディング C++・Java・Objective C・.NET・PHP では、このメソッドは、*PDFlibException* オブジェクトの *get_errnum( )* としても利用できます。

C++ Java C# ***String get_errmsg( )***

Perl PHP ***string get_errmsg( )***

C ***const char *PDF_get_errmsg(PDF *p)***

最後に発生した例外か、またはメソッド呼び出し失敗の原因のテキストを取得します。

戻り値 もっとも最近のエラー条件の記述されたテキスト。

スコープ PDFlib で例外が発生してから、PDFlib オブジェクトが死ぬまでの間。あるいは、メソッドがエラーコード –1（PHP では 0）を返してから、この節で示しているものを除く任意の関数を呼び出すまでの間に、このメソッドを呼び出すこともできます。

バインディング C++・Java・Objective C・.NET・PHP では、このメソッドは、*PDFlibException* オブジェクトの *get_errmsg( )* としても利用できます。

C++ Java C# ***String get_apiname( )***

Perl PHP ***string get_apiname( )***

C ***const char *PDF_get_apiname(PDF *p)***

最後の例外を発生させたか、または失敗した API メソッドの名前を取得します。

戻り値 例外を発生させた API メソッドの名前か、またはもっとも最近に呼び出されて失敗しエラーコードを返したメソッドの名前。

スコープ PDFlib で例外が発生してから、PDFlib オブジェクトが死ぬまでの間。あるいは、メソッドがエラーコード –1（PHP では 0）を返してから、この節で示しているものを除く任意の関数を呼び出すまでの間に、このメソッドを呼び出すこともできます。

バインディング C++・Java・Objective C・.NET・PHP では、このメソッドは、*PDFlibException* オブジェクトの *get_apiname( )* としても利用できます。

C++ ***void *get_opaque()***

C ***void *PDF_get_opaque(PDF *p)***

PDFlib 内に格納されている不透明なアプリケーションポインターを取り出します。

戻り値 *PDF_new2()* を呼び出した時に与えておいた、PDFlib 内に格納されている不透明なアプリケーションポインター。

詳細 PDFlib は、この不透明ポインターを一切さわらずに、そのままでクライアントに与えます。これは、マルチスレッドのアプリケーションでプライベートなスレッド独自データを PDFlib オブジェクト内に格納するために使えるでしょう。特に、スレッド独自の例外処理に有用です。

スコープ 任意

バインディング C・C++ バインディングでのみ得られます。

1.5 Unicode 変換

C++ ***string convert_to_unicode(string inputformat, string input, string optlist)***
Java ***string convert_to_unicode(string inputformat, byte[ ] input, string optlist)***
Perl PHP ***string convert_to_unicode(string inputformat, string input, string optlist)***
C ***const char *PDF_convert_to_unicode(PDF *p,***
const char *inputformat, const char *input, int inputlen, int *outputlen, const char *optlist)

任意のエンコーディングの文字列を、さまざまな形式の Unicode 文字列へ変換します。

inputformat 入力文字列の解釈を指定する Unicode テキスト形式またはエンコーディング名：

- Unicode テキスト形式：***utf8*・*ebcdicutf8***（EBCDIC プラットフォームの場合）・***utf16*・*utf16le*・*utf16be*・*utf32***
- ***font*** オプションが指定されている場合のみ：***builtin*・*glyphid***
- すべての構成されている 8 ビットエンコーディングと、ホストシステム上で利用可能なエンコーディング
- キーワード ***auto*** は以下の動作を指定します：もし入力文字列が UTF-8 か UTF-16 の BOM を含んでいるなら、それを用いて適切な形式を決定し、そうでない場合にはカレントのシステムコードページであると見なされます。

input Unicode へ変換したい文字列。

inputlen （C・RPG 言語バインディングのみ）入力文字列の長さをバイト単位で表したもの。***inputlen=0*** の場合には、ヌル終端文字列を与える必要があります。

outputlen （C・RPG 言語バインディングのみ）返される文字列の長さが格納されるメモリー位置への C スタイルのポインター。

optlist 入力解釈と Unicode 変換のためのオプション群を指定したオプションリスト：

- 表 4.5 に従ったテキストフィルターオプション：***charref*・*escapesequence***
- 表 1.6 に従った Unicode 変換オプション：
  bom*・*errorpolicy*・*font*・*inflate*・*outputformat

戻り値 入力文字列から、指定された引数群とオプション群に従って作成された Unicode 文字列。入力文字列が、指定された入力形式に準拠していないときは（たとえば無効な UTF-8 文字列）、***errorpolicy=return*** の場合には空の出力文字列が返され、***errorpolicy=exception*** の場合には例外が発生します。

詳細 このメソッドは、汎用の Unicode 文字列変換のために有用でしょう。これは、然るべき Unicode 変換機能を有さない環境で作業をするユーザーの便宜のために提供されています。

スコープ 任意

バインディング C 言語バインディング：返される文字列は、最大 10 項目を持つリングバッファに格納されます。もし 10 個より多い文字列が変換されると、このバッファは再利用されますので、クライアントは、10 個を超える文字列を並行して利用したい場合には、この文字列群を複製する必要があります。たとえば、このメソッドへの呼び出しが 10 回までであれば、それをステートメントに対する引数として用いて差し支えありません。なぜなら、10 個

を超える文字列が同時に使用されないならば、返される文字列は独立であることが保証されるからです。

C++ 言語バインディング：引数 *inputformat* と *optlist* は、構成された文字列型を持ち、一方、*input* と戻りデータは型 *std::string* を持ちます。

表 1.6 PDF_convert_to_unicode( ) のオプション

オプション	説明
bom	（キーワード。outputformat=utf32 では無視されます。.NET・Java・Objective-C・Python では none のみ許されます）出力文字列にバイト順序マーク（BOM）を付加するにあたってのポリシー。使えるキーワード（デフォルト：none）： ***add*** BOM を付加。 ***keep*** 入力文字列に BOM があるなら BOM を付加。 ***none*** BOM を付加しない。 ***optimize*** outputformat=utf8 か ebcdicutf8 かつ出力文字列が範囲＜U+007F 内のキャラクターのみを内容としている場合を除いて BOM を付加。
errorpolicy	（キーワード）変換エラーの際の動作（デフォルト：errorpolicy グローバルオプションの値。表 1.5 参照）： ***return*** 文字参照が解決できない場合、または指定されたフォント内にコードかグリフ ID が存在しない場合に、代替キャラクターが用いられます。変換エラーの場合には空文字列が返されます。 ***exception*** 変換エラーの場合には例外が発生します。
font	（フォントハンドル。inputformat=builtin・glyphid の場合には必須）指定されたフォントに応じて、フォント固有の変換を適用します。
inflate	（論理値。inputformat=utf8 の場合のみ）true にすると、無効な UTF-8 入力文字列は例外を発生させず、指定された出力形式のインフレートされたバイト文字列が生成されます。このインフレートされた文字列は、その入力文字列内のバイト群の ASCII 解釈に対応する Unicode キャラクター群を内容とします。これはデバッグのために有用でしょう。デフォルト：false
output-format	（キーワード）生成される文字列の Unicode テキスト形式：utf8・ebcdicutf8（EBCDIC プラットフォームの場合）・utf16・utf16le・utf16be・utf32。空文字列は utf16 と等価です。デフォルト：Default: utf16 出力形式は、Unicode 対応言語バインディングでは強制的に utf16 になり、また、stringformat=utf8 なら utf8 になります。 IBM System i・IBM Z：stringformat=ebcdicutf8 なら出力形式は強制的に ebcdicutf8 になります。 RPG 言語バインディング：以下の出力形式のみが許されます：ebcdicutf8・utf8・utf16・utf32。

2 汎用メソッド

この章の API メソッド：

2.1 グローバルオプション

PDFlib は、ライブラリーと PDF 出力の書式を制御するためのさまざまなグローバルオプションを提供しています。これらのオプションの設定は、PDFlib オブジェクトが存在している間ずっと、もしくはクライアントが明示的に設定を変更するまで保持されます。

C++ Java C# ***void set_option(String optlist)***

Perl PHP ***set_option(string optlist)***

C ***void PDF_set_option(PDF *p, const char *optlist)***

---

1 個ないし複数のグローバルオプションを設定します。

optlist 表 2.1 に従ったグローバルオプション群を指定するオプションリスト。以下のオプションを用いることができます：

- 表 2.1 に従った、リソース処理とリソースカテゴリーのためのオプション：
  CMap・*Encoding*・*enumeratefonts*・*FontnameAlias*・*FontOutline*・*HostFont*・*ICCProfile*・*resourcefile*・*saveresources*・*searchpath*
- 表 2.1 に従った、ファイル処理とライセンシングのためのオプション：
  avoiddemostamp・*filenamehandling*・*license*・*licensefile*
- 表 2.1 に従ったテキストフィルターオプション：*charref*・*escapesequence*・*glyphcheck*
- 表 2.1 に従った、インタラクティブ要素のためのオプション：*usercoordinates*
- 表 2.1 に従ったその他のオプション：
  asciifile・*autospace*・*compress*・*kerning*・*logging*・*network*・*shutdownstrategy*・*usehostfonts*・*userlog*
- エラー処理のためのオプション：*errorpolicy*（表 1.5 参照）
- 表 2.1・表 8.2 に従った、色処理のためのオプション：
  iccprofilecmyk・*iccprofilegray*・*iccprofilergb*

- C 言語バインディングの場合、および、Perl・PHP・Ruby で *stringformat=legacy* の場合には、表 2.2 に従ったエンコーディング関連の以下のオプションも使用できます：
  hypertextencoding・*hypertextformat*・*stringformat*・*usehypertextencoding*・*textformat*

詳細 リソースカテゴリーオプション群を除き、新しい値は、以前に設定されたオプション値をオーバーライドします。

以下のオプションは、同名のテキストオプションのためのデフォルト値を与えます（表 4.5・表 4.6 参照）：
charref・*escapesequence*・*glyphcheck*・*kerning*・*textformat*

と同時に、これらのオプションは、カレントのテキストステートにおける同名のオプションを変更します。望まない副作用を避けるため、コンテキスト文字列のためのオプションを *PDF_set_text_option()* でのみ設定することを推奨します。

スコープ 任意、ただしいくつかのオプションでは制約されたスコープが適用されます。

表 2.1 PDF_set_option( ) のグローバルオプション

オプション	定義
asciifile	（論理値。IBM System i・IBM Z でのみ対応）テキストファイルを ASCII エンコーディングと見なします。デフォルト：IBM System i では true、IBM Z では false
autospace	true にすると、カレントフォントが U+0020 のためのグリフを含んでいる場合には、PDFlib は、各テキスト出力の後に自動的に空白キャラクターを付加します。これはタグ付き PDF を生成するために有用でしょう。空白が付加されることによって、その show 操作の後のカレントテキスト位置が変わることに留意してください。デフォルト：false
avoiddemo-stamp	（論理値）true にすると、有効なライセンスキーが見つからないときに例外が発生します。false にすると、すべてのページ上にデモスタンプが作成されます。このオプションは、PDF_begin_document( ) への最初の呼び出しの前に設定する必要があります。デフォルト：false
charref	（論理値）true にすると、すべての内容・名前・ハイパーテキスト文字列に対して、数値・文字実体参照とグリフ名参照の置き換えが有効になります。文字参照の置き換えを望まない箇所でそれが起こらないようにするには（ファイル名など）、PDF_set_text_option( ) でこのオプションを内容文字列に対してのみ設定することを推奨します。詳しくは PDFlib チュートリアルを参照してください。デフォルト：false
CMap	（名前文字列のペアのリスト。廃止）リソース定義を空白か等号「=」で区切ったキー／値ペアのリスト。
compress	（整数）圧縮レベル。圧縮なし＝ 0 から、1 ＝最高速、等々、最高圧縮＝ 9 まで。このオプションは、パススルーモードで処理される画像データでは効力を持ちません。デフォルト：6。スコープ：オブジェクト以外任意
Encoding	（名前文字列のペアのリスト）空白か等号「=」で区切った、リソース定義に対するキー / 値ペアのリスト（詳しくは PDFlib チュートリアルを参照）。複数回呼び出すと、内部リストに新たな項目群が追加されます。

表 2.1 PDF_set_option( ) のグローバルオプション

オプション	定義
enumerate-fonts	（論理値）true にすると、PDFlib は、SearchPath リソースを通じてアクセスできるすべてのディレクトリ内でフォントファイルを探します。ホストフォント群は処理されませんが、C:/Windows/Fonts などと検索パスにシステムフォントディレクトリを追加することによって含めることも可能です。 フォントの数が多いと、フォントの数え上げには多大な時間を要する場合があります。できたリソースリストを、saveresources オプションを用いてファイルへ保存することも可能です。推奨される戦略は、各文書や PDFlib オブジェクトごとにではなく、フォントの集合が変更された場合にのみリソースリストを生成して保存することです。 各フォントについて、PDFlib はその font-family 名・font-weight 名・font-style 名を決定し、以下の方式に従って API フォント名を合成します： <font-family>[,<font-weight>][,font-style] PDFlib は、この合成フォント名とそのフォントのフルパス名と紐付ける形式 <fontname>=<pathname> の FontOutline リソースを作成します。この API ファイル名に加えて、PDFlib は、そのフォントの PostScript 名が合成名と異なる場合には、この PostScript 名を持つ FontnameAlias リソースも作成します： <PostScript fontname>=<artificial fontname> 結果として、そのフォントは、その合成フォント名か PostScript 名のいずれかによって読み込めるようになります。デフォルト：false
escape-sequence	（論理値）true にすると、すべての内容・名前・ハイパーテキスト文字列内のエスケープシーケンスの置き換えが有効になります。エスケープシーケンスの置き換えを望まない箇所でそれが起こらないようにするには（ファイル名など）、PDF_set_text_option( ) でこのオプションを内容文字列に対してのみ設定することを推奨します。デフォルト：false
filename-handling	（キーワード）ファイル名のエンコーディングを示します。Unicode 非対応言語バインディングにおいて UTF-8 BOM なしでメソッド引数として与えられたファイル名は、このオプションに従って解釈されます。デフォルト：Windows・macOS では unicode、IBM System i では auto、その他では honorlang： ***ascii*** 7 ビット ASCII ***basicebcdic*** コードページ 1047 に従った基本 EBCDIC、ただし Unicode 値≦ U+007E のみ ***basicebcdic_37*** コードページ 0037 に従った基本 EBCDIC、ただし Unicode 値≦ U+007E のみ ***honorlang*** （IBM System i では不可）環境変数 LC_ALL・LC_CTYPE・LANG を解釈してファイル名に適用。LANG で指定されたコードセットが利用可能な場合にはそれがファイル名に適用されます。 ***unicode*** （EBCDIC-）UTF-8 形式の Unicode エンコーディング **すべての有効なエンコーディング名** glyphid・builtin 以外の任意のエンコーディング
Fontname-Alias	（名前文字列のペアのリスト）空白か等号「=」で区切った、リソース定義に対するキー / 値ペアのリスト（詳しくは PDFlib チュートリアルを参照）。複数回呼び出すと、内部リストに新たな項目群が追加されます。
FontOutline	（名前文字列のペアのリスト）空白か等号「=」で区切った、リソース定義に対するキー / 値ペアのリスト（詳しくは PDFlib チュートリアルを参照）。複数回呼び出すと、内部リストに新たな項目群が追加されます。
glyphcheck	（キーワード）説明は表 4.5 を参照してください。PDF_set_text_option( ) でこのオプションを内容文字列に対してのみ設定することを推奨します。詳しくは PDFlib チュートリアルを参照してください。デフォルト：replace

表 2.1 PDF_set_option( ) のグローバルオプション

オプション	定義
HostFont	（名前文字列のペアのリスト）空白か等号「=」で区切った、リソース定義に対するキー / 値ペアのリスト（詳しくは PDFlib チュートリアルを参照）。複数回呼び出すと、内部リストに新たな項目群が追加されます。
ICCProfile	（名前文字列のペアのリスト）空白か等号「=」で区切った、リソース定義に対するキー / 値ペアのリスト（詳しくは PDFlib チュートリアルを参照）。複数回呼び出すと、内部リストに新たな項目群が追加されます。
iccprofilecmyk iccprofilegray iccprofilergb(	（ICC プロファイルハンドル）iccbasedcmyk/gray/rgb 色オプションで用いるための CMYK かグレーか RGB 色空間を指定する ICC プロファイル。デフォルト：ICC 色空間なし
kerning	（論理値）true にすると、readkerning オプションを用いて開かれているフォントに対してカーニングが有効になります。そうでない場合にはカーニングを無効にします。デフォルト：true
license	（文字列）PDFlib か PDFlib+PDI か PPS のライセンスキー（詳しくは PDFlib チュートリアルを参照してください）。キーを設定できるのは、初めて PDF_begin_document( ) を呼び出す前です。ライセンスキーがないためにデモスタンプが生成されてしまう事故を防ぐため、avoiddemostamp オプションを用いてください。
licensefile	（名前文字列）ライセンスキーの入ったファイルの名前（詳しくは PDFlib チュートリアルを参照してください）。ライセンスファイルを設定できるのは、初めて PDF_begin_document( ) を呼び出す前に 1 回だけです。
logging	（オプションリスト）表 1.4 に従ったログ記録オプション群
network	（ネットワークオプションリスト）暗黙的にデータをダウンロードする API メソッドにおけるネットワークアクセスを制御するための、表 2.9 に従ったオプションリスト（PDF_download( ) を用いる明示的なダウンロード実行には影響しません）。ネットワーク設定はすべて、次に PDF_set_option( ) を呼び出す時まで有効です。呼び出しのたびにネットワークオプションはすべて無をを起点に設定されますので（指定していないオプションもデフォルト値になります）、複数回呼び出してもオプションが蓄積するわけではありません。url ネットワークサブオプションは使えません。なぜなら対象ネットワークアドレスは、たとえば SVG グラフィック内で参照されているフォントや画像のリソースなどコンテキストから採られるからです。
resourcefile	（名前文字列）PDFlib UPR リソースファイルの相対ファイル名または絶対ファイル名。リソースファイルはただちに読み込まれます。既存のリソースは保持されますが、再設定されたときには新しい値でオーバーライドされます。
saveresources	（名前文字列）PDFlib UPR リソースファイルの相対または絶対ファイル名。このリソースファイルは、いずれかのリソースへの最初のアクセスの直前に読み込まれます。既存のリソースは保持されます。その値は、再設定されている場合には、新しいものでオーバーライドされます。
searchpath	（名前文字列のリスト）読み取りたいファイルが位置する、1 個ないし複数のディレクトリの相対パス名または絶対パス名。検索パスは複数回設定することができ、その項目群は蓄積されて、設定された順に用いられます（詳しくは PDFlib チュートリアルを参照）。ディレクトリ名が空白キャラクターを含んでいる場合の問題を避けるために、すべてのエントリーに対して二重中括弧を用いることを推奨します。空の文字列リスト（すなわち {{}}）は、デフォルトエントリー群を含めて検索パス項目群をすべて削除します。Windows では、検索パスはレジストリー項目で設定することもできます。デフォルト：プラットフォーム依存。PDFlib チュートリアルを参照

表 2.1 PDF_set_option( ) のグローバルオプション

オプション	定義
shutdown-strategy	（整数）すべての PDFlib オブジェクトに対して一度割り当てられたグローバルリソースを解放する方式。各グローバルリソースはそれぞれ、それが初めて必要とされた時に要求によって初期化されます。このオプションは、1 つのプロセス内のすべての PDFlib オブジェクトに対して、同じ値に設定する必要があります。そうでない場合には動作は未定義です（デフォルト：0）： ***0*** 参照カウンタが、いくつの PDFlib オブジェクトがそのグローバルリソースを使っているかを追跡します。最後の PDFlib オブジェクトが削除さたとき、そのリソースは解放されます。 ***1*** リソース群はプロセスの終了まで保持されます。これはパフォーマンスを若干向上させますが、最後の PDFlib オブジェクトが削除された後により多くのメモリーを必要とします。
user-coordinates	（論理値）false にすると、インタラクティブ要素（注釈など）のための座標群は、デフォルト座標系で表されていると見なされます。そうでないときは、カレントユーザー座標系が用いられます。デフォルト：false
userlog	ログファイルへ複製される文字列
usehostfonts	（論理値）true にすると、フォント検索にホストフォントが含まれます。デフォルト：true

表 2.2 PDF_set_option( ) のオプションのうち、C の場合と、Perl・PHP・Ruby で stringformat=legacy の場合にのみ適用されるもの

オプション	定義
hypertext-encoding	（文字列）ハイパーテキスト文字列のためのエンコーディング。空文字列は unicode と等価です。デフォルト：auto
hypertext-format	（キーワード）関数引数としてのハイパーテキスト文字列のための形式。使えるキーワードは bytes・utf8・ebcdicutf8・utf16・utf16le・utf16be・auto。デフォルト：auto
stringformat	（キーワード）API におけるすべての文字列、すなわち名前文字列・内容文字列・ハイパーテキスト文字列・オプションリストの形式。使えるキーワード（デフォルト：C の場合は legacy、しかし Perl・PHP・Ruby の場合は utf8）： ***ebcdicutf8***（IBM System i・IBM Z でのみ可）すべての文字列とオプションリストは、BOM 有りか無しの EBCDIC-UTF-8 形式と見なされます。 ***legacy*** 名前文字列・内容文字列・ハイパーテキスト文字列・オプションリストは、textformat・hypertextformat・hypertextencoding オプションに従って扱われます。 ***utf8*** （IBM System i・IBM Z では不可）すべての文字列とオプションリストは、BOM 有りか無しの UTF-8 形式と見なされます。オプション textformat・hypertextformat・hypertextencoding は許容されません。テキストフローオプション fixedtextformat は強制的に true になります。C 言語バインディングでは、length 引数に 0 より大きな値を与えられた場合には、関数引数としての名前文字列はなお UTF-16 文字列として解釈されます。
usehypertext-encoding	（論理値）true にすると、hypertextencoding オプションで指定されたエンコーディングが名前文字列に対しても用いられます。false にすると、UTF-8 BOM のない名前文字列のためのエンコーディングは host になります。デフォルト：false
textformat	（キーワード）内容文字列を解釈するために用いられる形式。使えるキーワード：bytes・utf8・ebcdicutf8 （IBM System i・IBM Z のみ）・utf16・utf16le・utf16be・auto。デフォルト：auto

C++ Java C# ***double get_option(String keyword, String optlist)***

Perl PHP ***float get_option(string keyword, string optlist)***

C ***double PDF_get_option(PDF *p, const char *keyword, const char *optlist)***

何らかのオプションかその他の値を取得します。

keyword 取得したいオプションを指定したキーワード。以下のキーワードが使えます：それらの意味については *PDF_set_option( )*・*PDF_set_text_option( )*・*PDF_set_graphics_option( )* の説明を参照してください。対応するオプションが存在しないキーワードについては表 2.3 に説明しています：

- 指定したリソースの *n* 番目のエントリーの文字列番号のためのキーワード。ここで *n* は *resourcenumber* オプションに対応します：
  Encoding・*FontnameAlias*・*FontOutline*・*HostFont*・*ICCProfile*・*searchpath*
- 論理型オプション値のためのキーワード。*true* なら 1 を、*false* なら 0 を返します：
  asciifile・*autospace*・*avoiddemostamp*・*charref*・*decorationabove*・*escapesequence*・*fakebold*・*kerning*・*overline*・*pdi*・*spotcolorlookup*・*strikeout*・*tagged*・*underline*・*usercoordinates*・*usehostfonts*
- 整数・浮動小数点オプション値のためのキーワード：
  charspacing・*compress*・*ctm_a*・ *ctm_b*・*ctm_c*・ *ctm_d*・*ctm_e*・*ctm_f*・*currentx*・*currenty*・*icccomponents*・*flatness*・*font*・*fontsize*・*horizscaling*・*iccprofilecmyk*・*iccprofilegray*・*iccprofilergb*・*italicangle*・*leading*・*linecap*・*linejoin*・*linewidth*・*major*・*minor*・*miterlimit*・*pageheight*・*pagewidth*・*revision*・*scope*・*textrendering*・*textrise*・*textx*・*texty*・*underlineposition*・*underlinewidth*・*wordspacing*
- オプション値に対する文字列番号を返すキーワード。その文字列値が得られないときは -1 を返します：
  cliprule・*errorpolicy*・*filenamehandling*・*fillrule*・*glyphcheck*・*resourcefile*・*scope*
- カレント構造エレメントをクエリーするためのキーワード（タグ付き PDF モードでのみ）：
  activeitemid・*activeitemindex*・*activeitemisinline*・*activeitemkidcount*・*activeitemname*・*activeitemstandardname*
- C 言語の場合と、Perl・PHP・Ruby で *stringformat=legacy* の場合には、以下のエンコーディング関連のキーワードも使えます：
  hypertextencoding・*hypertextformat*・*stringformat*・*usehypertextencoding*・*textformat*

optlist 表 2.4 に従ってオプションを指定したオプションリスト。

戻り値 *keyword* によって要求された何らかのオプションの値。要求されたキーワードに対して値が得られないときは、このメソッドは -1 を返します。要求されたキーワードがテキストを生成する場合には、文字列番号が返され、その対応する文字列は *PDF_get_string( )* を用いて取得する必要があります。

スコープ 任意、ただしいくつかのキーワードでは、制限されたスコープが適用されます。

表 2.3 PDF_get_option( ) のさらなるキーワード

キーワード	説明
activeitemid	（整数）カレントでアクティブな構造アイテムのアイテム ID。これは、PDF_activate_item() か、PDF_begin_item() の parent サブオプションと、tag オプションで用いることができます。ルートエレメントがまだ作成されていない場合には -1 が返されます。スコープ：文書・ページ

表 2.3 PDF_get_option( ) のさらなるキーワード

キーワード	説明
activeitem-index	（整数）カレントでアクティブな構造アイテムの、その親の中での、ゼロベースの番号。これは、index タグオプションで用いることができます。カレントアイテムが擬似エレメントかルートエレメントである場合、またはルートエレメントがまだ作成されていない場合には、-1 が返されます。スコープ：文書・ページ
activeitem-isinline	（整数）カレントでアクティブな構造アイテムが直接エレメントであるなら 1、そうでないなら 0。スコープ：文書・ページ
activeitem-kidcount	（整数）カレントでアクティブな構造エレメントの、この時点までに作成された子エレメントの数（擬似エレメントを数えない）。ルートエレメントがまだ作成されていない場合には -1 が返されます。スコープ：文書・ページ
activeitem-name	カレントでアクティブな構造エレメントか擬似エレメントの種別名に対する文字列番号、あるいはルートエレメントがまだ作成されていない場合には -1。スコープ：文書・ページ
activeitem-standard-name	カレントでアクティブなアイテムがロールマップされている標準構造種別名に対する文字列番号、あるいはルートエレメントがまだ作成されていないか、カレントエレメントがロールマッピングが得られないカスタムエレメントである場合には -1。ロールマップがアクティブでないときは、元の種別名が返されます。スコープ：文書・ページ
ctm_a *ctm_b* *ctm_c* *ctm_d* *ctm_e* *ctm_f*	（float）ベクトルグラフィックに対するカレント変換マトリックス（CTM）の構成要素群。スコープ：ページ・パターン・テンプレート・グリフ・パス
currentx *currenty*	（float）カレント点のそれぞれ x・y 座標（カレント座標系の単位で）。スコープ：ページ・パターン・テンプレート・グリフ・パス
icccomponents	（整数）iccprofile オプションで与えたハンドルによって参照される ICC プロファイル内の色要素の数。
major *minor* *revision*	（整数）PDFlib のそれぞれメジャー・マイナー・リビジョン番号。スコープ：任意・null[1]
pageheight *pagewidth*	（float）カレントページ（MediaBox の寸法）のページサイズ。スコープ：オブジェクト以外任意、ただしページスコープでのみ意味を持ちます。
pdi	（整数）基礎をなすライブラリーをビルドした際に PDI がインクルードされていたなら 1 を返します。これは、PDFlib GmbH によって頒布されている PDFlib・PDFlib+PDI・PPS のバイナリーすべてについて、ライセンスキーによらず真です。そうでない場合には 0 を返します。スコープ：任意・null[1]
scope	（整数）カレントスコープの名前に対する文字列番号（表 1.3 参照）
textx *texty*	（float）カレントテキスト位置の x・y 座標。スコープ：ページ・パターン・テンプレート・グリフ

1. C 言語バインディング：PDF * 引数 NULL か 0 とともに呼び出すことができます。

表 2.4 PDF_get_option( ) のオプション

オプション	説明
textstate	（論理値）true にすると、以下のオプションの値がカレントテキストステートから取得され、そうでない場合にはグローバルオプションから取得されます（デフォルト：false）： charref・escapesequence・glyphcheck・kerning・textformat

表 2.4 PDF_get_option( ) のオプション

オプション	説明
iccprofile	（ICC プロファイルハンドル）icccomponents キーワードで用いるための ICC プロファイル
resource-number	（整数）取得したいリソースの番号。リソースは 1 から数えます。デフォルト：1

C++ Java C# ***String get_string(int idx, String optlist)***

Perl PHP ***string get_string(int idx, string optlist)***

C ***const char *PDF_get_string(PDF *p, int idx, const char *optlist)***

文字列値を取得します。

idx *PDF_get_option()*か*PDF_info_*()* メソッドのいずれか一つによって返される文字列番号、あるいはオプションを与える場合には –1。

optlist 表 2.5 に従ってオプションを指定したオプションリスト。

戻り値 *idx*・*optlist* によって要求された何らかの文字列の値。

スコープ 任意

バインディング C：返された文字列は、次に何らかの API メソッドを呼び出すまで有効です。

表 2.5 PDF_get_string( ) のオプション

オプション	説明
version	（論理値）<major>.<minor>.<revision> 形式に、場合によっては beta・rc などといったさらなる修飾子を末尾付加した、フルな PDFlib バージョン文字列。スコープ：任意・null[1]

1. C 言語バインディング：PDF * 引数 NULL か 0 を用いて呼び出すことができます。

2.2 PDFlib オブジェクトを作成・削除

*C PDF *PDF_new(void)*

新規 PDFlib オブジェクトを作成します。

戻り値 PDFlib オブジェクトのハンドル。以後の PDFlib の呼び出しで使えます。このメソッドは、メモリー不足で成功しなかったときは、NULL を返します。

スコープ *null*。このメソッドは**オブジェクト**スコープを開始させます。対応する *PDF_delete( )* と必ずペアにして呼び出す必要があります。

バインディング C：実行時に PDFlib DLL を動的にロードするには *PDF_new_dl( )* を使います。*PDF_new_dl( )* は、すべての API メソッドへのポインターを代入された *PDFlib_api* 構造体へのポインターを返します。DLL がロードできないときや、メジャーまたはマイナーバージョン番号の不一致が検出されたときは、NULL が返されます。

その他の言語バインディング：このメソッドは PDF コンストラクタ内に隠れていて得られません。

*C PDF *PDF_new2(void (*errorhandler)(PDF *p, const char *msg),*
void (*allocproc)(PDF *p, size_t size, const char *caller),*
void (*reallocproc)(PDF *p, void *mem, size_t size, const char *caller),*
*void (*freeproc)(PDF *p, void *mem),*
*void *opaque)*

クライアントから与えるエラー処理ルーチンとメモリー割り当てルーチンを使って、新規 PDFlib オブジェクトを作成します。

errorhandler ユーザー定義のエラー処理関数へのポインター。このエラー処理関数は、*PDF_TRY/PDF_CATCH* セクション内では無視されます。

allocproc ユーザー定義のメモリー割り当て関数へのポインター。

reallocproc ユーザー定義のメモリー再割り当て関数へのポインター。

freeproc ユーザー定義のメモリー解放関数へのポインター。

opaque 何らかのユーザーデータへのポインター。このデータは以後、*PDF_get_opaque( )* で取得できます。

戻り値 PDFlib オブジェクトのハンドル。以後の PDFlib の呼び出しで使えます。この関数は、メモリー不足で成功しなかったときは、C では NULL を返し、C++ では例外を発生させます。

詳細 この関数は、クライアントから与えられたエラー処理ルーチンとメモリー割り当てルーチンを使って、新規 PDFlib オブジェクトを作成します。*PDF_new( )* と異なり、呼び出す側から独自のエラー処理やメモリー割り当てのためのプロシージャを与えることが可能です。このエラー処理関数とメモリープロシージャ群は、どちらかまたは両方が NULL でもかまいません。その場合、PDFlib はデフォルトのルーチンを用います。メモリールーチンは、3 個とも与えるか、3 個とも与えないか、そのどちらかにする必要があります。

スコープ *null*。この関数は**オブジェクト**スコープを開始させます。対応する *PDF_delete( )* と必ずペアにして呼び出す必要があります。

バインディング C++：この関数は PDF コンストラクタで間接的に得られます。すべての引数関数を与える必要は必ずしもありません。デフォルト値 NULL が与えられるからです。与えるメソッドはすべて「C」スタイルの関数でなければならず、C++ のメソッドであってはいけません。

*C void PDF_delete(PDF *p)*

PDFlib オブジェクトを削除し、内部リソースをすべて解放します。

詳細 この関数は、PDFlib オブジェクトを削除し、文書に関連する PDFlib 内部のリソースをすべて解放します。この関数は、1 つの PDFlib オブジェクトにつき 1 回しか呼び出してはいけません。***PDF_delete()*** は、例外が発生したときにも、クリーンアップのために呼び出す必要があります。***PDF_delete()*** 自体は、例外を一切発生させないことが保証されています。PDF 文書を複数生成するときは、文書の終わりごとに ***PDF_delete()*** を呼び出す必要はなく、すべての PDF 文書が完了したときだけでかまいません。

スコープ 任意。この呼び出しの後には、同一の PDFlib オブジェクトを用いての API メソッドへの呼び出しは、この PDF オブジェクトを用いては許されなくなります。

バインディング C：***PDF_new_dl()*** で実行時に PDFlib DLL を動的にロードしていた場合は、***PDF_delete_dl()*** を使って PDFlib オブジェクトを削除してください。

C++：このメソッドは PDF デストラクタで間接的に得られます。

Java：このメソッドはラッパーのコードで自動的に呼び出されます。ただし、Java のファイナライザの不備を回避する目的で、クライアントのコードから明示的に呼び出すこともできます。

Objective-C：このメソッドは、PDFlib の ***release*** メソッドが呼び出された時に呼び出されます。

Perl・PHP：このメソッドは、PDFlib オブジェクトがスコープ外に出ると自動的に呼び出されます。

2.3 PDFlib 仮想ファイルシステム（PVF）

C++ Java C# ***void create_pvf(string filename, const void *data, size_t size, string optlist)***

Java C# ***void create_pvf(String filename, byte[ ] data, String optlist)***

Perl PHP ***create_pvf(string filename, string data, string optlist)***

C ***void PDF_create_pvf(PDF *p,***
const char *filename, int len, const void *data, size_t size, const char *optlist)

メモリー内で与えたデータから、名前付きの仮想の読み取り専用ファイルを作成します。

filename （名前文字列）仮想ファイルの名前。これは任意の文字列で、以後の PDFlib の呼び出しで仮想ファイルを参照するために使えます。仮想ファイルの名前は、それがディレクトリまたはファイル名の区切りキャラクターとしてスラッシュ「/」キャラクターのみを用いているときは、*SearchPath* 機構に従います。

len （C 言語バインディングのみ）*filename* の長さ（バイト単位）。*len=0* にすると null 終了文字列を与える必要があります。

data 仮想ファイルに入れたいデータへの参照。C・C++ ではこれはメモリー位置へのポインターです。Java ではこれはバイト配列です。Perl・Python・PHP ではこれは文字列です。

size （C・C++・RPG のみ）データの入ったメモリー領域の長さをバイト単位で表したもの。

optlist 表 2.6 に従ったオプションリスト。次のオプションが使えます。*copy*

詳細 仮想ファイル名は、入力ファイルを使うあらゆる API メソッドに与えることができます。生成される PDF 出力を内容とする PVF ファイルを作成するには、*PDF_begin_document( )* の *createpvf* オプションを用います。こうしたメソッドのなかには、データが不要になるまで仮想ファイルをロックできるものもあります。仮想ファイルは、*PDF_delete_pvf( )* で明示的に、または *PDF_delete( )* で自動的に削除されるまでメモリー内に保持されます。

PDFlib オブジェクトはそれぞれ、PVF ファイルのセットを独立して保持します。仮想ファイルは、複数の PDFlib オブジェクト間で共有することはできませんが、それを使って同じ PDFlib オブジェクトから複数の文書を作成することはできます。別々の PDFlib オブジェクトを用いて動作している複数のスレッドは、PVF の使用を同期させる必要はありません。*filename* という仮想ファイルがすでにあるときは、例外が発生します。このメソッドは、通常のディスクファイルですでに *filename* が使われていないかという検証は行いません。

copy オプションを与えていないときは、ペアになる *PDF_delete_pvf( )* を呼び出して成功するまでは、与えたデータを呼び出し側で変更したり解放（削除）してはいけません。この規則に従わないとおそらくクラッシュが発生します。

スコープ 任意

表 2.6 PDF_create_pvf( ) のオプション

オプション	説明
copy	（論理値）true にすると、PDFlib は、与えたデータの内部的なコピーを作成します。この場合、呼び出し側は与えたデータをこの呼び出しのすぐ後に捨ててもかまいません。デフォルト：C・C++ では false、しかしその他すべての言語バインディングでは true

C++ Java C# ***int delete_pvf(String filename)***

Perl PHP ***int delete_pvf(string filename)***

C ***int PDF_delete_pvf(PDF *p, const char *filename, int len)***

名前付きの仮想ファイルを削除して、そのデータ構造を解放します。

filename （名前文字列。グローバル *filenamehandling* オプションに従って解釈されます。表 2.1 参照）*PDF_create_pvf( )* に与えてあるのと同じ仮想ファイル名。

len （C 言語バインディングのみ）*filename* の長さ（バイト単位）。*len=0* にすると null 終了文字列を与える必要があります。

戻り値 その仮想ファイルがあるがロックされているときは -1（PHP では 0）、それ以外なら 1。*filename* が、有効な仮想ファイルを参照していないときには 1 が返されます。

詳細 ファイルがロックされていなければ、PDFlib は *filename* に関連づいたデータ構造を削除します。このメソッドを呼び出して成功した後は、*filename* は使いまわすこともできます。仮想ファイルは、*PDF_delete( )* ですべて自動的に削除されます。

細かい意味は、ペアになる *PDF_create_pvf( )* を呼び出した時に *copy* オプションを与えておいたかによって異なります。*copy* オプションを与えておいたなら、ファイルのための管理データ構造もファイル内容本体（データ）も両方解放されますが、そうでないなら内容はクライアントが解放するという前提なので解放されません。

スコープ 任意

C++ Java C# ***double info_pvf(String filename, String keyword)***

Perl PHP ***float info_pvf(string filename, string keyword)***

C ***double PDF_info_pvf(PDF *p, const char *filename, int len, const char *keyword)***

仮想ファイルか PDFlib 仮想ファイルシステム（PVF）の特性をクエリーします。

filename （名前文字列）仮想ファイルの名前。このファイル名は、*keyword=filecount* の場合には空でかまいません。

len （C 言語バインディングのみ）*filename* の長さ（バイト単位で）。*len=0* の場合には、ヌル終端文字列を与える必要があります。

keyword 表 2.6 に従ったキーワード。

表 2.7 PDF_info_pvf( ) のキーワード

キーワード	説明
filecount	カレント PDFlib オブジェクトのために維持されている PDFlib 仮想ファイルシステム内のファイルの総数。filename 引数は無視されます。
exists	そのファイルが PDFlib 仮想ファイルシステム内に存在している（かつ削除されていない）場合には 1、そうでないなら 0
size	（存在する仮想ファイルに対してのみ）指定された仮想ファイルのサイズをバイト単位で。
iscopy	（存在する仮想ファイルに対してのみ）指定された仮想ファイルが作成された際に copy オプションが与えられたなら 1、そうでないなら 0

表 2.7 PDF_info_pvf( ) のキーワード

キーワード	説明
lockcount	（存在する仮想ファイルに対してのみ）指定された仮想ファイルに対して PDFlib API メソッド群によって内部的にかけられたロックの数。ファイルは、このロックカウントが 0 のときにのみ削除できます。

戻り値 *keyword* によって要求された何らかのファイルパラメーターの値。

詳細 このメソッドは、仮想ファイルか PDFlib 仮想ファイルシステム（PVF）のさまざまな特性を返します。特性は ***keyword*** で選択されます。

スコープ 任意

2.4 データをネットワークからダウンロード

C++ Java C# ***int download(string filename, string optlist)***

Perl PHP ***int download(string filename, string optlist)***

C ***int PDF_download(PDF *p, const char *filename, int len, const char *optlist)***

データをネットワークリソースからダウンロードして、それをディスクベースか仮想のファイルの中に保存します。

注 このメソッドは IBM System i では利用できません。

filename （名前文字列）ダウンロードされたデータを保存したいディスクベースか仮想のターゲットファイルの名前。

len （C 言語バインディングのみ）*filename* の長さ（バイト単位で）。*len=0* ならば、null で終わる文字列を与える必要があります。

optlist 表 2.8 に従ったオプションリスト。以下のオプションを使えます：
createpvf・*source*

戻り値 エラーの場合には -1（PHP では 0）、そうでないなら 1

詳細 このメソッドは、データをネットワークリソースからダウンロードして、それをディスクベースか仮想の（*createpvf* オプションによる）ファイルの中に保管します。キャッシング機能が次のように効きます：もし *createpvf=true* であり、かつ、以前の呼び出しと同じ *filename* と URL がまた与えられたときには、以前のその呼び出しで得ていた既存のデータが使われ、今回の呼び出しはネットワーク活動を一切行わず成功します。キャッシングが望ましくない場合には（たとえば、同一の URL にアクセスするたび違うデータを送って来る、動的に生成されるリソースの場合など）、今回の呼び出しに先立って PVF ファイルを *PDF_delete_pvf( )* で削除しておくか、または違うファイル名を与える必要があります。ディスクベースのファイルに対してはキャッシングは利用できません。

ネットワーク操作の詳細をオプション群で制御することもできます。次のプロトコルに対応しています：*file*・*http*・*https*・*ftp*。

スコープ 任意。

表 2.8 PDF_download( ) のオプション

オプション	説明
createpvf	（論理値）true にすると、ダウンロードされたデータを内容とする PVF ファイルが生成されます。もし指定された名前のファイルがすでに PVF 内に存在しており、かつ、同一 URL への PDF_download( ) への前回の呼び出しで得ていたデータを内容としているときには、今回の呼び出しはそのデータを再度ダウンロードせず成功します（すなわちキャッシング）。PVF ファイルの内容に直接アクセスはできないことから、このオプションが有用なのは、ダウンロードされたデータを PDFlib への入力ファイルとして与える場合のみです。 このオプションを false にすると、ディスクベースのファイルが生成され、ダウンロードされたデータがその内容となります。デフォルト：false
source	（ネットワークオプションリスト。必須）表 2.9 に従ってネットワーク操作の詳細を制御するためのオプション群。

表 2.9 ネットワークオプションリスト（PDF_download( ) の source オプションと PDF_set_option( ) の network オプション）のサブオプション

オプション	説明
disable	（論理値。PDF_set_option( ) でのみ可）true にすると、ネットワーク操作がすべて抑止されます。デフォルト：false
httpauthentication	（キーワード。http でのみ可）試行させたい認証方式（複数可）。サーバーによっては、対応していない認証方式がある（ないしは、どの方式にも対応していない）ものもあります。認証の種類を明示的に設定するほうが、パフォーマンスの観点からデフォルトよりも望ましい場合もあります（たとえ結果的に同じメソッドが選ばれたとしても）。使えるキーワード（デフォルト：any）： ***any*** そのサーバーが対応している認証方式のうち最もセキュアなものを選択。 ***anysafe*** any と同様ですがベーシック認証を除外。 ***basic*** ユーザー名とパスワードによるベーシック認証。この方式は推奨されません。なぜならユーザー名とパスワードがネットワークへプレーンテキスト形式で送信されるからです。 ***digest*** RFC 2617 に従ってハッシュ化されたユーザー名とパスワードによるダイジェスト認証。これのほうがベーシック認証よりもセキュアです。 ***ntlm*** Microsoft 製品群で使用されているものと同様の NTLM 認証
password	（文字列）ベーシック・ダイジェスト認証の場合のパスワード
proxy	（オプションリスト）プロクシサーバーを通してネットワークアクセスを構成するためのサブオプション群： ***httpauthentication*** （キーワード。http プロキシでのみ使用可能）同名のメインのネットワークオプションを参照してください。 ***noproxy*** （文字列）プロキシが必要でないホスト名のカンマ区切りリスト。数値 IPv6 アドレスは角括弧で挟まずに与える必要があります。 ***password*** （文字列）同名のメインのネットワークオプションを参照してください。 ***sslcertdir*** （文字列。https プロキシでのみ可）同名のメインのネットワークオプションを参照してください。 ***sslcertfile*** （文字列。https プロキシでのみ可）同名のメインのネットワークオプションを参照してください。 ***sslverifyhost*** （文字列。https プロキシでのみ可）同名のメインのネットワークオプションを参照してください。 ***sslverifypeer*** （文字列。https プロキシでのみ可）同名のメインのネットワークオプションを参照してください。 ***url*** （文字列。必須）プロキシサーバーのホスト名か数値 IP アドレス。URL にユーザー名とパスワードを入れることもできます。数値 IPv6 アドレスは角括弧 [...] で挟む必要があります。末尾にコロン「:」を付けてポート番号を付加することもできます。ポート番号が指定されていないときにはデフォルトポート番号 1080 が使われます。プロトコルが指定されていないときには http が使われます。 ***username*** （文字列）同名のメインのネットワークオプションを参照してください。 プロキシサーバーは、共通の環境変数 http_proxy・https_proxy・no_proxy・all_proxy を用いて構成することもできます。環境変数よりもオプションが優先します。
sslcertdir	（文字列。https でのみ可）SSL 接続を確立するために必要となる可能性のある、PEM エンコーディングの信頼済み CA 証明書群が入っているディレクトリの名前。OpenSSL の命名規則（OpenSSL コマンド c_rehash）が証明書ファイル名に適用されます。
sslcertfile	（文字列。https でのみ可）SSL 接続を確立するために必要となる可能性のある、PEM エンコーディングの、1 個ないし複数の信頼済み CA 証明書が入っているファイルの名前。

表 2.9 ネットワークオプションリスト（PDF_download( ) の source オプションと PDF_set_option( ) の network オプション）のサブオプション

オプション	説明
sslverifyhost	（論理値。https でのみ可）true にすると、接続を確立するにはサーバー証明書内の Subject Alternate Name フィールドが URL 内のホスト名と合致しなければなりません。デフォルト：true
sslverifypeer	（論理値。https でのみ可）true にすると、sslcertdir か sslcertfile オプションで与えた信頼済み証明書の集合に対して、サーバー証明書が検証可能でなければなりません。false にすると、既知の信頼済みルートが取得できないので検証できないサーバー証明書も、受容されます。デフォルト：false
timeout	（整数）リソースにアクセスする際のタイムアウトをミリ秒単位で。値 0 にするとタイムアウトは設定されません。デフォルト：5000
username	（文字列）ベーシック・ダイジェスト認証に対するユーザー名
url	（文字列。PDF_download( ) でのみ可、かつその場合には必須）ネットワークリソースの、完全修飾された URL。http:// などプロトコル識別子を含める必要があります。キャラクターを %20 など URL エンコーディングで指定することもできます。ユーザー名とパスワードを標準文法で URL に含めることもできます。

2.5 PDF オブジェクト作成 API（POCA）

オブジェクト種別と凍結オブジェクト　PDF オブジェクト作成 API（PDF object creation API ＝ POCA）は、PDF オブジェクトを作成するための低レベルインターフェイスです。POCA は以下のオブジェクト種別に対応しています：

- 単純オブジェクト種別：論理値・整数・名前・float・文字列。
- コンテナオブジェクト種別：配列・辞書・ストリーム。
- PDFlib ブロック独自の種別：パーセント値・色。

PDF オブジェクトを表す POCA ハンドルは以下のように使用できます：

- *PDF_begin/end_dpart( )* の *dpm* オプションを用いて、文書部分メタデータを作成。
- *PDF_begin/end_page_ext( )*の***blocks***オプションを用いて、PPSで使用するためのPDFlibブロックを作成。
- *PDF_create_action( )* の ***richmediaargs*** オプションを用いて、リッチメディア注釈に関連付けられた JavaScript のための引数群を指定。

PDF コンテナオブジェクトを、上に挙げたオプションのいずれかに与えると、そのコンテナオブジェクト自体と、そのコンテナから直接または間接に参照されているすべてのオブジェクトが、すなわち、そのコンテナによって作成されたオブジェクトツリー全体が凍結されます。凍結されたオブジェクトは、上記のオプションで再び使うことはできますが、*PDF_poca_insert( )* または *PDF_poca_remove( )* を用いて変更することはできなくなります。

C++ Java C# ***int poca_new(String optlist)***

Perl PHP ***int poca_new(string optlist)***

C ***int PDF_poca_new(PDF *p, const char *optlist)***

種別が辞書か配列の新規 PDF コンテナオブジェクトを作成し、オブジェクト群を挿入します。

optlist　コンテナを作成して中身を入れるためのオプションリスト。

- 表 2.10 に従った、コンテナを作成するためのオプション：***containertype*・*usage***
- 表 2.12 に従った、コンテナ内にオブジェクト群を挿入するためのオプション：***direct*・*hypertextencoding*・*index*・*key*・*type*・*value*・*values***

戻り値　POCA コンテナハンドル。これは、*PDF_poca_delete( )* を用いて削除されるまで使用できます。

詳細　このメソッドは、指定されたコンテナ種別の、空の PDF コンテナオブジェクトを作成します。このコンテナは、同一呼び出し内でただちに中身を入れることもできますし、後から *PDF_poca_insert( )* への呼び出しで中身を入れることもできます。

PDF/VT　種別が辞書で *usage=dpm* のオブジェクトに対する POCA コンテナハンドルは、*PDF_begin/end_dpart( )* の *dpm* オプションを用いて文書部分メタデータ（DPM）として与えることができます。

スコープ　任意

表 2.10 PDF_poca_new( ) のオプション

オプション	説明
container-type	（キーワード。必須）コンテナの種別：dict か array。指定されていない配列スロットと、新規オブジェクトを挿入することなく除去されている配列スロットは、PDF 出力内でキーワード null を含みます。
usage	（キーワード。必須）その新規コンテナが使用されるコンテキスト。このオプションは、そのコンテナが意図する用途に適合しているかを確認するいくつかのチェックを有効にします。 ***blocks*** （containertype=dict の場合にのみ意味を持ちます。製品でのみ可）そのブロック辞書（PDF_begin/end_page_ext( ) の blocks オプションに与えられるコンテナ）は、1 個ないし複数の PDFlib ブロック定義を内容とする必要があります。このオプション usage=blocks は、この新規辞書内に直接または間接に挿入されるすべてのコンテナオブジェクトに対しても与える必要があります。 ***dpm*** （containertype=dict の場合にのみ意味を持ちます）その新規辞書と、その中に含まれるすべての辞書の中のすべてのキーは、ASCII キャラクターで構成され、XML NMTOKEN の規則に準拠する必要があります。これにより、その辞書が文書部分メタデータ（DPM）辞書として使用できることが保証されます。このオプション usage=dpm、この新規辞書内に直接または間接に挿入されるすべてのコンテナオブジェクトに対しても与える必要があります。 ***richmediaargs*** （containertype=array の場合にのみ可）その配列は、種別が文字列か整数か float か論理値のオブジェクトを内容とすることができます。

C++ Java C# ***void poca_delete(int container, String optlist)***

Perl PHP ***poca_delete(int container, string optlist)***

C ***void PDF_poca_delete(PDF *p, int container, const char *optlist)***

PDF コンテナオブジェクトを削除します。

container *PDF_poca_new( )* を用いて取得された有効な PDF コンテナハンドル。

optlist 表 2.11 に従ったオプションリスト。以下のオプションを使えます：
recursive

詳細 そのコンテナは削除され、もう使えなくなります。そのコンテナが他の辞書または配列から参照されている場合には、その削除されるコンテナへのすべての辞書参照は除去され、その削除されるコンテナへのすべての配列参照は *null* オブジェクトへ置き換えられます。POCA コンテナオブジェクトは、*PDF_end_document( )* で自動的に削除されません。

スコープ 任意。対応する *PDF_poca_new( )* への呼び出しと必ずペアにする必要があります。

表 2.11 PDF_poca_delete( ) のオプション

オプション	説明
recursive	（論理値）true にすると、そのコンテナオブジェクト自体と、それから参照されているすべてのオブジェクトが再帰的に削除されます。これは、オブジェクトツリー全体を削除するためのショートカットとして有用でしょう。デフォルト：false

C++ Java C# ***void poca_insert(int container, String optlist)***

Perl PHP ***poca_insert(int container, string optlist)***

C ***void PDF_poca_insert(PDF *p, int container, const char *optlist)***

PDF コンテナオブジェクト内に単純またはコンテナオブジェクトを挿入します。

container *PDF_poca_new( )* を用いて取得された有効な POCA コンテナハンドル。凍結されたコンテナ（41 ページ「オブジェクト種別と凍結オブジェクト」参照）は、もう変更できませんので、許容されません。

optlist 表 2.12 に従ったオプションリスト。以下のオプションを使えます：
direct・*hypertextencoding*・*index*・*key*・*type*・*value*・*values*

詳細 このメソッドは、コンテナ内にオブジェクトを挿入します。コンテナ内にオブジェクトが挿入される順序は重要ではありません。挿入されたコンテナには、挿入後に中身を入れることもできます。挿入されるコンテナが、挿入の時点で完成している必要はありません。

コンテナ内へオブジェクトを挿入する際には、オブジェクトヒエラルキー内に直接オブジェクトのループを作成してはいけません。たとえば、直接挿入される辞書は、そのコンテナへの直接参照を含んでいてはいけません。循環参照を作成するためには、*direct=false* を用いて間接オブジェクトを作成します。この間接オブジェクトは、任意の他のオブジェクトを参照することができます。

スコープ 任意

表 2.12 PDF_poca_new( )・PDF_poca_insert( )・PDF_poca_remove( ) のオプション

オプション	説明
direct[1]	（論理値。type=array・dict の場合にのみ可。それ以外の種別では無視されます）true にすると、そのオブジェクトはそのコンテナ内に直接挿入されます。false にすると、間接 PDF オブジェクトが作成され、その間接 PDF オブジェクトへの参照がそのコンテナ内に挿入されます。間接オブジェクトは、1 個のオブジェクトが複数回使用される場合に、生成される PDF 内の容量を節約するために有用です。デフォルト：true
hypertext-encoding	（キーワード）key・value・values オプションのためのエンコーディング。空文字列は unicode と等価です。デフォルト：グローバル hypertextencoding オプションの値
index	（整数。type=array のコンテナの場合にのみ可。PDF_poca_remove( ) の場合には必須）配列内で値（1 個ないし複数）が挿入または削除される場所の、ゼロベースの番号。値 -1 を用いると、そのエレメントを新規末尾アイテムとして挿入することができます。配列は、指定された番号でエレメントを含めるために必要に応じて拡大されます。その配列が指定された番号にすでに値を持っているときは、それは新しい値で置き換えられます。PDF_poca_new( )・PDF_poca_insert( ) に対するデフォルト：-1
key	（ハイパーテキスト文字列。type=dict のコンテナの場合にのみ可、かつその場合は必須）辞書コンテナの中の、値を挿入したいキー。このキーは、先頭の「/」スラッシュキャラクターを含んではいけません。このキーは、その辞書の usage オプションで指定された条件に準拠する必要があります。その辞書がすでに同じキーのエントリーを持っている場合には、それは新しい値へ置き換えられます。
type[1]	（キーワード。必須）挿入されるオブジェクトの種別：array・boolean・dict・integer・name・float・stream・string・percentage・color そのコンテナが usage=dpm を用いて作成されている場合には、以下の種別は許容されません：name（かわりに type=string を用います） 以下の種別は、そのコンテナが usage=blocks を用いて作成されている場合にのみ許容されます：color・percentage

表 2.12 PDF_poca_new( ) ・ PDF_poca_insert( ) ・ PDF_poca_remove( ) のオプション

オプション	説明
value[1]	(type オプションに従ったデータ型。オプション value か values のいずれか一つのみを必ず与える必要があります) コンテナ種別と type オプションに従った、挿入されるオブジェクトの値 : 配列・辞書コンテナの場合 : type=boolean ならば、この値はオプション種別 string を持つ必要があり、かつ文字列 true か false のいずれか一つを内容とする必要があります。 type=string か name ならば、この値はオプション種別 Hypertext string を持つ必要があり、かつそのターゲットを直接内容とする必要があります。type=name の場合の値は、UTF-8 表現で 127 バイトを上限とし、かつ「/」スラッシュキャラクターを先頭に付けてはいけません。 type=integer ならば、この値はオプション種別 integer を持つ必要があり、かつそのターゲットを直接内容とする必要があります。 type=float ならば、この値はオプション種別 float か integer を持つ必要があり、かつそのターゲットを直接内容とする必要があります。 type=array か dict ならば、この値はオプション種別 POCA コンテナハンドル (すなわち PDF_poca_new( ) を用いて作成された) を持つ必要があり、かつ挿入されるコンテナを指定する必要があります。挿入されるオブジェクトは、このコンテナと同じ usage オプションを用いて作成されている必要があります。 type=percentage ならば、この値はオプション種別 number を持つ必要があります。これはパーセント値として解釈され、パーセント記号を含む必要があります (例 : 50%)。これはブロックデータ種別パーセント値として書き込まれます。 type=color ならば、この値はオプション種別 color を持つ必要があります (表 1.2 (15 ページ) 参照)。これはブロックデータ種別色として書き込まれます。以下の色空間キーワードは許容されません : iccbased ・ iccbasedgray ・ iccbasedrgb ・ iccbasedcmyk ・ pattern ・ devicen このオプションを用いて任意の文字列を渡すためには、10 ページ「括弧で囲われていない文字列」に記述したオプションリスト文法が有用でしょう。
values[1]	(type オプションに従った、1 個ないし複数の値のリスト。type=array を用いたコンテナの場合にのみ可。オプション value か values のいずれか一つのみを必ず与える必要があります) 配列内の、index オプションで指定された位置に挿入される、同じ種別の 1 個ないし複数の値。個別の種別に対する条件についてオプション value を参照してください。この指定されたリストが要素 1 個だけを内容とする場合には、その効果は value オプションと同じです。このリストが複数の要素を内容とする場合には、このリスト内のすべての値がその配列内へ順番に挿入され、既存の要素群をオーバーライドする可能性もあります。この配列は、この指定されたリスト内のすべての要素群を含むように、必要に応じて拡大します。

1. PDF_poca_new( ) ・ PDF_poca_insert( ) の場合のみ

C++ Java C# ***void poca_remove(int container, String optlist))***

Perl PHP ***poca_remove(int container, string optlist)***

C ***void PDF_poca_remove(PDF *p, int container, const char *optlist)***

PDF コンテナオブジェクトから、単純またはコンテナオブジェクトを除去します。

container *PDF_poca_new( )* を用いて取得された有効な POCA 辞書または配列ハンドル。凍結されたコンテナ (41 ページ「オブジェクト種別と凍結オブジェクト」参照) は、もう変更できませんので、許容されません。

optlist 表 2.12 内の *PDF_poca_insert( )* の以下のオプションを使えます :
hypertextencoding ・ index ・ key

詳細 このメソッドは、種別配列か辞書のコンテナからオブジェクトを除去します。この指定されたオブジェクトがそのコンテナ内に存在しないときは何も起こりません。

スコープ 任意

3 文書・ページメソッド

この章の API メソッド：

- *PDF_begin_document()*
- *PDF_begin_document_callback()*
- *PDF_end_document()*
- *PDF_get_buffer()*
- *PDF_begin_page_ext()*
- *PDF_end_page_ext()*
- *PDF_suspend_page()*
- *PDF_resume_page()*
- *PDF_define_layer()*
- *PDF_set_layer_dependency()*
- *PDF_begin_layer()*
- *PDF_end_layer()*

3.1 文書メソッド

C++ Java C# ***int begin_document(String filename, String optlist)***

Perl PHP ***int begin_document(string filename, string optlist)***

C ***int PDF_begin_document(PDF *p, const char *filename, int len, const char *optlist)***

C++ ***void begin_document_callback(size_t (*writeproc) (^PDF *p, void *data, size_t size), string optlist)***

C ***void PDF_begin_document_callback(PDF *p,***
size_t (*writeproc) (PDF *p, void *data, size_t size), const char *optlist)

新規 PDF ファイルをさまざまなオプションに従って作成します。

filename （名前文字列。グローバル *filenamehandling* オプションに従って解釈されます。表 2.1 参照）生成したい PDF 出力ファイルの絶対名または相対名。*filename* が空ならば、PDF 文書はファイル上でなくメモリー内に生成され、その生成 PDF データをクライアントへ取り出すには *PDF_get_buffer()* メソッドを使う必要があります。Windows では、UNC パスや、割り当てられたネットワークドライブを使ってもかまいません。

len （C 言語バインディングのみ）*filename* の長さ（バイト単位）。*len=0* にすると null 終了文字列を与える必要があります。

writeproc （C・C++ 言語バインディングのみ）生成された PDF データ（の一部分ずつ）を受け渡すために PDFlib によって呼び出される C コールバック関数。

optlist 文書オプション群を指定したオプションリスト：

- 一般オプション：*errorpolicy*（表 1.5 参照）・*hypertextencoding*（表 2.1 参照）
- 表 3.1 に従った文書オプション。これらのオプションのうちのいくつかは、*PDF_end_document()* でも指定できます。その場合にはそれらは、*PDF_begin_document()* で指定した同等のオプションよりも優先されます：

attachments • *autoxmp* • *destination* • *emitdocinfo* • *groups* • *labels* • *linearize* • *metadata* • *objectstreams* • *openmode* • *optimize* • *pagelayout* • *portfolio* • *search* • *uri* • *viewerpreferences*

- 表 3.2 に従った、PDF 互換性と規格のためのオプション：
  compatibility • *limitcheck* • *nodenamelist* • *pdfa* • *pdfua* • *pdfvt* • *pdfx* • *recordlevel* • *usestransparency*
- 表 3.3 に従った、タグ付き PDF のためのオプション：
  checktags • *lang* • *pdftagset* • *rolemap* • *structureassociatedfiles* • *structuretype* • *tag* • *tagged* • *tagsets*
- 表 3.4 に従ったセキュリティーオプション：
  attachmentpassword • *masterpassword* • *permissions* • *userpassword*
- 表 3.5 に従った出力処理オプション：
  createoutput • *createpvf* • *filemode* • *flush* • *inmemory* • *recordsize* • *removefragments* • *tempdirname* • *tempfilenames*

戻り値 エラーなら -1（PHP では 0）、そうでなければ 1。*filename* が空のときは、このメソッドは必ず成功し、決してエラー値を返しません。

詳細 このメソッドは、与えられた *filename* を使って新規 PDF ファイルを作成します。PDFlib は、その与えられた名前のファイルを開こうと試み、そしてその PDF 文書が完了したときにはファイルを閉じます。

PDF_begin_document_callback( ) は、新規 PDF 文書を開きますが、ディスクファイルには書き込むのではなく、ユーザーから与えられたコールバックメソッドを呼び出して PDF 出力データを受け渡します。*writeproc* で与える関数は、書かれたバイト数を返す必要があります。もしその戻り値が、PDFlib が与える引数 *size* と一致しないと、例外が発生します。*writeproc* の呼び出される頻度は *flush* オプションで設定できます。
このコールバック関数は、*PDF_get_opaque( )* 以外、同一のコンテキストポインターを用いた API メソッドまたはマクロを呼び出してはいけません。

スコープ **オブジェクト**。このメソッドは、ファイルをうまく開けたときは**文書**スコープを開始させます。対応する *PDF_end_document( )* と必ずペアにして呼び出す必要があります。

バインディング ASP：このメソッドに渡すフルパス名を作るには *MapPath* 機能を使う必要があります。

C・C++・Java・JScript：バックスラッシュのパス区切りを適切にエスケープするよう気をつけてください。たとえば、ネットワークドライブ上のファイルは次のように表します。*\\\\malik\\rp\\foo.pdf*。
PDF_begin_document_callback( ) は C・C++ でのみ利用可能です。

C++ Java C# ***void end_document(String optlist)***
Perl PHP ***end_document(string optlist)***
C ***void PDF_end_document(PDF *p, const char *optlist)***

生成された PDF ファイルを閉じて、さまざまなオプションを適用します。

optlist 文書処理オプション群を指定したオプションリスト：

- 一般オプション：*hypertextencoding*（表 2.1 参照）
- 表 3.1 に従った文書オプション。*PDF_end_document( )* で指定するオプションは、*PDF_begin_document( )* で指定した同等のオプションよりも優先されます。以下のオプションが使えます：

action・associatedfiles・attachments・autoxmp・destination・destname・labels・metadata・openmode・pagelayout・portfolio・uri・viewerpreferences

詳細 このメソッドは、生成された PDF 文書を完了し、文書に関連するリソースをすべて解放し、その PDF 文書が *PDF_begin_document()* で開かれていたときには出力ファイルを閉じます。このメソッドは、PDF 文書を開いたときの手段が何であろうと、クライアントがすべてのページを生成しおわった時点で呼び出す必要があります。

文書をメモリー内に生成したときは（ファイル上でなく）、その文書のバッファはこのメソッドを呼び出した後も保持され（*PDF_get_buffer()* で取り出せるように）、次に *PDF_begin_document()* を呼び出したときか、または PDFlib オブジェクトがスコープ外に出たときに解放されます。

スコープ **文書**。このメソッドは**文書**スコープを終了させます。対応する *PDF_begin_document()* 関数か *PDF_begin_document_callback()* 関数のいずれかと必ずペアにして呼び出す必要があります。

表 3.1 PDF_begin_document( )・PDF_end_document( ) の文書オプション

オプション	説明
action[1]	（アクションリスト）以下の 1 個ないし複数のトリガーイベントに対する文書アクションのリスト（デフォルト：空リスト）： ***open*** 文書が開かれた時に実行させたいアクション。Acrobat 内の実行順序の関係上、文書レベルの JavaScript を open アクションとして使うことはできません。 ***didprint/didsave/willclose/willprint/willsave*** （PDF/A では不可）文書の印刷後 / 文書の保存後 / 文書を閉じる前 / 文書の印刷前 / 文書の保存前に実行させたい JavaScript アクション。 この action オプションに与えたアクションはそれぞれ、PDF_open_pdi_document( ) の useactions オプションを用いて PDF 文書から取り込まれたアクションをオーバーライドします。文書レベルの JavaScript は、PDF_create_action( ) の script・scriptname オプションを用いて作成できます。
associated-files[1]	（アセットハンドルのリスト。PDF 2.0・PDF/A-3 でのみ可）文書全体に紐付けるファイル群のアセットハンドル群。このファイル群は、PDF_load_asset( ) で type=attachment を用いて読み込んである必要があります。
attachments	（オプションリストのリストまたはアセットハンドルのリスト。PDF/X-3・PDF/A-1 では不可。PDF/A-2：PDF/A-1・PDF/A-2 文書のみ添付可。PDF/A-3：許容されませんので、かわりに associatedfiles を使用してください）PDF_load_asset( ) で type=attachment を用いて読み込んである文書レベルファイル添付。PDF_begin_document( ) と PDF_end_document( ) の両方でファイル添付を与えるのは OK です。ただし、アセットハンドルは PDF_end_document( ) でのみ与えることができます。使えるサブオブオプション群：表 13.4 参照
autoxmp	（論理値。PDF/X-3/4/5・PDF/A・PDF 2.0 では強制的に true になります）true にすると、PDFlib は文書情報フィールドから XMP 文書メタデータを生成します（307 ページ「15.2 XMP メタデータ」参照）。デフォルト：true
destination	（オプションリスト。文書を開くアクションを指定している場合には無視されます）文書を開くアクションを表 12.11 に従って指定したオプションリスト。
destname[1]	（ハイパーテキスト文字列。destination オプションを指定している場合には無視されます）文書を開くアクションとして使わせたい、PDF_add_nameddest( ) で定義してある移動先の名前。
emitdocinfo	（論理値。PDF 2.0 でのみ意味を持ちます）true にすると、文書情報辞書が出力されます。デフォルト：false

表 3.1 PDF_begin_document( )・PDF_end_document( ) の文書オプション

オプション	説明
groups[2]	（文字列のリスト。PDF/VT モードの場合、または文書部分ヒエラルキーが作成されている場合には不可）文書で使いたいページグループの名前と順序を定義します。ページグループは複数のページをまとめます（ページラベルをつけるときなどに有用です）。文書で定義したページグループの 1 つにページを割り当てて、そのグループの中で指し示すことができます。文書でページグループを定義したときは、すべてのページをページグループに割り当てる必要があります。
labels	（オプションリストのリスト）シンボリックなページ名を表 3.6 に従って指定した 1 個ないし複数のオプションリストを内容として持つリスト。このページ名は Acrobat のステータスバーにページラベルとして表示されます（ページ番号のかわりに）。style・prefix・start 値の組み合わせが文書内で一意である必要があります。デフォルト：ページラベルなし
linearize[2]	（論理値。objectstreams=none を強制します）true にすると出力文書は線形化されます。z/OS では、このオプションはインコア生成（すなわち filename を空に）と併用できません。デフォルト：false
metadata	（オプションリスト。PDF 1.4）文書の XMP メタデータを与えます（307 ページ「15.2 XMP メタデータ」参照）。XMP プロパティはそれぞれ、PDF_set_info( ) で与えられた文書情報フィールドによってオーバーライドされることがあります。PDF/A モードでは、与える XMP メタデータが準拠するべき要請が追加されます（PDFlib チュートリアル参照）。
objectstreams[2]	（キーワードのリスト。PDF 1.5。linearize が true の場合には強制的に none になります）出力ファイルサイズを劇的に縮小する、圧縮されたオブジェクトストリーム群を生成します（デフォルト：{other nodocinfo}）： ***bookmarks*** しおりオブジェクト群を圧縮。 ***docinfo*** 文書情報フィールド群を圧縮。 ***dpartarrays*** 文書部分ヒエラルキーに関連する辞書群を圧縮。 ***dpartdicts*** 文書部分ヒエラルキーに関連する配列群を圧縮。 ***fields*** フォームフィールド群を圧縮。 ***names*** 名前付き参照先を持つオブジェクト群を圧縮。 ***none*** 圧縮されたオブジェクトストリームを一切生成しません（このオプションの後で明示的に有効にされたカテゴリー群を除いて）。 ***other*** このキーワードの後で明示的に無効にされなかったすべてのカテゴリー、およびその他の自身のキーワードを持たないオブジェクト種別群。 ***pages*** ページツリーを構成するオブジェクト群を圧縮。 ***poca*** POCA インターフェイスで作成されたすべての単純オブジェクトを圧縮。 ***tags*** マークされた内容タグ群を圧縮。 ***xref*** 圧縮された xref ストリームを生成。このカテゴリーは、他のカテゴリーを 1 つでも有効にすると自動的に有効になります。 none・other 以外のすべてのキーワードは、頭に no をつけて（例：nodocinfo）そのカテゴリーの圧縮を無効にすることができます。このような否定キーワードを 1 つでも与えると、キーワード other はリストの先頭に付加されます。
openmode	（キーワード）文書を開いた時の表示方式を設定します。デフォルト：文書にしおりがあるなら bookmarks、なければ none。 ***none*** パネルを追加表示せずに開きます。 ***bookmarks*** しおりパネルを表示して開きます。 ***thumbnails*** ページパネルを表示して開きます。 ***fullscreen*** 全画面表示で開きます（ブラウザーでは効果なし）。 ***layers*** （PDF 1.5）レイヤーパネルを表示して開きます。 ***attachments*** （PDF 1.6）ファイル添付パネルを表示して開きます。

表 3.1 PDF_begin_document( ) ・ PDF_end_document( ) の文書オプション

オプション	説明
optimize[2]	（論理値）true にすると、出力文書が生成された後、別途のパスで最適化されます。最適化は、冗長な重複オブジェクトを除いてファイルサイズを小さくします。通常、クライアントコードに非効率がある場合（同一画像や同一 ICC プロファイルを、ハンドルの再利用をせずに、何度も読込んだ場合など）を除き、最適化は目立った効果をもたらしません。z/OS では、このオプションはインコア生成（すなわち filename を空に）と併用できません。デフォルト：false
pagelayout	（キーワード）文書を開いた時に使わせたいページレイアウト（デフォルト：default）： ***default*** PDF ビューアーのデフォルト設定。 ***singlepage*** 1 ページずつ表示。 ***onecolumn*** ページを縦一列に並べて表示。 ***twocolumnleft*** ページを見開きで、奇数ページを左に表示。 ***twocolumnright*** ページを見開きで、奇数ページを右に表示。 ***twopageleft*** （PDF 1.5）2 ページずつ、奇数番号ページを左に表示。 ***twopageright*** （PDF 1.5）2 ページずつ、奇数番号 ページを右に表示。
portfolio[1]	（オプションリスト。PDF 1.7）表 15.5 に従った、PDF ポートフォリオを生成するためのサブオプション群
search[2]	（オプションリスト。廃止。PDF 2.0 では利用できません）文書を開く時に検索インデックスを読み込むよう Acrobat に命令します： ***filename*** （ハイパーテキスト文字列。必須）検索インデックスの入ったファイルの名前。 ***indextype*** （名前文字列）インデックスの種類。Acrobat に対しては PDX とする必要があります。デフォルト：PDX
uri	（文字列）文書にベース URL を設定します。これは、他の文書への相対 Web リンクを持った文書を他の所へ移すときに便利です。ベース URL を「元の」場所に設定しておけば、相対リンクは確実に有効でありつづけます。デフォルト：ベース URI なし
viewer-preferences	（オプションリスト）さまざまな表示設定を表 3.7 に従って指定したオプションリスト。デフォルト：空

1. PDF_end_document( ) でのみ可
2. PDF_begin_document( ) ・ PDF_begin_document_callback( ) でのみ可

表 3.2 PDF_begin_document( ) の PDF 互換性・規格のためのオプション

オプション	説明
compatibility	（キーワード。pdfa か pdfua か pdfvt か pdfx オプションが none 以外の値で用いられている場合には無視されます）文書の PDF バージョンを、以下のキーワードのいずれかに設定します。このオプションは、どの PDF 機能が利用可能かと、どの PDF 文書を PDFlib+PDI で取り込めるかに影響を与えます（デフォルト：1.7ext8）： ***1.4*** PDF 1.4。Acrobat 5 以上が必要。 ***1.5*** PDF 1.5。Acrobat 6 以上が必要。 ***1.6*** PDF 1.6。Acrobat 7 以上が必要。 ***1.7*** PDF 1.7。ISO 32000-1 で定義されており、Acrobat 8 以上が必要。 ***1.7ext3*** PDF 1.7 拡張レベル 3。Acrobat 9 以上が必要。 ***1.7ext8*** PDF 1.7 拡張レベル 8。Acrobat X 以上が必要。 ***2.0*** PDF 2.0。ISO 32000-2 で定義されています。
limitcheck	true にすると、PDF/A-1/2/3・PDF/X-4/5 モードにおいて間接 PDF オブジェクトの数の制限（8,388,607）が強制されます。デフォルト：true

表 3.2 PDF_begin_document( ) の PDF 互換性・規格のためのオプション

オプション	説明
nodenamelist	（名前文字列のリスト。pdfvt=PDF/VT-1 では必須）文書部分ヒエラルキーのすべてのレベル群に対する名前群。すべての名前は、ASCII キャラクター群で構成されている必要があり、かつ XML NMTOKEN の規則群に準拠している必要があります。最初の文字列が、その文書部分ヒエラルキー内のレベルに対する名前を指定します。
pdfa	（キーワード）PDF/A 準拠レベルを、以下のいずれか一つに設定します（デフォルト：none）： ***PDF/A-1a:2005*・*PDF/A-1b:2005*** compatibility=1.4 を暗黙に前提します ***PDF/A-2a*・*PDF/A-2b*・*PDF/A-2u*** compatibility=1.7 を暗黙に前提します ***PDF/A-3a*・*PDF/A-3b*・*PDF/A-3u*** compatibility=1.7 を暗黙に前提します ***none*** PDF/A 出力なし PDF/A1-a:2005・PDF/A-2a・PDF/A-3a は tagged=true. を暗黙に前提します。PDF/A-1/2/3 は、以下の他の規格に互換です： pdfx=PDF/X-3:2003・PDF/X-4 pdfvt=PDF/VT-1 pdfua=PDF/UA-1 PDF 規格に対して複数のオプションが指定された場合には、最も低い互換性値が用いられます。
pdfua	（キーワード）PDF/UA 準拠レベルを、以下のいずれか一つに設定します（デフォルト：none）： ***PDF/UA-1*** compatibility=1.7 と tagged=true を暗黙に前提します。 ***none*** PDF/UA 出力なし
pdfvt	（キーワード）PDF/VT 準拠レベルを、以下のいずれか一つに設定します（デフォルト：none）： ***PDF/VT-1*** pdfx=PDF/X-4 を暗黙に前提します。pdfx オプションに対するこれ以外の値はエラーです。 ***none*** PDF/VT 出力なし
pdfx	（キーワード）PDF/X 準拠レベルを、以下のいずれか一つに設定します（デフォルト：none）： ***PDF/X-1a:2003*** compatibility=1.4 を暗黙に前提します。廃止 ***PDF/X-3:2003*** compatibility=1.4 を暗黙に前提します。廃止 ***PDF/X-4*・*PDF/X-4p*** compatibility=1.6 を暗黙に前提します ***PDF/X-5n*** compatibility=1.6 を暗黙に前提します ***none*** PDF/X 出力なし
recordlevel	（非負整数。文書部分ヒエラルキーが作成されている場合にのみ意味を持ちます）受領者レコード群に対応する、文書部分ヒエラルキーのゼロベースのレベル。
uses-transparency	（論理値。PDF/A-1・PDF/X-3 では不可）false にすると、生成される文書の中のページはいずれも透過オブジェクトを一切含みません。PDFlib は、この表明が違反された場合には例外を発生させます。このオプションを false に設定することは、透過を持たない文書に対してのみ許容されます。その場合、取り込まれる PDF ページ群と SVG グラフィック群に対して、透過グループが一切自動生成されないことになります。かつ、PDF/VT のためのカプセル化フォーム XObject が透過グループを伴わずに生成されることになり、より効率的な RIP キャッシングを可能にします。デフォルト：true

表 3.3 PDF_begin_document( ) ・ PDF_end_document( ) のタグ付き PDF のためのオプション

オプション	説明
checktags[1]	（キーワード。PDF/UA-1 モードでは strict とする必要があります）タグ群について構造エレメントネスト化規則群（PDFlib チュートリアル参照）がチェックされるかどうかを指定します。このオプションは、pdftagset=2.0 の場合、PDF 2.0 で廃止されているタグ群の使用をも制御します。これは移植支援のためだけに提供されています。このオプションは、取り込まれるページ群の中のタグ群には影響を与えません（PDF_open_pdi_document( ) のオプション checktags を参照）。使えるキーワード（デフォルト：strict）： ***none*** タグネスト化規則群は強制されません。この設定は、無効な構造ヒエラルキーを生み出すおそれがありますので、推奨されません。 ***relaxed*** strict と同様ですが、ただし PDF 1.7 のいくつかの規則が強制されません（PDFlib チュートリアル参照）。 ***strict*** タグがネスト化規則群に違反しているときには、例外が発生します。PDF 2.0 モードでは、廃止されたタグも例外を発生させます。
lang	（文字列。PDF/UA-1 に対しては必須）文書のデフォルト言語を、ISO 639-1/2 に従って 2 字か 3 字の言語コードで構成された BCP 47 言語タグとして設定します。（例：de ・ en ・ fr ・ ja ・ fil）。その末尾に、ハイフンと 2 字の ISO 3166 リージョンコードをつけることも可能です（例：en-us ・ en-gb ・ es-mx）。大文字・小文字は区別されません。値 zxx は、プログラムコードなど非言語内容を意味します。空文字列は、言語が未知であることを意味します（これは PDF/UA モードでは許されません）。 このデフォルト言語は、個別の構造エレメントについて、lang タグ付けオプションを用いてオーバーライドすることも可能です。
pdftagset[1]	（キーワード。PDF 2.0）標準 PDF エレメント種別に対するタグセット。文書内で使用されている標準 PDF 種別がすべて、この選択したタグセットの中になければならず、かつ、対応するネスト化規則群が適用されます（デフォルト：compatibility=2.0 なら 2.0、そうでないなら 1.7）： ***2.0*** PDF 2.0 のタグセットとネスト化規則群 ***1.7*** PDF 1.7 のタグセットとネスト化規則群
rolemap[1]	（文字列リストのリスト。各文字列リストの 1 番目の要素は名前文字列、2 番目の要素は文字列。タグ付き PDF でのみ可。カスタム構造種別が用いられている場合には必須）カスタム構造種別から標準種別へのマッピング。各サブリストは、標準かカスタムの種別の名前と、この 1 番目の種別をマップさせたい標準種別の名前とを内容とします。インライン・擬似種別をサブリスト内の 2 番目の項目にすることは許されません。標準種別を別の標準種別へマップすることによって、既存の種別に別の意味付けを与えることもできます。間接マッピング、すなわち、カスタム種別を別のカスタム種別へマップして後者を標準種別へマップすることも許されます。同じものどうしのペアは警告なしで無視されます。 PDF/UA-1 では、標準種別を再マップすることは許容されません。
structure-associated-files[2]	（アセットハンドルのリスト。PDF 2.0）構造ツリー全体に紐付けるファイル（複数可）のアセットハンドル（複数可）。このファイル（複数可）は、PDF_load_asset( ) で type=attachment を用いて読み込んである必要があります。
structuretype[1]	（キーワード。PDF/UA-1 でのみ可）文書構造の種別（デフォルト：weak）： ***strong*** 文書は強く構造化されています。すなわち、その構造ツリーはその文書の論理的構成を反映しています。見出しに対する唯一許容される構造種別は H であり、H1 ・ H2 などは許容されません。構造ツリー内の各ノードは、高々 1 個の H タグと、1 個ないし複数の段落タグ P を内容とします。 ***weak*** 文書は弱く構造化されています。すなわち、その構造ツリーは 2 ～ 3 レベルであり、すべての見出し・段落などが直接子となっています。論理構造は見出しタグ H1 ・ H2 などで表現されることができ、H は許容されません。見出しは子孫を持つことができません。

表 3.3 PDF_begin_document( ) ・ PDF_end_document( ) のタグ付き PDF のためのオプション

オプション	説明
tag[1]	（オプションリスト）表 14.2 に従ったタグ付けオプション群。指定されある構造エレメントは、文書構造ルートを構成し、PDF_end_document( ) で自動的に閉じられます。tagname サブオプションでは、グループ化エレメント群のみが許容されます。
tagged[1]	（論理値）true にするとタグ付き PDF 出力が生成されます。タグ付き PDF モードでは、クライアントは適切な構造情報を与える必要があります。PDF/A-1a:2005 か PDF/A-2a か PDF/A-3a か PDF/UA-1 モードが有効なときは、このオプションは自動的に true に設定されます。デフォルト：false
tagsets[1]	（オプションリスト。PDF 2.0）文書内で使用できるカスタムタグセットのリスト。PDF 1.7・PDF 2.0 はこの tagsets オプションに記載する必要はありません： ***namespace***（文字列。必須）名前空間を識別するための名前を、統一資源識別子（URI）の形式で ***rolemap***（文字列リストのリスト。必須）タグセット内のすべての構造種別に対するロールマップ。このロールマップは、タグセット内の各タグのためのペアを内容としており、この各ペアは、この名前付きタグセットの中のカスタムタグと、pdftagset オプションを用いて選択されているセットの中のターゲットタグとで構成されます。

1. PDF_begin_document( ) ・ PDF_begin_document_callback( ) でのみ可
2. PDF_end_document( ) でのみ可

表 3.4 PDF_begin_document( ) のセキュリティーオプション。PDF/A・PDF/X では不可

オプション	説明
attachment-password[1]	（文字列[2]。PDF 1.6。userpassword か masterpassword が設定されている場合には無視されます。linearize・optimize オプションと組み合わせることはできません。PDF/A・PDF/X では不可）ファイル添付が、この与えた文字列をパスワードとして暗号化されます。文書の残りの部分は暗号化されません。EBCDIC プラットフォームでは、このパスワードは ebcdic エンコーディングか EBCDIC-UTF-8 と見なされます。
master-password[1]	（文字列。PDF 1.6。permissions が指定されている場合には必須。PDF/A・PDF/X では不可。PDF 1.6・1.7・1.7ext3 では暗号化は廃止）文書に対するマスターパスワード。これが空の場合には、マスターパスワードは適用されません。EBCDIC プラットフォームでは、このパスワードは ebcdic エンコーディングか EBCDIC-UTF-8 と見なされます。デフォルト：空

表 3.4 PDF_begin_document( ) のセキュリティーオプション。PDF/A・PDF/X では不可

オプション	説明
permissions	（キーワードのリスト。PDF/A・PDF/X では不可。masterpassword が必要）出力文書に対するアクセス制限のリスト。以下のキーワードを任意の数入れられます（デフォルト：空）。 ***noprint*** Acrobat でファイルを印刷できなくします。 ***nohiresprint*** Acrobat で高解像度印刷ができなくします。noprint を設定していないときは、「画像として印刷」機能でページを低解像度でレンダリングしたものしか印刷できなくなります。 ***nomodify*** Acrobat でページの編集・切り抜きとフォームフィールドの追加・変更をできなくします。 ***noassemble*** （nomodify を暗黙に前提）Acrobat でページの追加・削除・回転としおり・サムネールの作成ができなくします。 ***noannots*** Acrobat でしおり・フォームフィールドの追加・変更をできなくします。 ***noforms*** （nomodify・noannots を暗黙に前提）Acrobat でフォームフィールドへの記入をできなくします。 ***nocopy*** Acrobat でテキスト・グラフィックのコピー・抽出をできなくします。アクセシビリティーインターフェイスは noaccessible で制御します。 ***noaccessible*** （PDF 2.0 では廃止されており利用もできません。PDF/UA-1 では不可）Acrobat でアクセシビリティー（読み上げ機能など）でのテキストやグラフィックの抽出ができなくします。 ***plainmetadata*** 暗号化した文書でも、XMP 文書メタデータを暗号化しないままにします。
user-password[1]	（文字列。PDF 1.6。PDF/A・PDF/X では不可。PDF 1.6・1.7・1.7ext3 では暗号化は廃止）文書のユーザーパスワード。空にするとユーザーパスワードは適用されません。EBCDIC プラットフォームでは、このパスワードは ebcdic エンコーディングか EBCDIC-UTF-8 と見なされます。デフォルト：空

1. このオプションを用いて任意の文字列を渡すためには、10 ページ「括弧で囲われていない文字列」に記述したオプションリスト文法が有用でしょう。
2. Winansi エンコーディング外のキャラクターは、compatibility=1.7ext3 以上に対するパスワードでのみ許容されます。

表 3.5 PDF_begin_document( ) の出力処理オプション

オプション	説明
createoutput	（論理値）false にすると、filename 引数は無視され、出力ファイルまたはメモリー領域は生成されません。このオプションは、compress=0 と linearize=false と optimize=false を暗黙に前提します。デフォルト：true
createpvf	（論理値）true にすると、PDF ファイルがファイル上でなくメモリー内に生成されます。与えるファイル名は、PDF_end_document( ) の呼び出しによって生成される仮想ファイルの名前です。この場合、PDF 出力データを取り出すために PDF_get_buffer( ) を呼び出すことはできません。かわりに、生成された PVF ファイルの名前を、他の PDFlib メソッドに与えることができます。これは、PDF ポートフォリオに含める文書を生成するときに有用でしょう。デフォルト：false

表 3.5 PDF_begin_document( ) の出力処理オプション

オプション	説明
filemode	（文字列。z/OS・USS でのみ可）文書ファイルと任意の一時ファイル（linearize オプションなどによる）のファイルモードを設定するパラメーター文字列。与えた文字列は、デフォルトのファイルモード「wb,」に追加されます。recordsize オプションは、このオプションで指定するパラメーター群と整合させる必要があります。デフォルト：空、または非ブロック出力の場合（これがデフォルトです。オプション recordsize を参照）は recfm=v。 文字列の例： recfm=fb,lrecl=80,space=(cyl,(1,5) データセットの属性群がすでに割り当てられている場合にはそれに従う：recfm=*
flush	（キーワード。PDF_begin_document_callback( ) でのみ可）放出の方式を設定します（デフォルト：page）： ***none*** 文書が終わる時に 1 回だけ放出 ***page*** 各ページが終わるごとに 放出 ***content*** あらゆるフォント・画像・ファイル添付・ページが終わるごとに 放出 ***heavy*** 内部の 64 KB の文書バッファがいっぱいになるたびに 放出
inmemory	（論理値。PDF_begin_document_callback( ) では不可）true にすると、linearize か optimize オプションも true にしたときは、PDFlib は線形化のための一時ファイルを一切生成せず、ファイルをメモリー内で処理します。これはシステムによっては（特に z/OS）著しい速度向上につながりますが、文書サイズの 2 倍のメモリーが必要になります。false にすると、線形化・最適化のために一時ファイルが作成されます。デフォルト：false
recordsize	（整数。z/OS・USS でのみ可）出力ファイル、および linearize・optimize オプションで生成が必要となる可能性のあるすべての一時ファイルのレコードサイズ。デフォルト：0（非ブロック出力）
remove-fragments	true にすると、例外の後に存在する部分的な PDF 出力文書が PDF_delete( ) で除去されます。このような PDF 断片は文書としては決して使えません。このオプションは、メモリー内 PDF 生成のために空ファイル名が指定されている場合には効果を持ちません。デフォルト：false
tempdirname	（文字列。PDF_begin_document_callback( ) では不可）linearize・optimize オプションで必要な一時ファイルを生成したいディレクトリの名前。このオプションが見つからないときは、PDFlib はカレントディレクトリに一時ファイルを生成します。tempfilenames オプションを与えている場合にはこのオプションは無視されます。デフォルト：存在しない
temp-filenames	（文字列 2 個のリスト。z/OS・USS でのみ可）linearize・optimize オプションで必要な一時ファイル 2 個のフルファイル名。空にすると、PDFlib は一意な一時ファイル名を生成します。PDF_end_document( ) の後にこの一時ファイルを削除するのはユーザー側の役割です。このオプションを与えるときは、引数 filename は空にしてはいけません。デフォルト：存在しない

表 3.6 PDF_begin/end_document( ) の labels オプションと PDF_begin/end_page_ext( ) の label オプションのサブオプション

オプション	説明
group	（文字列。PDF_begin_document( ) でのみ可。文書がページグループを使っている場合には必須、それ以外の場合には禁止）指定するグループの全ページにラベルが適用されます。後続するグループについても、新たなラベルを適用するまではこのラベルが全ページに適用されつづけます。グループ名は、PDF_begin_document( ) の groups オプションで定義してある必要があります。
hypertext-encoding	（キーワード）prefix オプションのエンコーディング。空文字列にすると unicode を意味します。デフォルト：グローバル hypertextencoding オプションの値。

表 3.6 PDF_begin/end_document( ) の labels オプションと PDF_begin/end_page_ext( ) の label オプションのサブオプション

オプション	説明
pagenumber	（整数。PDF_end_document( ) でのみ可。文書がページグループを使っていない場合には必須、それ以外の場合には禁止）指定するページにラベルが適用されます。後続するページについても、新たなラベルを適用するまではこのラベルが適用されつづけます。
prefix	（ハイパーテキスト文字列）範囲内の全ラベルにつけたいラベル接頭辞。デフォルト：なし
start	（整数≧ 1）範囲内の最初のラベルに与えたい数値。範囲内の後続するページにも、この値に続けて番号が振られていきます。デフォルト：1
style	（キーワード）使いたい番号スタイル。デフォルト：none。 *none* ページ番号なし。すなわちラベルの中身は接頭辞だけになります。 *D* 10 進アラビア数字（1, 2, 3, ...） *R* 大文字ローマ数字（I, II, III, ...） *r* 小文字ローマ数字（i, ii, iii, ...） *A* 大文字アルファベット（A, B, C, ..., AA, BB, CC, ...） *a* 小文字アルファベット（a, b, c, ..., aa, bb, cc, ...）

表 3.7 PDF_begin_document( )・PDF_end_document( ) の viewerpreferences オプションのサブオプション

オプション	説明
centerwindow	（論理値）true にすると、文書のウインドウが画面の中央に置かれます。デフォルト：false
direction	（キーワード）文書の閲覧方向。見開き表示でのスクロール順序に対して効力を持ちます（デフォルト：l2r）。 *l2r* 左から右へ *r2l* 右から左へ（縦書きなど）
displaydoctitle	（論理値。PDF/UA-1 モードでは true のみ可）Acrobat のタイトルバーに Title 文書情報フィールドを表示（true）、あるいはファイル名を表示（false）します。デフォルト：PDF/UA-1 では true、それ以外では false
duplex	（キーワード。PDF 1.7）印刷ダイアログの用紙の扱いのオプション（デフォルト：none）。 *DuplexFlipShortEdge* 両面印刷して短辺を綴じる。 *DuplexFlipLongEdge* 両面印刷して長辺を綴じる。 *none* 用紙の扱いを設定しない。 *Simplex* 片面印刷。
enforce	（キーワードのリスト。PDF 2.0）PDF ビューアーのユーザーインターフェイスでオーバーライドされずに強制されるべきビューアー環境設定。以下のキーワードを使えます： *printscaling*（オプション printscaling=none を与えている場合にのみ意味を持ちます）印刷を拡大縮小せず原寸で行うよう強制。
fitwindow	（論理値）文書のウインドウを先頭ページの大きさに合わせるかどうかを指定します。デフォルト：false
hidemenubar	（論理値）ビューアーのメニューバーを隠すかどうかを指定します。デフォルト：false
hidetoolbar	（論理値）ビューアーのツールバーを隠すかどうかを指定します。デフォルト：false

表 3.7 PDF_begin_document( )・PDF_end_document( ) の viewerpreferences オプションのサブオプション

オプション	説明
hidewindowui	（論理値）ビューアーのウインドウコントロールを隠すかどうかを指定します。デフォルト：false
nonfullscreenpagemode	（キーワード。openmode オプションを fullscreen に設定している場合にのみ意味を持ちます）全画面表示から抜けた時の文書の表示方式を指定します（デフォルト：none）： ***bookmarks*** ページとしおりパネルを表示 ***thumbnails*** ページとページパネルを表示 ***layers*** ページとレイヤーパネルを表示 ***none*** ページだけを表示
numcopies	（整数。PDF 1.7）印刷ダイアログの部数。デフォルト：ビューアー依存
picktraybypdfsize	（論理値。PDF 1.7。macOS では効果なし）印刷ダイアログで PDF のページサイズに合わせて給紙トレイを選択するかどうかを指定します。デフォルト：ビューアー依存
printscaling	（キーワード。PDF 1.6）文書で印刷ダイアログを出したときに選択させておきたい、ページの拡縮の選択肢。使えるキーワード（デフォルト：appdefault）： ***none*** ページの拡縮なし。ページの内容を正確な大きさで印刷させたいときに有用です。 ***appdefault*** PDF ビューアーで指定されているカレントの印刷の拡縮を使用します。
printpagerange	（整数ペアのリスト。PDF 1.7）印刷ダイアログのページ番号。各ペアは、印刷したいページ範囲の開始ページと終了ページの番号を表します（先頭ページが 1）。デフォルト：ビューアー依存

3.2 PDF 文書をメモリーから取得

空でない *filename* 引数が *PDF_begin_document( )* に与えられているとき、PDFlib は名前付きディスクファイルに PDF 文書を書き込みます。あるいは、*filename* 引数が空であれば、PDF 文書はメモリー内に生成されます。この場合、PDF 文書データはメモリーから *PDF_get_buffer( )* で取得する必要があります。これはとりわけ、PDF を Web サーバーから頒布するときに有用です。

C++ ***const char *get_buffer(long *size)***
Java ***final byte[] get_buffer( )***
Perl PHP ***string get_buffer( )***
C ***const char * PDF_get_buffer(PDF *p, long *size)***

PDF 出力バッファの内容を取得します。

size （C・C++・RPG 言語バインディングのみ）返されたデータの長さをバイト単位で表したものを格納させたいメモリー位置への C スタイルのポインター。

戻り値 クライアント側で消費するためのバイナリー PDF データで満たされたバッファ。このメソッドは、バイナリーデータのための言語独自のデータ型を返します。返されたバッファは、クライアントで他の何らかの PDFlib メソッドを呼び出す前に使う必要があります。

詳細 生成された PDF データの入ったバッファ全体ないし一部分を取り出します。このメソッドがページ記述の合間に呼び出されたときは、それまでに生成された PDF データを返します。PDF をメモリー内に生成している場合は、このメソッドは *PDF_end_document( )* の後には少なくとも呼び出す必要があり、そのときに PDF 文書の残りを返します。もっと前に呼び出して文書データを部分的に取り出してもよいでしょう。このメソッドを 1 回だけ *PDF_end_document( )* の後に呼び出す場合は、返されるバッファにはその PDF 文書がまるごとまとめて入っていることが保証されます。

PDF 出力にはバイナリーキャラクターが含まれていますので、クライアントソフトウェア側では、null 値を含む印刷不可能キャラクターを受け入れる態勢が必要です。

スコープ **オブジェクト・文書**（言い換えれば、*PDF_end_page_ext( )* から *PDF_begin_page_ext( )* までの間か、または *PDF_end_document( )* から *PDF_delete( )* までの間）。このメソッドは、*PDF_begin_document( )* に空の *filename* を与えているときだけ使えます。

PDF_begin_document( ) で *linearize* オプションを *true* に設定しているときは、スコープは *object* に限られますので、すなわちこのメソッドは *PDF_end_document( )* の後でしか呼び出せません。

バインディング C・C++：*size* 引数は C・C++ クライアントでだけ使えます。

それ以外のバインディング：適切な長さのオブジェクトが返されますので、*size* 引数は与えてはいけません。

3.3 ページメソッド

C++ Java C# ***void begin_page_ext(double width, double height, String optlist)***

Perl PHP ***begin_page_ext(float width, float height, string optlist)***

C ***void PDF_begin_page_ext(PDF *p, double width, double height, const char *optlist)***

文書に新規ページを追加して、さまざまなオプションを指定します。

width・height 引数 *width* と *height* は、新規ページの寸法をポイント単位（*userunit* オプションを指定してあるときはユーザー単位）で指定します。これらは同名のオプションでオーバーライドできます（その場合、引数にはダミーの値 0 を使えばよいでしょう）。広く利用されるページ判型の一覧を表 3.8 に示します。詳しくは表 3.9 も参照してください（*width・height* オプション）。

表 3.8 代表的な ISO 標準ページサイズの寸法をポイント単位で表したもの[1]

判型	幅	高さ	判型	幅	高さ	判型	幅	高さ
a0	2380	3368	a4	595	842	letter	612	792
a1	1684	2380	a5	421	595	legal	612	1008
a2	1190	1684	a6	297	421	ledger	1224	792
a3	842	1190	b5	501	709	11x17	792	1224

1. ISO B5 は JIS B5 と異なることに留意してください。ISO・日本・米国の標準形式に関する詳しい情報は次の URL を参照してください。*www.cl.cam.ac.uk/~mgk25/iso-paper.html*

optlist 表 3.9 に従ったページオプション群を持ったオプションリスト。これらのオプションは、*PDF_end_page_ext()* で指定する同等のオプションよりも優先順位が低いです：*action*・*artbox*・*associatedfiles*・*bleedbox*・*blocks*・*cropbox*・*defaultcmyk*・*defaultgray*・*defaultrgb*・*duration*・*group*・*height*・*label*・*mediabox*・*metadata*・*outputintents*・*pagenumber*・*rotate*・*separationinfo*・*taborder*・*topdown*・*transition*・*transparencygroup*・*trimbox*・*userunit*・*viewports*・*width*

詳細 このメソッドは、すべてのテキスト・グラフィック・色ステータスパラメーターをそのデフォルト値にリセットし、*topdown* オプションに従って座標系を確立します。

PDF/A *transparencygroup* オプションに制約が課せられます。

PDF/UA *taborder* オプションに制約が課せられます。

PDF/VT 以下のオプションは許されません：*group*・*pagenumber*。

PDF/X *transparencygroup*・*defaultgray/rgb/cmyk* オプションに制約が課せられます。

スコープ **文書**。このメソッドは**ページ**スコープを開始させます。対応する *PDF_end_page_ext()* と必ずペアにして呼び出す必要があります。

C++ Java C# **void end_page_ext(String optlist)**

Perl PHP **end_page_ext(string optlist)**

C **void PDF_end_page_ext(PDF *p, const char *optlist)**

ページを終了させて、さまざまなオプションを適用します。

optlist 表 3.9 に従ったオプションリスト。*PDF_end_page_ext()* で指定するオプションは、*PDF_begin_page_ext()* で指定した同等のオプションよりも優先されます。以下のオプションが使えます：
associatedfiles・*action*・*artbox*・*bleedbox*・*cropbox*・*defaultcmyk*・*defaultgray*・*defaultrgb*・*duration*・*group*・*height*・*label*・*mediabox*・*metadata*・*rotate*・*taborder*・*transition*・*transparencygroup*・*trimbox*・*userunit*・*viewports*・*width*

スコープ **ページ**。このメソッドは**ページ**スコープを終了させます。対応する *PDF_begin_page_ext()* と必ずペアにして呼び出す必要があります。

タグ付き PDF モードでは、すべての直接・擬似アイテムは、この関数を呼び出す前に閉じる必要があります。

表 3.9 PDF_begin_page_ext( )・PDF_end_page_ext( ) のオプション

オプション	説明
action	（アクションリスト。PDF/A では不可）以下の 1 個ないし複数のイベントに対するページアクションのリスト（デフォルト：空リスト）： ***open*** ページを開く時に実行させたいアクション。 ***close*** ページを閉じる時に実行させたいアクション。 PDF_end_page_ext( ) の action オプションにアクションを与えると、PDF_fit_pdi_page( ) で useactions オプションを用いて PDF ページから取り込んだアクションを個別にオーバーライドします。 target オプションを用いて作成しておいた表現アクションは、そのターゲット注釈が配置されているページにおいてのみ許されます。
associatedfiles	（アセットハンドルのリスト。PDF 2.0・PDF/A-3 でのみ可）ページに紐付けるファイル群のアセットハンドル群。このファイル群は、PDF_load_asset( ) で type=attachment を用いて読み込んである必要があります。
artbox ***bleedbox*** ***cropbox***	（長方形）カレントページのページ枠のパラメーターを変更します。座標はデフォルト座標系で指定します。デフォルト：枠項目なし
blocks	（POCA コンテナハンドル。PDF_begin_page_ext( ) か PDF_end_page_ext( ) に与えることができますが、同一ページ上で両方のメソッドに与えることはできません。PPS でのみ利用可能）PDF_poca_new( ) で作成された、PDFlib Personalization Server（PPS）のための PDFlib ブロック定義群を内容とする辞書コンテナに対するハンドル。この指定されたブロック群がページに添付されます。この辞書は、オプション usage=blocks を用いて作成してある必要があります。デフォルト：ブロックなし
defaultgray[1] ***defaultrgb***[1] ***defaultcmyk***[1]	（ICC ハンドルかキーワード）与える ICC プロファイルハンドルに従って、ページにデフォルトのグレー・RGB・CMYK 色空間を設定します。 オプション defaultrgb ではキーワード srgb も使えます。この指定された色空間が、ページ上のデバイス依存グレーか RGB か CMYK カラーをマップするために使用されます。
duration	（float）openmode=fullscreen にしているとき（表 3.1 参照）、カレントページにページ表示時間を秒単位で設定します。デフォルト：1

表 3.9 PDF_begin_page_ext( ) ・ PDF_end_page_ext( ) のオプション

オプション	説明
group[1]	（文字列。文書がページグループを使っている場合には必須、それ以外の場合には不可。PDF/VT モードの場合、または文書部分ヒエラルキーが作成されている場合には不可）ページを属させたいページグループの名前。この名前を使うと、複数のページを 1 つのページグループにまとめて、ページを PDF_resume_page( ) で指定できるようになります。このグループ名は、PDF_begin_document( ) で groups オプションを使って定義してある必要があります。
height	（float または キーワード。PDF_end_page_ext( ) では、topdown オプションが true なら禁止）新規ページの寸法をポイント単位（userunit オプションを指定してあるときはユーザー単位）で指定します。横置きのページを作るには、width ＞ height とするか、または rotate オプションを使います。PDFlib は width と height を使ってページの MediaBox を作りますが、MediaBox は明示的に mediabox オプションを使って設定することもできます。width ・ height オプションは、同名の引数をオーバーライドします。 以下のシンボリックな ISO ページサイズ名の後に .width か .height を付けてキーワードとして使うこともできます（例 : a4.width ・ a4.height）。 a0 ・ a1 ・ a2 ・ a3 ・ a4 ・ a5 ・ a6 ・ b5 ・ letter ・ legal ・ ledger ・ 11x17
label	（オプションリスト）シンボリックなページ名を表 3.6 に従って指定したオプションリスト。このページ名は、Acrobat のステータスバーにページラベルとして（ページ番号のかわりに）表示されます。指定する付番方式はカレントページに使われるほか、後続するページについても、また変更するまではこの方式が使われつづけます。style ・ prefix ・ start 値の組み合わせが文書内で一意である必要があります。
mediabox	（長方形。topdown オプションが true なら禁止）カレントページの MediaBox を変更します。座標はデフォルト座標系で指定します。デフォルトでは、MediaBox は引数 width ・ height を使って作成されます。この mediabox オプションは、width ・ height オプションおよび引数をオーバーライドします。
metadata	（オプションリスト）そのページに対するメタデータ（307 ページ「15.2 XMP メタデータ」参照）
outputintents	（出力インテントハンドル。PDF 2.0）ページに対する出力インテントとして使いたい ICC プロファイル。このハンドルは、PDF_load_iccprofile( ) で usage=pageoutputintent か usage=outputintent を用いて、あるいは PDF_process_pdi( ) で action=copypageoutputintent を用いて取得している必要があります。
pagenumber[1]	（整数。PDF/VT モードの場合、または文書部分ヒエラルキーが作成されている場合には不可）このオプションで値 n を指定したとすると、ページは、group オプションで指定しているページグループ（文書がページグループを使っていないなら文書）の既存の n 番目のページの前に挿入されます。このオプションを指定しないと、ページはグループの末尾に挿入されます。
rotate	（整数）ページの回転値。この回転は、ページ表示に対して効力を持ちますが、座標系は変更しません。とりうる値は 0 ・ 90 ・ 180 ・ 270。デフォルト : 0
separation-info[1]	（オプションリスト。PDF 2.0 では廃止されており利用もできません）カレントページに対する色分版の詳細 : ***pages*** （整数。各色版ページセットの先頭ページには必須、同じセット内のそれ以降のページでは不可）多色ページ一枚の色データを成す各色版ページ群のセット 1 つに属するページの数。セットのページはすべてファイル内で連続させておく必要があります。 ***spotname*** （文字列。spotcolor を与えていないなら必須）カレントページのためのインキの名前。 ***spotcolor*** （スポットカラーハンドル）カレントページのためのインキを記述したカラーハンドル。

表 3.9 PDF_begin_page_ext( ) ・ PDF_end_page_ext( ) のオプション

オプション	説明
taborder	（キーワード。PDF 1.5。PDF/UA-1 では structure のみ可）ページ上のフォームフィールドと注釈にタブ順序を指定するキーワード（デフォルト：PDF 1.5 以上に対するタグ付き PDF モードでは structure、それ以外では none）。 ***annotations*** （PDF 2.0）タブ順序は注釈の順序によって定義されます。 ***column*** 段組みごとに上から下へ。段組みの順序は、PDF_begin/end_document( ) で viewerpreferences オプションの direction サブオプションで指定した通りになります。 ***none*** タブ順序を指定しません。 ***structure*** フォームフィールドと注釈が、構造ツリー内での出現順に選ばれていきます。 ***row*** 上の行から下の行へ。行内での方向は、PDF_begin/end_document( ) で viewerpreferences オプションの direction サブオプションで指定した通りになります。 ***widgets*** （PDF 2.0）タブ順序はフォームフィールドの順序によって、次いでその他の注釈の順序によって定義されます。
topdown[1]	（論理値）true にすると、ページが始まる時の座標系は、ページの左上隅が原点で、y 座標が下向きに増加します。そうでなければデフォルト座標系が使われます。デフォルト：false
transition	（キーワード）openmode=fullscreen にしているとき（表 3.1 参照）、カレントページにページ効果を設定して特殊な表示遷移をさせます。これは、PDF を Acrobat で全画面表示してプレゼンテーションを行う時に有用でしょう。デフォルト：replace ***split*** 画面上に線が 2 本走ってページが見えてくる ***blinds*** 画面上に線が多数走ってページが見えてくる ***box*** 長方形からページが見えてくる ***wipe*** 画面上に線が 1 本走ってページが見えてくる ***dissolve*** 元ページが溶けてページが見えてくる ***glitter*** 溶け効果が画面の一辺から他辺へ動いていく ***replace*** 単純に元ページから新ページに切り替わる ***fly*** （PDF 1.5）元ページの中へ新ページが飛んでくる ***push*** （PDF 1.5）元ページを新ページが画面外へ押し出していく ***cover*** （PDF 1.5）元ページの上に新ページがすべり込んでくる ***uncover*** （PDF 1.5）元ページが画面外へすべり出て新ページが見えてくる ***fade*** （PDF 1.5）元ページから新ページが透けてくる

表 3.9 PDF_begin_page_ext( ) ・ PDF_end_page_ext( ) のオプション

オプション	説明
trans-parency-group	（オプションリストまたはキーワード。PDF/A-1・PDF/X-3 では不可。PDF/A-2/3・PDF/X-4/5 では制約が課されます）そのページに対する透過グループを作成します。以下のキーワードを使えます（デフォルト：auto）： ***auto*** 透過オブジェクトが、そのページ自体に、または配置されたテンプレートか画像に、または取り込みグラフィック上に（分離された透過グループは、そのグループの外にあるオブジェクトに影響を与えないので、勘定に入れません）、または注釈上に存在している場合には、ページ透過グループが、然るべき色空間を用いて作成されます。そうでないなら透過グループは全く作成されません。 ***none*** （出力インテントを持たない PDF/A-2/3 に対して、および、PDF/X-5n に対しては、透過がそのページ上で用いられているならば不可）そのページに対して透過グループを作成しません。 以下のサブオプションを用いて明示的に透過グループを作成することもできます： ***colorspace*** （キーワードまたは ICC プロファイルハンドル。出力インテントを持たない PDF/A-2/3 にに対して、および、PDF/X-5n に対しては、透過がそのページ上で用いられているならば必須）ページに対するブレンディング色空間（デフォルト：none）： ***DeviceCMYK*** PDF/A-2/3・PDF/X-4：CMYK 出力インテントを用いるか、defaultcmyk オプションが与えられている場合にのみ可。 PDF/X-5n：出力インテントがインキ Cyan・Magenta・Yellow・Black を含んでいるか、defaultcmyk オプションが与えられている場合のみ可。 ***DeviceGray*** PDF/A-2/3：グレー・RGB・CMYK 出力インテントを用いるか、defaultgray オプションが与えられている場合にのみ可。PDF/X-4：グレーか CMYK 出力インテントを用いるか、defaultgray オプションが与えられている場合のみ可。 PDF/X-5n：出力インテントがインキ Black を含んでいるか、defaultgray オプションが与えられている場合にのみ可。 ***DeviceRGB*** PDF/A-2/3・PDF/X-4：RGB 出力インテントを用いるか、defaultrgb オプションが与えられている場合にのみ可。 PDF/X-5n：defaultrgb オプションが与えられている場合にのみ可。 ***none*** （出力インテントを持たない PDF/A-2/3 に対して、および、PDF/X-5n に対しては、透過がそのページ上で用いられているならば不可）透過グループに対して色空間が出力されません。 ***srgb*** sRGB 色空間を選択するためのキーワード ***knockout*** （論理値）ページグループが抜きグループかどうかを指定します。抜きグループとは、グループの要素群が互いに複合しないことを意味します。すなわち、オブジェクトが、このグループ内でそれより早いエレメントを抜きます。デフォルト：false
trimbox	（長方形）カレントページの TrimBox を指定します。座標はデフォルト座標系で指定します。デフォルト：TrimBox 項目なし
userunit	（float またはキーワード。PDF 1.6）ユーザー単位の大きさをポイント単位で表した、範囲 1 ～ 75 000 の数。またはキーワード mm・cm・m のいずれかを指定すると、その単位に拡大されます。ユーザー単位は、実際のページ内容を変えません。それは Acrobat に対するヒントでしかなく、ページを印刷したり、計測ツールを使ったりするときに利用されるだけです。デフォルト：1（すなわち 1 単位は 1 ポイント）
viewports	（オプションリストのリスト。PDF 1.7ext3）ページ上の 1 個ないし複数の地理参照付き領域（ビューポート）。詳しくは 290 ページ「13.4 地理空間機能」を参照。 ビューポートは、ページ上の複数の領域（複数の地図など）ごとに、異なる地理空間参照（georeference オプションで指定される）を用いることを可能にします。ビューポートリスト内のオプションリスト群の順序は、重なり合うビューポート群に対応します。すなわち、ある点を含む最後のビューポートが、その点に対して用いられます。

表 3.9 PDF_begin_page_ext( ) ・ PDF_end_page_ext( ) のオプション

オプション	説明
width	(float または キーワード。PDF_end_page_ext( ) では、topdown オプションが true なら禁止) height オプションを参照してください。

1. PDF_begin_page_ext( ) でのみ可

C++ Java C# ***void suspend_page(String optlist)***

Perl PHP ***suspend_page(string optlist)***

C ***void PDF_suspend_page(PDF *p, const char *optlist)***

カレントページを一時停止して、後で再開できるようにします。

optlist 将来使用するためのオプションリスト。

詳細 カレントページのグラフィック・色・テキスト・レイヤーステートがまるごと、内部的に保存されます。このページは、後で *PDF_resume_page( )* で再開してまた内容を追加することができます。一時停止したページは、再開しなければ閉じることができません。

スコープ **ページ**。このメソッドは**文書**スコープを開始させます。対応する *PDF_resume_page( )* と必ずペアにして呼び出す必要があります。タグ付き PDF モードでは、すべての直接・擬似アイテムを、このメソッドを呼び出す前に閉じる必要があります。

C++ Java C# ***void resume_page(String optlist)***

Perl PHP ***resume_page(string optlist)***

C ***void PDF_resume_page(PDF *p, const char *optlist)***

ページを再開して、またそれに内容を追加できるようにします。

optlist 表 3.10 に従ったオプションリスト。以下のオプションが使えます：
group ・ *pagenumber*

詳細 ページは、*PDF_suspend_page( )* で一時停止してある必要があります。これが再び開かれて、また内容を追加できるようになります。一時停止したページはすべて、たとえもう内容を追加していなくても、再開しなければ閉じることができません。

タグ付き PDF モードでは、ページを再開しても、構造アイテムは再開されないことに留意する必要があります。そうではなく、*PDF_resume_page( )* が呼び出された時点でアクティブなアイテムは、それ以後のページコンテンツに対するカレントアイテムになります。以後に生成されるコンテンツに対する親としてページ上の特定の構造エレメントを再開させるには、*PDF_activate_item( )* を使用することを推奨します。

スコープ **文書**。このメソッドは**ページ**スコープを開始させます。対応する *PDF_suspend_page( )* と必ずペアにして呼び出す必要があります。

表 3.10 PDF_resume_page( ) のオプション

オプション	説明
group	(文字列。文書がページグループを使っている場合には必須、そうでない場合には禁止) 再開されるページのページグループの名前。このグループ名は、PDF_begin_document( ) で groups オプションを使って定義してある必要があります。

表 3.10 PDF_resume_page( ) のオプション

オプション	説明
pagenumber	（整数）このオプションを与えると、group オプションで選んでいるページグループ（文書がページグループを使っていないなら文書）の、指定した番号のページが再開されます。このオプションを与えないと、グループの最終ページが再開されます。

3.4 レイヤー

C++ Java C# ***int define_layer(String name, String optlist)***

Perl PHP ***int define_layer(string name, string optlist)***

C ***int PDF_define_layer(PDF *p, const char *name, int len, const char *optlist)***

新規レイヤー定義を作成します。

name （ハイパーテキスト文字列）レイヤーの名前。

len （C 言語バインディングのみ）*name* の長さ（バイト単位）。*len=0* にすると null 終了文字列を与える必要があります。

optlist レイヤー設定群を持ったオプションリスト：

- 表 3.11 に従ったレイヤー制御オプション：
  creatorinfo・*defaultstate*・*initialexportstate*・*initialprintstate*・*initialviewstate*・*intent*・*language*・*onpanel*・*pageelement*・*printsubtype*・*removeunused*・*zoom*
- C・Perl・Ruby で *stringformat=legacy* の場合のエンコーディングオプション：
  hypertextencoding・*hypertextformat*（表 2.1 参照）

戻り値 レイヤーハンドル。*PDF_begin_layer()*・*PDF_set_layer_dependency()* への呼び出しで、カレントの**文書**スコープを終えるまで使えます。

詳細 レイヤーを定義したにもかかわらず、文書で使っていないときは、PDFlib は警告を発します。複数のページで使うレイヤーであっても、その定義は 1 回だけ行うべきです（たとえば最初のページを作成する前に）。*PDF_define_layer()* を複数のページで繰り返し呼び出すと、レイヤーの定義は蓄積されてしまい（たとえ名前が同じでも）、それは通常望ましくないでしょう。

このメソッドを呼び出した後に、*PDF_open_pdi_document()* をレイヤー付き PDF 文書に対してオプション *uselayers=false* で呼び出してはいけません。逆も同様に、*PDF_open_pdi_document()* をレイヤー付き PDF 文書に対してオプション *uselayers=false* で呼び出した後に、このメソッドを呼び出してはいけません。

PDF/A PDF/A-2/3：いくつかのオプションは制約されます。

PDF/UA いくつかのオプションは制約されます。

PDF/X PDF/X-4/5: いくつかのオプションは制約されます。

スコープ **オブジェクト**以外任意。要 PDF 1.5

表 3.11 PDF_define_layer( ) のオプション

オプション	説明
creatorinfo	（オプションリスト。PDF/A-2/3・PDF/X-4/5・PDF/UA-1 では不可）内容と作成アプリケーションを記述したオプションリスト。このオプションを使うときは、以下の両方の項目が必須です： ***creator*** （ハイパーテキスト文字列）レイヤーを作成したアプリケーションの名前 ***subtype*** （文字列）内容の種類。推奨値は `Artwork`・`Technical`。
defaultstate	（論理値）レイヤーをデフォルトで表示するかどうかを指定します。デフォルト：`true`

表 3.11 PDF_define_layer( ) のオプション

オプション	説明
initial-exportstate	（論理値。PDF/A-2/3・PDF/X-4/5・PDF/UA-1 では不可）レイヤーの推奨書き出しステータス。true にすると、Acrobat は、以前の PDF バージョンや他の文書形式へ変換・書き出しを行う際、このレイヤーを含めます。デフォルト：true
initial-printstate	（論理値。PDF/A-2/3・PDF/X-4/5・PDF/UA-1 では不可）レイヤーの推奨印刷ステータス。true にすると、Acrobat は、文書を印刷する際、このレイヤーを含めます。デフォルト：true
initial-viewstate	（論理値。PDF/A-2/3・PDF/X-4/5・PDF/UA-1 では不可）レイヤーの推奨表示ステータス。true にすると、Acrobat は、文書を開いた際、このレイヤーを表示します。デフォルト：true
intent	（キーワード）グラフィックの想定用途。View または Design。デフォルト：View
language	（オプションリスト。PDF/A-2/3・PDF/X-4/5・PDF/UA-1 では不可）レイヤーの言語： ***lang*** （文字列。必須）言語を、場合によってはロケールとあわせて、表 3.1 の lang オプションについて説明した形式で指定します ***preferred*** （論理値）true にすると、レイヤーは、レイヤーとシステムの言語が部分的に一致するだけでも使われます。デフォルト：false
onpanel	（論理値。PDF/A-2/3・PDF/X-4/5・PDF/UA-1 では不可）false にすると、レイヤーは Acrobat のレイヤーパネルに表示されず、したがってユーザーが操作できなくなります。デフォルト：true
pageelement	（キーワード。PDF/A-2/3・PDF/X-4/5・PDF/UA-1 では不可）レイヤーがページネーションのためのページ装飾を含んでいることを示します。HF（ヘッダー / フッター）・FG（前面の画像またはグラフィック）・BG（背景の画像またはグラフィック）・L（ロゴ）のいずれか 1 つ。
printsubtype	（オプションリスト。PDF/A-2/3・PDF/X-4/5・PDF/UA-1 では不可）レイヤーが印刷を想定しているかどうかを指定します： ***subtype*** （キーワード）レイヤーの内容の種類を表す、Trapping・PrintersMarks・Watermark のいずれか 1 つ。 ***printstate*** （論理値）true にすると、Acrobat は印刷の際にこのレイヤーの内容を可視化します。
removeunused	（論理値）true にすると、レイヤーがページ上で使われていないときは、このレイヤーはそのページのレイヤー一覧には表示されなくなります。レイヤーがページ上で使われていると見なされるのは、そのレイヤーがそのページ上で少なくとも 1 回、PDF_begin_layer( ) に与えられているときです。デフォルト：false。
zoom	（float かパーセント値のリスト。PDF/A-2/3・PDF/X-4/5・PDF/UA-1 では不可）表示倍率によるレイヤーの表示切り替えを指定する、1 個か 2 個の値（1.0 は表示倍率 100 パーセントを意味します）。値 1 個を与えると、レイヤーの表示される最大の表示倍率として使われます。値 2 個を与えると、最小と最大の表示倍率を指定します。キーワード maxzoom を使うと、可能な限り最大の表示倍率を指定することができます。

C++ Java C# ***void set_layer_dependency(String type, String optlist)***

Perl PHP ***set_layer_dependency(string type, string optlist)***

C ***void PDF_set_layer_dependency(PDF *p, const char *type, const char *optlist)***

レイヤー間の関係を定義します。

type 表 3.12 に従った、依存または関係の種別。

optlist レイヤー依存群のためのオプションリスト：

- 表 3.13 に従ったレイヤー依存オプション：
  children・*depend*・*group*・*parent*

表 3.12 レイヤーの依存と関係の種別

種別	説明。この種別に関連するオプション
GroupAllOn	depend オプションで指定するレイヤーは、group オプションで指定するレイヤーがすべて表示されているときに表示されます。この種別に関連するオプション：depend・group
GroupAnyOn	depend オプションで指定するレイヤーは、group オプションで指定するレイヤーのいずれかが表示されているときに表示されます。この種別に関連するオプション：depend・group
GroupAllOff	depend オプションで指定するレイヤーは、group オプションで指定するレイヤーがすべて隠されているときに表示されます。この種別に関連するオプション：depend・group
GroupAnyOff	depend オプションで指定するレイヤーは、group オプションで指定するレイヤーのいずれかが隠されているときに表示されます。この種別に関連するオプション：depend・group
Lock	（PDF 1.6）group オプションで指定するレイヤーはロックされます。すなわち、レイヤーのステータスを Acrobat で対話的に変更できません。この種別に関連するオプション：group
Parent	parent オプションで指定するレイヤーと、children オプションで指定するレイヤー群との間に、階層関係を指定します。親を不可視に設定するとその子群も自動的に不可視に設定されます。1 つのレイヤーを、複数のレイヤーに属させることはできません。この種別に関連するオプション：children・parent
Radiobtn	group オプションで指定するレイヤーどうしの間に、ラジオボタン関係を指定します。すなわち、そのグループのレイヤーは、同時に 1 つしか表示されなくなります。これはとりわけ、複数の言語レイヤーで有用です。この種別に関連するオプション：group
Title	parent オプションで指定するレイヤーは、直接制御するページ内容を一切持たず、children オプションで指定するレイヤー群に対する階層区切りとしての役目を持ちます。この種別に関連するオプション：children・parent

- C・Perl・PHP で *stringformat=legacy* の場合のエンコーディングオプション：*hypertextencoding*（表 2.1 参照）

詳細 レイヤー関係は、Acrobat のレイヤーペーン内でのレイヤー名の表示を指定するとともに、ユーザーが対話的にレイヤーの表示・非表示を切り替えた時の 1 個ないし複数のレイヤーの表示・非表示をも指定します。

スコープ **オブジェクト**以外任意。レイヤー関係は、すべてのレイヤーが定義された後に指定する必要があります。要 PDF 1.5

表 3.13 PDF_set_layer_dependency( ) のオプション

オプション	説明
children	（レイヤーハンドルのリスト。type=Parent・Title でのみ可）与えている親レイヤーの子にしたいレイヤー群を指定する 1 個ないし複数のレイヤーハンドル。
depend	（レイヤーハンドル。type=GroupAllOn・GroupAnyOn・GroupAllOff・GroupAnyOff でのみ可）group オプションで指定しているレイヤー群に制御させたいレイヤー。
group	（レイヤーハンドルのリスト。type=GroupAllOn・GroupAnyOn・GroupAllOff・GroupAnyOff・Lock・Radiobtn でのみ可）グループを構成させたい 1 個ないし複数のレイヤーハンドル。type=Lock にしているときは、グループのレイヤーはすべてロックされます。
parent	（レイヤーハンドル。type=Parent・Title でのみ可）children オプションで指定しているレイヤー群の親にしたいレイヤー。

C++ Java C# ***void begin_layer(int layer)***

Perl PHP ***begin_layer(int layer)***

C ***void PDF_begin_layer(PDF *p, int layer)***

ページ上の以後の出力に対して、レイヤーを開始します。

layer レイヤーのハンドル。*PDF_define_layer()* で取得しておく必要があります。

詳細 この呼び出しの後、次に *PDF_begin_layer()* か *PDF_end_layer()* を呼び出すまでにページ上に配置する内容はすべて、指定するレイヤーの一部分になります。内容が表示されるかどうかは、レイヤーの設定に依存します。

このメソッドは、指定されたレイヤーを活性化し、その時点で活性なレイヤーがもしあればそれを不活性化します。

注釈・画像・グラフィック・テンプレート・フォームフィールドのレイヤーは、それぞれのメソッドで ***layer*** オプションを使って制御することができます。

スコープ **ページ**。要 PDF 1.5

C++ Java C# ***void end_layer()***

Perl PHP ***end_layer()***

C ***void PDF_end_layer(PDF *p)***

すべての活性なレイヤーを不活性化します。

詳細 この呼び出しの後、ページ上に配置する内容は、どのレイヤーにも属しません。レイヤーはすべて、ページの終わりには閉じる必要があります。

レイヤー A からレイヤー B へ切り換えるには、*PDF_begin_layer()* を 1 回呼び出せば充分です。明示的に *PDF_end_layer()* を呼び出してレイヤー A を閉じる必要はありません。*PDF_end_layer()* が必要なのは、無条件内容（つねに表示される）を作成するときと、ページの終わりに全レイヤーを閉じるときだけです。

スコープ **ページ**。要 PDF 1.5

4 フォント・テキストメソッド

この章の API メソッド：

- *PDF_load_font()*
- *PDF_close_font()*
- *PDF_info_font()*
- *PDF_set_text_option()*
- *PDF_setfont()*
- *PDF_set_text_pos()*
- *PDF_show()*
- *PDF_show_xy()*
- *PDF_continue_text()*
- *PDF_stringwidth()*
- *PDF_begin_font()*
- *PDF_end_font()*
- *PDF_begin_glyph_ext()*
- *PDF_end_glyph()*
- *PDF_encoding_set_char()*

4.1 フォント処理

C++ Java C# ***int load_font(String fontname, String encoding, String optlist)***

Perl PHP ***int load_font(string fontname, string encoding, string optlist)***

C ***int PDF_load_font(PDF *p, const char *fontname, int len, const char *encoding, const char *optlist)***

フォントを検索して、以後の利用に備えます。

fontname （名前文字列）フォントの名前。これはあるいは、この引数をオーバーライドする *fontname* オプションで与えることもできます。詳しくは表 4.1 の *fontname* オプションを参照。

len （C 言語バインディングのみ）*filename* の長さをバイト単位で指定します。*len=0* にすると null 終了文字列を与える必要があります。

encoding エンコーディングの名前。これはあるいは、この引数をオーバーライドする *encoding* オプションで与えることもできます。詳しくは表 4.1 の *encoding* オプションを参照。

optlist 以下のオプションによるオプションリスト：

- 一般オプション：*errorpolicy*（表 1.5 参照）
- 表 4.1 に従ったフォント読み込みオプション：
  ascender・*autosubsetting*・*capheight*・*colormode*・*descender*・*embedding*・*encoding*・*fallbackfonts*・*fontname*・*fontstyle*・*foregroundcolor*・*foregroundopacity*・*initialsubset*・*keepfont*・*keepnative*・*linegap*・*metadata*・*optimizeinvisible*・*preservepua*・*readfeatures*・*readkerning*・*readselectors*・*readshaping*・

readverticalmetrics・*replacementchar*・*simplefont*・*skipembedding*・*strikeselect*・*subsetlimit*・*subsetminsize*・*subsetting*・*unicodemap*・*vertical*・*xheight*

戻り値 フォントハンドル。以後の *PDF_info_font()* とテキスト出力メソッドと、テキスト書式オプション *font* で使えます。要求されたフォントとエンコーディングの組み合わせが、設定の問題（フォントが見つからなかったなど）が原因で読み込めないときは、エラーコード -1（PHP では 0）が返されるか例外が発生します。エラー動作は、*errorpolicy* オプションで変えることができます。

メソッドがエラーを返したときは、その失敗の原因を *PDF_get_errmsg()* で取得することができます。そうでないときは、このメソッドによって返された値を、他のフォント関連のメソッドを呼び出す時にフォントハンドルとして用いることができます。返されるハンドルは、フォントハンドルとして使える以外に、ユーザーにとって何の意味も持ちません。

返されたフォントハンドルは、そのフォントを *PDF_close_font()* で閉じるまで有効です。文書を *PDF_end_document()* で完了させると、それぞれの開かれているフォントハンドルは閉じられますが、ただし *keepfont* オプションを与えていた場合、またはフォントを**オブジェクト**スコープで（すなわちあらゆる文書の外で）読み込んでいた場合はその限りではありません。

詳細 このメソッドはフォントを用意し、以後使えるようにします。

繰り返し呼び出し：このメソッドを、同じフォント名かつ同じエンコーディングで再度呼び出すと、最初の呼び出しと同じフォントハンドルが返されます。最初の呼び出しで *embedding=false* とし、2 度目の呼び出しで *embedding= true* とすると、フォントの再度読み込みの試みは失敗します。

暗黙的フォント読み込み：明示的に *PDF_load_font()* でフォントを読み込むほかに、API メソッドのなかには、オプションリストでフォント名を指定することで暗黙的にフォントを読み込めるものがあります（*PDF_add/create_textflow()*・*PDF_fill_textblock()* 等）。その時点までにそのフォントがすでに読み込まれていなければ、新規フォントハンドルが作成されます。

Unicode 非対応言語バインディングでは、オプション `textformat=auto` は以下のように動作します（どちらの場合もすべての UTF 形式が許容されます）：

- ワイドキャラクターエンコーディング群：読み込まれるフォントのテキストはテキスト形式 *utf16* と見なされます。
- バイト・マルチバイトエンコーディング群：読み込まれるフォントのテキストはテキスト形式 *bytes* と見なされます。

PDF/A すべてのフォントを埋め込む必要があります。

PDF/UA すべてのフォントを埋め込む必要があります。

PDF/X すべてのフォントを埋め込む必要があります。

スコープ 任意

表 4.1 PDF_load_font( ) と、オプションリストを用いた暗黙的フォント読み込みのフォントオプション

オプション	説明
ascender	（-2048 から 2048 までの整数）アセンダー。同名のタイポグラフィー特性が強制的に指定値になります。これは、フォント内に値が見つかったときはオーバーライドされますが、フォントにそのような情報が入っていないときに特に有用です（Type 3 フォント等）。デフォルト：あるならフォント内の値、ないなら推算値（PDF_info_font( ) で取得できます）

表 4.1 PDF_load_font( ) と、オプションリストを用いた暗黙的フォント読み込みのフォントオプション

オプション	説明
auto-subsetting	（論理値。subsetting オプションを与えている場合には無視されます）フォントをサブセット化するかを、subsetlimit・subsetminsize オプションと、グリフの実際の使われ方に従って自動的に決定します。デフォルト：true
capheight	（−2048 から 2048 までの整数）キャップハイト。上記 ascender 参照。
colormode	（キーワード。OpenType カラーフォントに対してのみ意味を持ちます）OpenType カラーフォントに対する処理モード（詳しくは PDFlib チュートリアルを参照）。フォントのカラーモードを変えることはできません。すなわち、フォントが再度読み込まれた場合には、colormode オプションは無視され、最初の読み込み操作のカラーモードが使用されます（デフォルト：combined）： ***ignorecolor*** カラーグリフ群を無視してモノクログリフ群だけから Type 3 フォントを生成。白黒ワークフローの場合に推奨します。しかし、輪郭のないダミーのモノグリフ群だけを含んでいるカラーフォント（とくに sbix フォント）もありますので、それを ignorecolor を用いて使うことはできません。 ***ignoremono*** モノクログリフ群を無視してカラーグリフ群だけから Type 3 フォントを生成。他のフォントのためのフォールバックとしてそのカラーフォントを使うことを意図している場合に推奨します。OpenType 機能・合字・シェーピングはこのモードでは利用できません。 ***combined*** モノクロとカラーのグリフ群から Type 3 フォントを生成。できるフォントの中のグリフ ID 群は、元の OpenType フォントの中のものと同一になります。
descender	（−2048 から 2048 までの整数）ディセンダー。上記 ascender 参照。
embedding	（論理値。PDF/A・PDF/UA・PDF/X では true にする必要があります）フォントを埋め込むかを制御。フォントを埋め込むためには、そのフォントアウトラインファイルが得られる必要があり、そのフォントアウトライン定義が PDF 出力内へ書き込まれます。フォントが埋め込まれないときは、グリフ幅と一般的記述だけが PDF 出力へ書き込まれます。デフォルト：true フォント埋め込みを、skipembedding オプションを用いて制御することもできます。
encoding	（文字列。PDF_load_font( ) の引数としては必須ですが、暗黙的フォント読み込みではオプショナル）このフォントに対して、入ってくるテキストが解釈されるエンコーディング： ワイドキャラクターエンコーディング：unicode か glyphid バイト・マルチバイトエンコーディング： ▸ 8 ビットエンコーディングか（廃止）非 Unicode（レガシー）CMap の名前 ▸（Unicode 対応言語バインディングでは不可）日中韓コードページの cp932・cp936・cp949・cp950 のいずれか ▸ 自動的に選択されるエンコーディングの host か auto、またはユーザー定義エンコーディングか、オペレーティングシステムに知られているエンコーディングの名前 ▸ フォントの内部エンコーディングを選択するための builtin（主にレガシーな記号フォントで使用） デフォルト（オプションリストによる暗黙的フォント読み込みの場合のみ）：unicode PDF_load_font( )：このオプションをメソッドの引数として与えることもできます。 PDF_fill_textblock( )：このオプションは、text 引数のテキストが空で defaulttext プロパティが用いられている場合と、font オプションを与えている場合を除き、必須です。

表 4.1 PDF_load_font( ) と、オプションリストを用いた暗黙的フォント読み込みのフォントオプション

オプション	説明
fallbackfonts	（表 4.2 に従ったオプションリストのリスト）読み込むフォントに対する 1 個ないし複数の予備フォントを指定します。各予備フォントは、font サブオプションのフォントハンドルによって、または暗黙的フォント読み込みのための適切なサブオプションによって、定義する必要があります。 glyphcheck=replace かつベースフォントの 8 ビットエンコーディングに含まれないキャラクターがテキスト内にあるとき、またはベースフォントがキャラクターに対するグリフを含んでいないとき、またはグリフ置き換えが forcechars サブオプションで強制されているとき、PDFlib は、すべての指定された予備フォントの中で、リストされた順番に、このキャラクターに対するグリフを検索します。適合するグリフがいずれかの予備フォントの中に見つかったときは、そのキャラクターはこのグリフで印字され、見つからなかったときは、通常のグリフ置き換えのしくみが適用されます。
fontname	（名前文字列。PDF_fill_textblock( ) 以外の暗黙的フォント読み込みの場合、テキスト書式オプション font を指定していない場合には必須）フォントの実際の名前またはエイリアス名（大文字・小文字は区別されます）。この名前を用いてフォントデータが場所特定されます。フォントファイル名を与える場合には、その接尾辞を削る必要があります。Windows では、カンマの後にフォントスタイル名をフォント名に付加することができます（詳しくは PDFlib チュートリアル参照）。fontname の先頭が「@」キャラクターの場合、そのフォントには縦書きが適用されます。 PDF_load_font( )：フォント名をメソッドの引数として与えることもできます。
fontstyle	（キーワード・廃止）擬似フォントスタイルの作成を制御します。使えるキーワードは normal・bold・italic・bolditalic です。このフォントを用いて作成されるテキストは、適切に fakebold および / または italicangle テキスト書式オプションを用いてスタイルされます。italicangle に別の値が設定されていない限り、−12 が用いられます。デフォルト：normal
foreground-color	（色。COLR テーブルを持つ OpenType カラーフォントに対してのみ意味を持ちます）「前面色」として指定されているグリフレイヤーの色。使える色空間は lab・gray・rgb（sRGB として解釈されます）・cmyk。デフォルト：黒
foreground-opacity	（float かパーセント値。COLR テーブルを持つ OpenType カラーフォントに対してのみ意味を持ちます。PDF/A-1・PDF/X-3 では値 1 のみが許されます）「前面色」として指定されているグリフレイヤーの不透明度。デフォルト：1
initialsubset	（Unichar か Unicode 範囲のリスト、またはキーワードのリスト。embedding=true かつ subsetting=true のときのみ意味を持ちます）初期フォントサブセットにグリフを含めたい Unicode 値群を指定します。これを活用すると、同等のサブセットを生成させることによって PDF 出力フォント容量を削減させることができ、複数の文書を連結する際の最適化が容易になります。Unicode 値群は、Unichar 群か Unicode 範囲群によって明示的に指定することもできますし、8 ビットエンコーディングの名前によって暗黙的に指定することも可能です。使えるキーワード（デフォルト：empty）： ***empty*** 初期フォントサブセットは空になります。 **任意の *8* ビットエンコーディング名** そのエンコーディング内のすべての Unicode 値が初期サブセットへ含まれます。
keepfont	（論理値。Type 3 フォント・OpenType カラーフォントでは不可）false にすると、フォントは PDF_end_document( ) で自動的に削除されます。true にすると、フォントは後続する文書群でも、PDF_close_font( ) を呼び出すまで使うことができます。デフォルト：PDF_load_font( ) がオブジェクトスコープ内で呼び出されたなら true、そうでないなら false

表 4.1 PDF_load_font( ) と、オプションリストを用いた暗黙的フォント読み込みのフォントオプション

オプション	説明
keepnative	（論理値。Unicode 非対応言語バインディングでのみ意味を持ちます。埋め込みフォントの場合には強制的に false になります。廃止）false にすると、このフォントのテキストは、PDF 出力作成時に CID 値へ変換されます（glyphid 指定と Identity-H エンコーディングを使って）。API メソッドに与えるテキストは、選んだ CMap に依然一致していなければなりません。しかしそのフォントは、テキストフローと、あらゆる単純テキスト出力メソッドに使うことができます（ですがフォームフィールドには使えません）。 true にすると、このフォントのテキストは、選んだ CMap に従ってそのネイティブな形式のまま PDF 出力へ書き込まれます。そのフォントは、フォームフィールドと、あらゆる単純テキスト出力メソッドに使うことができますが、テキストフローには使えません。デフォルト：TrueType フォントの場合は false、それ以外の場合は true。
linegap	（−2048 から 2048 までの整数）行間。上記 ascender 参照。
metadata	（オプションリスト）フォントのためのメタデータを与えます（307 ページ「15.2 XMP メタデータ」参照）
optimize-invisible	（論理値。PDF/X-1/2/3 では不可）true にすると、不可視テキスト（すなわち textrendering=3）に対してのみ用いられているフォントは、embedding=true であっても埋め込まれません。これは、OCR 出力を不可視テキストとして持つ PDF/A 出力へのフォント埋め込みを抑制するために有用でしょう。デフォルト：false
preservepua	（論理値）true にすると、フォント内で私用領域（PUA）内の Unicode 値へマップされているキャラクターは、PDF 出力内でその PUA 値を保ちます。これは、その PUA キャラクター群が日本語の外字 /EUDC キャラクターのようにローカルに定義された意味付けを伴っている場合に有用でしょう。false にすると、PUA キャラクターは、PDF 出力内の ToUnicode CMap 内で U+FFFD（Unicode 置換キャラクター）へマップされます。デフォルト：false
readfeatures	（論理値）OpenType 機能テーブルをフォントから読み込むかを指定します。OpenType 機能のテキストへの適用は features オプションによって制御されます（表 5.4 参照）。OpenType 機能が必要でないときは、このオプションを false に設定すると、フォントの読み込みが速くなる可能性があります。デフォルト：true
readkerning	（論理値）カーニング値をフォントから読み込むかどうかを制御します。カーニング値のテキストへの実際の適用はテキストオプション kerning によって制御されます（表 4.6 参照）。カーニングが必要でないときは、このオプションを false に設定すると、フォントの読み込みが速くなる可能性があります。デフォルト：横書きの場合には true、縦書きの場合には false
readselectors	（論理値。TrueType・OpenType フォントに対してのみ意味を持ちます）true にすると、異体字セレクターがフォント内にあれば読み込まれます。これは、Unicode テキスト内の表意文字異体字シーケンス（IVS）を自動的に置き換えるために必要です。
readshaping	（論理値）TrueType または OpenType フォントの、複雑用字系のシェーピングに必要なシェーピングテーブルを読み込むかどうかを指定します。テキストの実際のシェーピングは shaping オプションによって制御されます（表 5.4 参照）。シェーピングが必要でないときは、readfeatures を false に設定すると、メモリーを節約することができます。デフォルト：true
readvertical metrics	（論理値）true にし、かつオプション vertical を true にすると、TrueType または OpenType フォントの縦書きメトリックを（存在すれば）使用してテキスト出力が整形されます。デフォルト：false

表 4.1 PDF_load_font( ) と、オプションリストを用いた暗黙的フォント読み込みのフォントオプション

オプション	説明
replacementchar	（Unichar かキーワード。glyphcheck=replace の場合にのみ意味を持ちます）選択しているフォントで利用できない、かつ予備フォントまたはタイポグラフィー的に類似のキャラクターで置き換えできないグリフを、指定した Unicode 値で置き換えます。そのフォントが、指定された Unicode キャラクターに対するグリフを含んでいない場合には、auto の動作が適用されます。U+0000 を使うとフォントの「グリフなし」記号を指定できます（PDF/A・PDF/UA・PDF/X-4/5 では不可）。encoding= builtin で読み込んだ記号フォントに対しては、Unicode 値でなくバイトコードを与える必要があります。Unicode キャラクターのかわりに以下のキーワードを用いることもできます（デフォルト：auto）： ***auto*** 以下の一覧の中で、そのフォント内でグリフが得られる最初のキャラクターが、置き換えキャラクターとして使用されます：U+00A0（NO-BREAK SPACE）、U+0020（SPACE）、U+0000（グリフが見つからない記号）。 ***drop*** そのキャラクターに対して出力が作成されません。 ***error*** タイポグラフィー的に類似にキャラクターが得られない場合に例外が発生します。これは、読めないテキスト出力を回避するために使えます。
simplefont	（論理値。TrueType アウトラインを持つ 8 ビットエンコーディングのフォントに対してのみ意味を持ちます）このオプションを true にし、かつ subsetting=false にした場合には、そのフォントは CID フォントとしてではなくシンプルなフォントとして出力されます。PDF ビューアーによっては。表示するマシンにそのフォントがインストールされていないときフォント置き換えを成功させるためにこれが必要です。デフォルト：false
skipembedding	（キーワードのリスト。embedding=true の場合にのみ意味を持ちます。PDF/A・PDF/X・PDF/UA では空リストのみ可）1 個ないし複数の条件をそのフォントが満たす場合には、フォント埋め込みを無視します。これは、全般としてはフォント埋め込みを望むけれども、ある特定の状況に対してはフォントが埋め込まれないほうが望ましいという場合に有用でしょう。使えるキーワード： ***fstype*** フォントが、TrueType か OpenType フォントであり、かつ、そのフォントの OS/2 テーブル内の fsType フラグに従って埋め込みを許可していない。 ***hostfont*** フォントがホストフォントして読み込まれた。 ***latincore*** フォントが、標準 Type 1 フォント（Latin コアフォントとも呼ばれます）の集合に含まれている（全一覧については PDFlib チュートリアルを参照）が、フォントファイルが得られないので埋め込めない。 デフォルト：PDF/A・PDF/UA・PDF/X モードの場合か、embedding オプションが指定されている場合には空リスト、そうでないなら {latincore}
strikeselect	（整数。sbix カラーテーブルを持つ OpenType カラーフォントに対してのみ意味を持ちます）ppem 値、すなわちデザイン単位内のビットマップピクセル数に基づいてビットマップストライクを選択： 0 最小の ppem 値を持つストライクを選択することにより、ファイルサイズを最小化します。 100 最大の ppem 値を持つストライクを選択することにより、レンダリング品質を最適化します。 フォントが再度読み込まれる際には strikeselect 値は無視されます。デフォルト：100
subsetlimit	（float またはパーセント値。Type 3 フォントでは無視されます）フォントのグリフの総数に対する、文書で使っているグリフの比率が、指定パーセント値を超えるときは、自動フォントサブセット化を無効にします。デフォルト：100%
subsetminsize	（float）元のフォントファイルのサイズが、KB 単位の指定値より小さいときは、自動 TrueType/OpenType フォントサブセット化を無効にします。デフォルト：50

表 4.1 PDF_load_font( ) と、オプションリストを用いた暗黙的フォント読み込みのフォントオプション

オプション	説明
subsetting	（論理値）埋め込みフォントをサブセット化するかを制御。Type 3 フォントをサブセット化するには、フォント定義を 2 回に分けて行う必要があります（PDFlib チュートリアル参照）。デフォルト：サブセット化は、subsetlimit・subsetminsize 設定に基づいて自動的に有効にされます。OpenType カラーフォントから生成された Type 3 フォントに対してはサブセット化はデフォルトで有効化されます。
unicodemap	（論理値。PDF/A-1a/2a/2u/3a/3u・PDF/UA-1 では false に設定してはいけません）ToUnicode CMap の生成を制御します。このオプションはタグ付き PDF モードでは無視されます。デフォルト：true
vertical	（論理値。TrueType・OpenType フォントのみ。フォント名が @ で始まるときは true を強制されます）true なら、縦書き用にフォントを用意し、Opentype の vrt2・vert 機能を、もしそのフォントが対応していれば有効化します。
xheight	（−2048 から 2048 までの整数）x ハイト。上記 ascender 参照。

表 4.2 PDF_load_font( ) の fallbackfonts オプションのサブオプション

オプション	説明
フォントオプション群	フォントを暗黙的に（すなわち font オプションでなく fontname オプションで）指定したときは、表 4.1 に従った fallbackfonts 以外のすべてのフォントオプションをサブオプションとして与えることができます。
font	（フォントハンドル）PDF_load_font( ) を fallbackfonts オプションなしで呼び出して返されたフォントハンドル。このオプションを与えると、すべてのフォント読み込みオプションが無視されます（fontname・encoding も）。
fontsize	（float またはパーセント値）予備フォントの文字サイズを、ユーザー座標で、またはカレント文字サイズに対するパーセント値として表したもの。このオプションを利用すると、予備フォントのデザインされた文字サイズがベースフォントと合わないときに、予備フォントの文字サイズを調節することができます。デフォルト：100%
forcechars	（Unichar か Unicode 範囲のリスト、またはキーワード 1 個）つねに予備フォント内のグリフで印字させたい（glyphcheck 設定によらず）キャラクター群を指定します。予備フォントは、指定したキャラクターに対するグリフを含んでいるか（個々のキャラクターを指定した場合）、あるいは少なくとも指定した Unicode 範囲の最初と最後のキャラクターに対するグリフを含んでいる必要があります。ベースフォントに対して 8 ビットエンコーディングを指定している場合でも、このオプションには Unicode 値を指定することが可能です。 以下のキーワードのいずれか一つを与えることもできます： ***gaiji*** 予備フォントは SING フォントを参照する必要があり、そしてこのキーワードは、SING フォントのメイングリフの Unicode 値へのショートカットとして利用することができます。 ***all*** キャラクターがベースフォント内で得られる場合であっても、予備フォント内のすべてのグリフが、ベースフォント内の対応するキャラクターを置換するために使われます。
textrise	（float またはパーセント値）テキストライズ値（表 4.6 参照）。パーセント値は、予備フォントに対して指定した（すなわち、fontsize サブオプションがあるならそれを適用した後の）文字サイズに対する比です。このオプションを利用すると、予備フォントのデザインされたテキストライズがベースフォントと合わないときに、予備フォントのテキストの位置を調節することができます。デフォルト：0

C++ Java C# **void close_font(int font)**

Perl PHP **close_font(int font)**

C **void PDF_close_font(PDF *p, int font)**

文書内でまだ使われていない、開かれているフォントハンドルを閉じます。

font 文書内でまだ使われていない、かつ閉じられていない、*PDF_load_font()* によって返されたフォントハンドル。

詳細 このメソッドはフォントハンドルを閉じ、そのフォントに関連づけられていたすべてのリソースを解放します。この呼び出しの後に、そのフォントハンドルを使ってはいけません。通常、フォントは文書の終わりに自動的に閉じられます。しかし、フォントを閉じることは以下のような場合に有用です：

- フォントのプロパティを*PDF_info_font()*で取得した後、そのフォントをカレントのPDF文書内では使わないことが決定された。
- フォントが文書の境界を越えて保持されていたが（*PDF_load_font()* の *keepfont* オプションで）、もう必要ないので捨てるべきである。

フォントがカレント文書内ですでに使われているときは、閉じてはいけません。

スコープ 任意

C++ Java C# **double info_font(int font, String keyword, String optlist)**

Perl PHP **float info_font(int font, string keyword, string optlist)**

C **double PDF_info_font(PDF *p, int font, const char *keyword, const char *optlist)**

読み込んだフォントに関する詳しい情報を取得します。

font *PDF_load_font()* によって返されたフォントハンドル、あるいはキーワードによっては -1（PHP では 0）。

keyword ほしい情報を表 4.4 に従って指定したキーワード。使えるキーワード：

- グリフマッピングのためのキーワード：*cid*・*code*・*glyphid*・*glyphname*・*unicode*
- フォントメトリック：*ascender*・*capheight*・*descender*・*italicangle*・*linegap*・*xheight*
- フォントのファイル・名前・種別：*cidfont*・*familyname*・*fontfile*・*fontname*・*fonttype*・*outlineformat*・*singfont*・*standardfont*
- フォントの技術的情報：*feature*・*featurelist*・*hostfont*・*kerningpairs*・*numglyphs*・*vertical*
- 表意文字異体字セレクターのためのキーワード：*maxuvsunicode*・*minuvsunicode*・*selector*・*selectorlist*
- フォントとエンコーディングの関係：*codepage*・*codepagelist*・*encoding*・*fallbackfont*・*keepnative*・*maxcode*・*numusableglyphs*・*replacementchar*・*symbolfont*・*unicodefont*・*unmappedglyphs*
- カレント文書に対するフォント処理の結果：*numusedglyphs*・*usedglyph*・*willembed*・*willsubset*
- カラーフォント処理：*colormode*・*colortype*・*numcolorglyphs*

optlist 選択したキーワードをさらに修飾するオプションリスト。以下のオプションが使えます：

- 表 4.4 の各キーワードの欄で解説する、キーワード独自のオプション群。

- グリフを指定するための、表 4.3 に従ったマッピングオプション：
  cid・*code*・*glyphid*・*glyphname*・*selector*・*unicode*
  これらのオプションは、マッピングオプション *cid*・*code*・*glyphid*・*glyphname*・*unicode* のソース値を定義します。これらのマッピングオプションは相互に排他的です。*code*・*glyphname*・*unicode* オプションは *encoding* オプションと組み合わせることが可能です。

表 4.3 PDF_info_font( ) でグリフを指定するオプション

オプション	説明
cid	（数値）グリフの CID 値。cidfont=1 のときのみ意味を持ちます
code	（範囲 0 ～ 255 の数値。8 ビットエンコーディングのフォントのみ）エンコーディングスロット
glyphid	（範囲 0 ～ 65535 の数値）内部グリフ ID
glyphname	（文字列）グリフの名前。cidfont=1 のときには意味を持ちません
selector	（Unichar）範囲 U+0xFE00 ～ U+FE0F または U+E0100 ～ U+E01EF 内の異体字セレクターの Unicode 値。selector キーワードによって返されたすべての値をここで与えることができます。
unicode	（Unichar）Unicode キャラクター

戻り値 *keyword* によって、および場合によってはオプションで補足して要求した、フォントまたはエンコーディングの特性の値。キーワードとオプションの組み合わせが仕様外のときは −1（PHP では 0）が返されます。要求されたテキストがテキストを生み出す場合には、文字列番号が返され、その対応する文字列を、*PDF_get_string( )* を用いて取得する必要があります。

詳細 このメソッドは、以下の異なるソースからの情報を提供します：
- 有効なフォントハンドルを与えたときは、そのフォントから集めた情報を返します。例：フォントのメトリック・名前・種別、特定の *glyphid* に対する *unicode* 値。
- *font* = *-1*（PHP では 0）かつ *encoding* オプションを与えたときは、このエンコーディングに関する情報を返します。例：エンコーディング内の 1 つの *code* に対する *unicode* 値。
- *font* = *-1*（PHP では 0）かつ *encoding* オプションを与えていないときは、PDFlib の内部テーブル群から集めた情報を返します。例：特定の *glyphname* に対する *unicode* 値。

スコープ 任意

表 4.4 PDF_info_font( ) のキーワード・オプション

キーワード	説明
ascender	アセンダーのメトリック値。使えるオプション（デフォルト：fontsize=1000）： *faked*　（論理値）値をフォントファイルから得られなかったため推算する必要があるときは 1。そうでないときは 0 *fontsize*　（文字サイズ）値を指定文字サイズにあわせて比例計算します
capheight	キャップハイトのメトリック値。ascender 参照。
cid	指定したグリフの CID、あるいは得られなかったときは −1。使えるオプション：cid・glyphid・unicode・selector
cidfont	フォントが CID フォントとして埋め込まれるなら 1、そうでないなら 0

表 4.4 PDF_info_font( ) のキーワード・オプション

キーワード	説明
code	エンコーディングスロットを表す範囲 0 ～ 255 の数値か、またはそのようなスロットがフォント内またはエンコーディング内（encoding オプションを与え、かつ font=-1（PHP では 0）の場合）に見つからなかったときは –1。使えるオプションはマッピングオプション code・glyphid・glyphname・unicode と以下のオプションです： ***encoding*** （文字列）8 ビットエンコーディングの名前
codepage	指定したコードページにフォントが対応しているかどうかを調べます。この情報は、TrueType/OpenType フォントの OS/2 テーブルから、もし入手可能であれば採られます。使えるオプション： ***name*** （文字列。必須）コードページの名前を cpXXXX の形で。ここで XXXX は、コードページの 10 進数値を表します（例：cp437・cp1252） 以下の戻り値で、指定されたコードページにフォントが対応しているかどうかを示します。 ***-1*** フォントが TrueType・OpenType フォントでないので不明です。 ***0*** 指定されたコードページにフォントは対応していません。 ***1*** 指定されたコードページにフォントは対応しています。
codepagelist	フォントが対応しているすべてのコードページ（cpXXXX の形で）のスペース区切りのリストの文字列番号、あるいはフォントが TrueType・OpenType フォントでないのでコードページリストが不明のとき（codepage 参照）は –1。
colormode	（文字列番号。colortype が none 以外のときにのみ意味を持ちます）カラーフォント内のグリフ群に対する処理モード。以下の文字列のうちの 1 つに対する文字列番号を返します：ignorecolor・ignoremono・combined。フォントが OpenType カラーフォントでない場合には、ignorecolor に対する文字列番号が返されます。
colortype	（文字列番号）フォントがカラーグリフを含んでいるかを示し、かつ、もし含んでいる場合には、どの形式で含んでいるか。以下の文字列のうちの 1 つに対する文字列番号を返します：COLR・none・sbix・SVG
descender	ディセンダーのメトリック値。ascender 参照。
encoding	フォントのエンコーディングの名前の文字列番号。使えるオプション（デフォルト：actual）： ***api*** （論理値）true にすると、API で指定されるのと同じエンコーディング名 ***actual*** （論理値）true にすると、フォントに対して使われる実際のエンコーディング名
fallbackfont	unicode オプションで指定したキャラクターを印字するために用いられるベースフォントか予備フォントのハンドル。これを利用すると、指定したキャラクターに対して用いられるグリフを予備フォント群の連なりの中のどのフォントが実際に提供するかを調べることができます。そのキャラクターがいずれのベースフォント・予備フォントでも印字できないときは、–1 が返されます。使えるオプション：unicode
familyname	フォントファミリーの名前の文字列番号、または得られないなら –1

表 4.4 PDF_info_font( ) のキーワード・オプション

キーワード	説明
feature	PDFlib が対応している OpenType 機能テーブルをフォントが含んでいるかどうかを調べます。使えるオプション： ***language*** （キーワード。script を与えている場合にのみ可）言語の名前。デフォルト：_none ***name*** （キーワード。必須）OpenType 機能の 4 文字の名前。例：liga・ital・vert。機能 kern については取得できません。 ***script*** （キーワード）用字系の名前。デフォルト：_none language・name・script のいずれかに対して、未知のキーワードが与えられた場合には、例外が発生します。既知のキーワードの一覧については PDFlib チュートリアルを参照してください。以下の戻り値で、指定された OpenType 機能がフォント内に存在していて PDFlib がそれに対応しているかどうかを示します： ***-1*** フォント内に機能テーブルが一切見つからない。 ***0*** その機能は、そのフォント内で、指定された用字系と言語に対しては利用可能でない、または、PDFlib にとって未知である。 ***1*** その機能は、指定された用字系と言語に対して利用可能である。
featurelist	フォント内で得られて PDFlib が対応しているすべての機能のスペース区切りのリストの文字列番号、あるいは機能テーブルが全く得られないときは –1。
fontfile	フォントのアウトラインファイルのパス名の文字列番号か、または得られないときは –1
fontname	フォント名の文字列番号か、または得られないときは –1。使えるオプション（デフォルト：acrobat）： ***acrobat*** （論理値）true にすると、PDF フォント記述子の中の BaseFont 項目 ***api*** （論理値）true にすると、API で指定されるのと同じフォント名 ***full*** （論理値）true にすると、PDF のフォント記述子の FontName 項目
fonttype	フォント種別の文字列番号、または得られないときは –1。知られているフォント種別は OpenType・TrueType・TrueType（CID）・Type 1・Type 1（CID）・Type 1 CFF・Type 1 CFF（CID）・Type 3
glyphid	（8 ビットエンコーディングによるフォントと、encoding=builtin による記号フォントと、encoding=unicode によるフォントのみ）以下のオプションで同定される、フォント内部のグリフ ID（GID）を表す範囲 0 ～ 65535 の数か、またはそのようなグリフが見つからなかったときは –1。使えるオプションは、マッピングオプション群 cid・code・glyphid・glyphname・unicode・selector。
glyphname	指定したグリフの名前の文字列番号、またはそのようなグリフがフォント内または指定したエンコーディング内（encoding オプションを与えていてかつ font=–1（PHP では 0）の場合）に見つからないときは –1。使えるオプションはマッピングオプション code・glyphid・glyphname・unicode と以下のフォントです： ***encoding*** （文字列）8 ビットエンコーディングの名前
hostfont	フォントがホストフォントなら 1、そうでないなら 0
italicangle	フォントのイタリック角度（PDF のフォント記述子の中の ItalicAngle）
keepnative	（廃止）keepnative オプションの出力値。
kernamount	オプション gidright・gidleft で指定される 2 個のグリフの間のカーニング量をフォント座標で。
kerningpairs	フォント内のカーニングペアの数
linegap	行間のメトリック値。ascender 参照。
maingid	メイングリフのグリフ ID（SING テーブルのメンバー mainGID）。

表 4.4 PDF_info_font( ) のキーワード・オプション

キーワード	説明
maxcode	フォントのエンコーディングの最大のコード値。具体的には、シングルバイトエンコーディングに対しては 0xFF、encoding=glyphid では numglyphs-1、それ以外の場合はエンコーディング内の最大の Unicode 値。
maxuvs-unicode	有効な表意文字異体字シーケンス（IVS）に含まれうる最大 Unicode 値。
minuvs-unicode	有効な表意文字異体字シーケンス（IVS）に含まれうる最小 Unicode 値。
numcolor-glyphs	（整数）OpenType カラーフォントから生成されたフォントの中のカラーグリフの数。元の OpenType フォントの中のカラーグリフの数とは、数えた結果が異なる場合もあります。なぜなら HYPHEN-MINUS U+002D 対 SOFT HYPHEN U+00AD など、1 つのグリフが複数の Unicode 値に対して使われる場合があるからです。
numglyphs	フォント内のグリフの数（.notdef グリフを含む）。GID は 0 から始まりますので、GID のとりうる最大値は numglyphs より 1 小さくなります。
numusable-glyphs	PDF_load_font( ) で与えたエンコーディングによって到達可能なフォント内のグリフの数
numused-glyphs	その時点までに生成されたテキストの中で使われているグリフの数。
outlineformat	フォント形式：PFA・PFB・TTF・OTF・TTC のいずれか一つ。TTC・WOFF/WOFF2 フォントの場合には、TTF など、その基礎をなすベースフォント形式が返されます。カラーフォントの場合には、その基礎をなす OpenType フォント形式が返されます。
replacement char	replacementchar オプションで指定されたキャラクターの Unicode 値。encoding=builtin で読み込まれた記号フォントについては、Unicode 値でなくコードが返されます。
selector	index オプションで指定された番号を持つ異体字セレクターの Unicode 値。index オプションが指定されていない場合、または指定されたセレクターがそのフォント内で得られない場合には、-1 が返されます。使えるオプション： ***index*** （非負整数）セレクターの番号。
selectorlist	そのフォント内のすべての異体字セレクターの Unicode 値の空白区切りリストを内容とする文字列の文字列番号。16 進数字を h とすると、各値は hhhhh 形式で与えられます。
singfont	フォントが SING（外字）フォントなら 1、そうでないなら 0
standardfont	フォントが PDF コアフォントのときは 1、そうでないときは 0
symbolfont	フォントが記号フォントなら 1、そうでないなら 0（PDF のフォント記述子の中の symbol フラグ）
unicode	指定したグリフに対する Unicode UTF-32 値、または Unicode 値がフォントまたはエンコーディング（encoding オプションを与えていてかつ font=-1（PHP では 0）の場合）の中に見つからなかったときは -1。使えるオプションはマッピングオプション群 cid・code・glyphid・glyphname・unicode・selector と以下のオプションです： ***encoding*** （文字列）8 ビットエンコーディングの名前
unicodefont	フォントとエンコーディングの組み合わせがグリフ群に対するマッピングを与えるなら 1、そうでないなら 0。
unmapped-glyphs	Unicode PUA 値へマップされたフォント内のグリフの数。PUA 値がフォント内にすでに存在するか、それとも PDFlib によって割り当てられているかによりません。
usedglyph	指定したグリフ ID がテキスト内で使われているなら 1、そうでないなら 0。使えるオプション：glyphid

表 4.4 PDF_info_font( ) のキーワード・オプション

キーワード	説明
vertical	フォントが縦書き用なら 1、そうでないなら 0
weight	フォントの太さを 100 ～ 900 の範囲で表したもの。400 ＝標準、700 ＝ボールド
willembed	フォントが埋め込まれるなら 1、そうでないなら 0
willsubset	フォントのサブセットが作成されるなら 1、そうでないなら 0
xheight	行間の x ハイト値。ascender 参照。

4.2 テキストフィルター・書式オプション

この項では、「テキスト」という言葉は内容文字列を、すなわち、指定された書式（フォント・色など）を持つテキストを意味します。これに対し、名前文字列とハイパーテキスト文字列（ファイル名など）は書式を持ちません。詳しくは PDFlib チュートリアルを参照してください。

テキストオプションは、*PDF_set_text_option()*・*PDF_fit/info_textline()*・*PDF_fill_textblock()*・*PDF_add/create_textflow()* で使えます。テキストオプションは、表セルとテキストブロックにも適用されます。以下のテキストオプションのグループが利用可能です：

- 表 4.5 に従ったテキストフィルターオプション。
- 表 4.6 に従ったテキスト書式オプション。
- 表 5.4 に従ったシェーピング・タイポグラフィーオプション（*PDF_set_text_option()* では不可）。

表 4.5 PDF_set_text_option( )・PDF_fit/info_textline( )・PDF_fill_textblock( )・PDF_add/create_textflow( ) のテキストフィルターオプション

オプション	説明
actualtext	（論理値。PDF_set_text_option( )・PDF_fit_textline( )・PDF_fill_textblock( ) でのみ有効）true にすると、ToUnicode CMap から Unicode 値（1 個ないし複数）を導出されると正しくないであろうとき、かつ、ユーザー指定の Alt か ActualText が与えられていないとき、PDFlib は、然るべき ActualText を持ったマーク付きコンテンツ Span を生成します。これにより、オームとオメガ等、同一フォント内で見た目が類似の複数の Unicode 値を表現するために用いられているグリフに対するテキスト抽出が確実に正しく行えるようになります。デフォルト：true
charref	（論理値）true にすると、内容文字列内の数値・文字実体参照とグリフ名参照の置き換えが有効になります。[1] デフォルト：グローバル charref オプション
escape-sequence	（論理値）true にすると、内容文字列内のエスケープシーケンスの置き換えが有効になります。[1] デフォルト：グローバル escapesequence オプション
glyphcheck	（キーワード）グリフ検査ポリシ：テキスト内のコードが、選択したフォント内のグリフへマップできないときにどのように動作するか（デフォルト：グローバル glyphcheck オプション）： ***error*** グリフが得られないときに例外を発生させます。具体的なエラーメッセージは PDF_get_errmsg( ) で取得できます。 ***none*** 検査なし。notdef グリフは、PDF/A か PDF/UA-1 か PDF/X-4/5 モードでは例外を引き起こし、それ以外の場合には notdef グリフが出力内に現れることができます。 ***replace*** 得られないグリフを、ベースフォント・予備フォント内の適切なグリフが得られるならば、それに置き換えようと試みます。合字は分解されます。適切な置き換えができないときは、そのグリフは replacementchar で 置き換えられます。 ***replace*** 得られないグリフを、ベース・予備フォント内のタイポグラフィー的に類似のキャラクターで置き換え、合字を分解しようと試みます。然るべきグリフが見つからなかった場合には、そのキャラクターは replacementchar オプションに従って処理されます。

表 4.5 PDF_set_text_option( ) ・ PDF_fit/info_textline( ) ・ PDF_fill_textblock( ) ・ PDF_add/create_textflow( ) のテキストフィルターオプション

オプション	説明
normalize	（キーワード。encoding=glyphid の場合には無視されます）入ってくるテキストを、Unicode 正規形のうちの一つへ正規化します（デフォルト：none）： ***none*** 正規化を適用しません。これがデフォルト動作であり、選択したフォント内のグリフ群で表せるテキストを与えるのはクライアント側の役割です。 ***nfc*** 正規形 C（NFC）：正準分解の後に正準合成を行います。NFC は、結合シーケンスを、結合済キャラクターへ置換します。これは、結合シーケンスのあるワークフローのために有用です。なぜなら、フォントは通常、結合済キャラクターに対するグリフのみを含んでいるからです。NFC 正規化がない場合、PDFlib は、結合済キャラクターではなく、複数キャラクターのシーケンスを出力します。 ***nfkc*** 正規形 KC（NFKC）：互換分解の後に正準合成を行います。これは、キャラクターの意味付けにのみ関心があり、組版差異には関心のないワークフローのために有用です。たとえば、表示形のアラビア文字キャラクターをその原形へ変換したり、合字や分数を解決したり、縦書き形を横書き形へ置換したり、全角キャラクターを通常キャラクターへ置換したりします。 ***nfd*** 正規形 D（NFD）：正準分解 ***nfkd*** 正規形 KD（NFKD）：互換分解 NFD と NFKD は、結合シーケンスを作成することがありますので、これらは PDFlib ワークフローでは有用ではなさそうです。
textformat	（キーワード。C の場合と、Perl, PHP, Ruby で stringformat=legacy の場合にのみ使用可能）内容文字列を解釈するために用いられる形式。使えるキーワード：bytes・utf8・ebcdicutf8（IBM System i・IBM Z のみ）・utf16・utf16le・utf16be・auto。デフォルト：グローバル textformat オプション

1. この値は、以後、同じオプションを持った PDF_set_option( ) への呼び出しによってオーバーライドされる可能性があります。

表 4.6 PDF_set_text_option( ) ・ PDF_fit/info_textline( ) ・ PDF_fill_textblock( ) ・ PDF_add/create_textflow( ) テキスト書式オプション

オプション	説明
charspacing	（float またはパーセント値）字間。すなわち、文字列内の個々のキャラクターを配置した後のカレント点の変位。float 値はユーザー座標系の単位を指定します。パーセント値は fontsize に対する比です。文字どうしの間隔を拡げるには、横書きでは正の値を、縦書きでは負の値を用います。デフォルト：0
dasharray	（非負 float のリストかキーワード）描線（袋文字）テキストと文字飾りに対する線と線間の長さ。キーワード none を用いて実線を生成させることもできます。デフォルト：none
decoration-above	（論理値）true にすると、underline・strikeout・overline オプションで有効にされた文字飾りはテキストの上に、そうでなければテキストの下に描かれます。描画順を変えることで、飾り線は見えたり隠れたりします。すなわち、テキストが線にオーバプリントするか、あるいはその逆かを制御することが可能です。デフォルト：false
fakebold	（論理値）true にすると、グリフ輪郭を描線するか複数回の重ね書きによって太字テキストに見せかけます。デフォルト：false
fillcolor	（色）テキストの塗り色。[1] 単純テキスト出力メソッド群と PDF_fit_textline( ) で inittextstate=false の場合のデフォルト：カレントグラフィックステートの中の対応するオプション。 テキストフローと PDF_fit_textline( ) で inittextstate=true の場合のデフォルト：{gray 0}（PDF/A モードでは {lab 0 0 0}）

表 4.6 PDF_set_text_option( ) ・ PDF_fit/info_textline( ) ・ PDF_fill_textblock( ) ・ PDF_add/create_textflow( ) テキスト書式オプション

オプション	説明
font	（フォントハンドル）使いたいフォントのハンドル。このオプションを与えると、すべてのフォント読み込みオプションは無視されます（fontname・encoding も）。 デフォルト：暗黙的に読み込まれたフォントが得られるならそれ、そうでないなら、単純テキスト出力と PDF_fit_textline( ) で inittextstate=false では PDF_setfont( ) を用いて選択されたフォント。それ以外の場合はフォントは得られず、エラーが発生します。
fontsize	（文字サイズ）文字サイズを、カレントユーザー座標系の単位で表したもの。PDF_fit_textline( ) の場合、パーセント値は、枠の幅（orientate=north・south の場合）または高さ（orientate=east・west の場合）に対する比です。PDF_set_text_option( ) とテキストフローでは、パーセント値は、直前のテキストのサイズに対する比です。 デフォルト：PDF_setfont( ) は、単純テキスト出力メソッド群と、PDF_fit_textline( ) で inittextstate=false の場合に対するデフォルトのみを設定します。それ以外の場合はフォントは得られず、エラーが発生します。
gstate	（PDF_create_gstate( ) に対するオプションのオプションリストかグラフィックステートハンドル）グラフィックステートオプション群またはハンドル。そのグラフィックステートが、このメソッドで生成するすべてのテキストに対して効力を持ちます。デフォルト：グラフィックステートなし、すなわち、カレントの設定が使われます。
horizscaling	（float またはパーセント値。0 以外にする必要があります）テキストを、与えたパーセント値に横へ伸縮。テキストの伸縮は、テキストを、与えたパーセント値に縮めたり伸ばしたりします。テキストの伸縮はつねに横座標での値となります。デフォルト：100
inittextstate	（論理値。PDF_fit_textline( ) でのみ可）true にすると、すべてのテキスト書式オプションがデフォルト値で初期化されます。false にすると、カレントテキストステート値群が用いられます。デフォルト：false
italicangle	（float。縦書きでは使えません）テキストのイタリック（斜体）角度を度単位で（−90° と 90° の間で）表したもの。負の値を利用すれば、特に日中韓フォントなど、正立フォントしか得られないときに、斜体に見せることができます。デフォルト：0
kerning	（論理値）true にすると、readkerning オプションを付けて開いてあるフォントに対してカーニングが有効になります。そうでないならカーニングは無効になります。[2] デフォルト：グローバルオプション kerning
leading	（float またはパーセント値）複数行テキストに対する行送り、すなわち、テキストの隣り合う行のベースラインの間隔を、ユーザー座標における絶対値か、fontsize に対するパーセント値として指定します。この行送りを文字サイズに等しく設定すると、詰まった行間になります。ただし、隣り合う行のアセンダーとディセンダーは、通常は重なり合いません（leading=0 とすると行が重なり合います）。デフォルト：100% PDF_add/create_textflow( ) に対する行送りは次のように決定されます：行の先頭にオプションリスト群がある場合には、行送りは、最後の関連するオプション（font・fontsize・leading など）によって決定されます。同一行上にさらにオプションリスト群がある場合には、行送りに関連するオプション群はいずれも、fixedleading=false の場合にのみ考慮されます。その行にオプションリストが全くない場合には、直前の行送り値が用いられます。
overline	（論理値）true にすると、テキストの上方に線が引かれます。デフォルト：false

表 4.6 PDF_set_text_option( ) ・ PDF_fit/info_textline( ) ・ PDF_fill_textblock( ) ・ PDF_add/create_textflow( ) テキスト書式オプション

オプション	説明
shadow	（オプションリスト。PDF_fit_textline( ) ・ PDF_fill_textblock( ) ・ PDF_add/create_textflow( ) でのみ意味を持ちます）テキストに影付き効果を作成します（デフォルト：影なし）： ***disable*** （論理値。PDF_add/create_textflow( ) でのみ可）true にすると、以前に指定された影が無効にされます。デフォルト：false ***fillcolor*** （色）影の塗り色。デフォルト：{gray 0.8} ***gstate*** （PDF_create_gstate( ) に対するオプションのオプションリストかグラフィックステートハンドル）影に適用されるグラフィックステートオプション群またはハンドル。デフォルト：グラフィックステートなし、すなわち、カレントの設定が使用されます。 ***offset*** （2 個の float かパーセント値のリスト）テキストの参照点からの影のオフセットを、ユーザー座標で、または文字サイズに対するパーセント値として表したもの。デフォルト：{5% -5%} ***strokecolor*** （色。textrendering がテキストを描線するよう設定されている場合のみ効果を持ちます）影の描線色。デフォルト：カレント描線色 ***strokewidth*** （float。パーセント値かキーワード。textrendering がテキストを描線するよう設定されている場合のみ効果を持ちます）影の中の袋文字テキストの線幅（ユーザー座標で、または文字サイズに対するパーセント値として表したもの）。キーワード auto か、これに等価な値 0 は、内蔵のデフォルトを使用します。デフォルト：メインテキストもテキストを描線するよう設定されている場合にはカレント描線幅、そうでない場合は auto ***textrendering*** （整数）影のテキスト表現モード。デフォルト：textrendering のカレント値
strikeout	（論理値）true にすると、テキストを貫いて線が引かれます。decorationabove も参照してください。デフォルト：false
strokecolor	（色。描線されたテキストでのみ有効。textrendering を参照）テキストの描線色。デフォルト：fillcolor 参照
strokewidth	（float・パーセント値・キーワードのいずれか。textrendering をテキストを描線するよう設定している場合のみ有効）袋文字テキストの線幅（絶対値、または fontsize に対する比で）。キーワード auto か、それと等価な値 0 を指定すると、内蔵のデフォルトを用います。デフォルト：auto
tagtrailing-hyphen	（Unichar かキーワード。タグ付き PDF でのみ意味を持ちます）テキスト（グリフ置換を適用した場合は、その後の）の中の最後のキャラクターが、指定された Unicode 値に等しい場合には、そのフォントによって要請される場合には ActualText ソフトハイフン U+00AD を持った Span としてタグ付けされ、かつ autospace は付加されません。キーワード none とすると、ソフトハイフンのためのタグ付けは一切行われません。デフォルト：U+00AD

表 4.6 PDF_set_text_option( ) ・ PDF_fit/info_textline( ) ・ PDF_fill_textblock( ) ・ PDF_add/create_textflow( ) テキスト書式オプション

オプション	説明
textrendering	（整数）テキスト表現モード（デフォルト：0）： *0* テキストを塗る　*4* テキストを塗り、クリッピングパスに追加 *1* テキストを描線（袋文字）　*5* テキストを描線し、クリッピングパスに追加 *2* テキストを塗って描線　*6* テキストを塗って描線し、クリッピングパスに追加 *3* 不可視テキスト　*7* テキストをクリッピングパスに追加 textrendering=4/5/6/7（クリッピングモード群）の動作： ▸ PDF_fit_textflow( ) ・ PDF_fit_table( ) ・ PDF_fill_textblock( ) ・ PDF_fit_textline( ) の後には、textpath オプションが指定されている場合には、クリッピング効果はありません。 ▸ クリッピング領域は、単純テキスト出力メソッド群への複数回の呼び出しにわたって蓄積することができますが、PDF_fit_textline( ) への複数回の呼び出しにわたっては蓄積されません。 ▸ PDF_fit_textline( ) : 指定された fillcolor と strokecolor は、メソッド呼び出しの後にも有効となります。 このテキストレンダリングモードは、OpenType カラーフォントから生成されたものなど Type 3 フォントには効力を持ちませんが、ただし textrendering=3 と 7 では不可視テキストになります。
textrise	（float またはパーセント値）テキストライズ値。テキストを配置したい位置とベースラインとの間隔を指定します。正の値のテキストライズはテキストを上へ移動させます。テキストライズはつねに縦座標での値となります。これは、上付き・下付きをさせたいときに有用でしょう。パーセント値は fontsize に対する比です。デフォルト：0
underline	（論理値）true にすると、テキストの下方に線が引かれます。デフォルト：false
underline-position	（float・パーセント値・キーワードのいずれか）下線テキストの下線の位置を、ベースラインからの距離で（絶対値か、または文字サイズに対する比。代表的な値は -10%）指定します。キーワード auto を指定すると、フォントから取得されるフォント固有の値が採用されます。デフォルト：auto
underline-width	（float・パーセント値・キーワードのいずれか）テキストの下線の線幅（絶対値か、または文字サイズに対する比）。キーワード auto か値 0 を指定すると、フォントからフォント固有の値が得られる場合はそれが採用され、そうでないなら 5% となります。デフォルト：auto
wordspacing	（float またはパーセント値）単語間隔。すなわち、行内の個々の単語を配置した後のカレント点の変位。言い換えれば、カレント点が、各スペースキャラクター（U+0020）の後で横へ移動されます。値は、ユーザー座標系で表すか、または文字サイズに対する比で指定します。デフォルト：0

1. この値は、以後、単純テキスト出力メソッド群と inittextstate=false を伴う PDF_fit_textline( ) への呼び出しのための PDF_setcolor( ) によってオーバーライドされることがあります。
2. この値は、以後、同じオプションを伴う PDF_set_option( ) への呼び出しによってオーバーライドされることがあります。

C++ Java C# ***void set_text_option(String optlist)***

Perl PHP ***set_text_option(string optlist)***

C ***void PDF_set_text_option(PDF *p, const char *optlist)***

単純テキスト出力メソッド群のための1個ないし複数のテキストフィルターまたはテキスト書式オプションを設定します。

optlist フォント・テキストオプションを以下のように指定したオプションリスト：

- 表 4.5 に従ったテキストフィルターオプション：
  actualtext・*charref*・*escapesequence*・*glyphcheck*・*normalize*・*textformat*
- 表 4.6 に従ったテキスト書式オプション：
  charspacing・*dasharray*・*decorationabove*・*fakebold*・*fillcolor*・*font*・*fontsize*・*gstate*・*horizscaling*・*inittextstate*・*italicangle*・*kerning*・*leading*・*overline*・*strikeout*・*strokecolor*・*strokewidth*・*tagtrailinghyphen*・*textrendering*・*textrise*・*underline*・*underlineposition*・*underlinewidth*・*wordspacing*

詳細　テキストオプション群の値は、すべての単純テキスト出力メソッドと、*inittextstate=false* を伴う *PDF_fit_textline()* に対して意味を持ちます。*PDF_set_text_option()* への呼び出しを、*PDF_setfont()*・*PDF_setcolor()* への呼び出しと混在させるべきではありません。

すべてのテキストオプションは、ページ・パターン・テンプレート・グリフ記述の開始でそれらのデフォルト値へリセットされ、そしてカレントの**ページ・パターン・テンプレート・グリフ**スコープが終了するまでそれらの値を保持します。ただし、テキストオプション群を *inittextstate* オプションを用いてリセットすることもできます。

スコープ　ページ・パターン・テンプレート・グリフ

4.3 単純テキスト出力

この節で挙げるメソッド群は、低レベルテキスト出力のために使うことができます。より高度なテキスト出力のためには、より強力なテキスト行・テキストフローメソッド群を使うことを推奨します（99 ページ「5.1 テキスト行による一行テキスト」・106 ページ「5.2 テキストフローによる複数行テキスト」参照）。

C++ Java C# ***void PDF_setfont(int font, double fontsize)***

Perl PHP ***setfont(int font, float fontsize)***

C ***void PDF_setfont(PDF *p, int font, double fontsize)***

カレントフォントを、指定するサイズで設定します。

font *PDF_load_font()* によって返されたフォントハンドル。

fontsize 文字サイズを、カレントユーザー座標系の単位で測ったもの。文字サイズは 0 にしてはいけません。文字サイズを負の値にすると、カレント変換行列について鏡映されたテキストになります。

詳細 このメソッドは、単純テキスト出力メソッド群（*PDF_show()*・*PDF_fit_textline()* など）で用いられるべきフォントと文字サイズを設定します。これは、*PDF_set_text_option()* をオプションリスト *font=<* フォント *> fontsize=<* 文字サイズ *>* とともに呼び出した場合とほぼ同等です。ただし、*PDF_set_text_option()* と異なり、このメソッドは、*leading* テキストオプションを *fontsize* に設定します。

フォントは、ページごとに、いかなる単純テキスト出力メソッドを呼び出すよりも前に設定する必要があります。フォントの設定はページを超えて保持されません。

PDF_setfont() よりも *PDF_set_text_option()* を使用することを推奨します。*PDF_setfont()* への呼び出しを *PDF_set_text_option()* への呼び出しと混在させるべきではありません。

スコープ ページ・パターン・テンプレート・グリフ

C++ Java C# ***void set_text_pos(double x, double y)***

Perl PHP ***set_text_pos(float x, float y)***

C ***void PDF_set_text_pos(PDF *p, double x, double y)***

ページ上の単純テキスト出力のための位置を設定します。

x・y 新たなテキスト位置

詳細 テキスト位置は、ページが始まるごとにデフォルト値 *(0, 0)* に設定されます。グラフィックのためのカレント点と、カレントテキスト位置は、別々に保持されます。

スコープ ページ・パターン・テンプレート・グリフ

C++ Java C# ***void show(String text)***

Perl PHP ***show(string text)***

C ***void PDF_show(PDF *p, const char *text)***

C ***void PDF_show2(PDF *p, const char *text, int len)***

テキストをカレントのフォントとサイズでカレントテキスト位置に印字します。

text （内容文字列）印字したいテキスト。C で *PDF_show( )* を使うときは、***text*** は null 終了と前提されているので、null バイトを含んでいてはいけません。null キャラクターを含む可能性のある文字列に対しては *PDF_show2( )* を使います。

len （C 言語バインディングのみ）***text*** の長さ（バイト単位）。***len=0*** にすると null 終了文字列を与える必要があります。

詳細 フォントと文字サイズをあらかじめ、*PDF_setfont( )* か *PDF_set_text_option( )* で設定しておく必要があります。カレントテキスト位置は、印字したテキストの末尾へ移動します。その最終グリフの後では ***charspacing*** パラメーターが考慮されます。

スコープ **ページ・パターン・テンプレート・グリフ**

バインディング *PDF_show2( )* は C でのみ得られます。なぜなら他のすべてのバインディングでは、任意の文字列内容を *PDF_show( )* に与えることができるからです。

C++ Java C# ***void show_xy(String text, double x, double y)***

Perl PHP ***show_xy(string text, float x, float y)***

C ***void PDF_show_xy(PDF *p, const char *text, double x, double y)***

C ***void PDF_show_xy2(PDF *p, const char *text, int len, double x, double y)***

テキストをカレントフォントで、指定した位置に印字します。

text （内容文字列）印字したいテキスト。C で *PDF_show_xy( )* を使うときは、***text*** は null 終了と前提されているので、null バイトを含んでいてはいけません。null キャラクターを含む可能性のある文字列に対しては *PDF_show_xy2( )* を使います。

x・y テキストを印字させたい位置をユーザー座標系で指定します。

len （C 言語バインディングのみ）***text*** の長さ（バイト単位）。***len=0*** にすると null 終了文字列を与える必要があります。

詳細 フォントと文字サイズをあらかじめ、*PDF_setfont( )* か *PDF_set_text_option( )* で設定しておく必要があります。カレントテキスト位置は、印字したテキストの末尾へ移動します。その最終グリフの後では ***charspacing*** パラメーターが考慮されます。

スコープ **ページ・パターン・テンプレート・グリフ**

バインディング *PDF_show_xy2( )* は C でのみ得られます。なぜなら他のすべてのバインディングでは、任意の文字列内容を *PDF_show_xy( )* に与えることができるからです。

C++ Java C# **void continue_text(String text)**

Perl PHP **continue_text(string text)**

C **void PDF_continue_text(PDF *p, const char *text)**

C **void PDF_continue_text2(PDF *p, const char *text, int len)**

テキストを次の行に印字します。

text （内容文字列）印字したいテキスト。これを空文字列にすると、テキスト位置はそれでも次の行へ移動します。C で *PDF_continue_text()* を使うときは、*text* は null 終了と前提されているので、null バイトを含んでいてはいけません。null バイトを含む可能性のある文字列に対しては *PDF_continue_text2()* を使います。

len （C 言語バインディングのみ）*text* の長さ（バイト単位）。*len=0* にすると、*PDF_continue_text()* 同様、null 終了文字列を与える必要があります。

詳細 テキストの置かれる位置（*x・y* 位置）と、行の間隔は、*leading* テキストオプション（*PDF_set_text_option()* を用いて設定できます）と、もっとも最近の *PDF_show_xy()* か *PDF_set_text_pos()* への呼び出しによって決定されます。カレント位置は、印字したテキストの末尾へ移動します。このメソッドを連続で呼び出すと、*x* 位置は変わりません。

スコープ **ページ・パターン・テンプレート・グリフ。**このメソッドは縦書きでは使ってはいけません。

バインディング *PDF_continue_text2()* は C でのみ得られます。なぜなら他のすべてのバインディングでは、任意の文字列内容を *PDF_continue_text()* に与えることができるからです。

C++ Java C# **double stringwidth(String text, int font, double fontsize)**

Perl PHP **float stringwidth(string text, int font, float fontsize)**

C **double PDF_stringwidth(PDF *p, const char *text, int font, double fontsize)**

C **double PDF_stringwidth2(PDF *p, const char *text, int len, int font, double fontsize)**

任意のフォントでのテキストの幅を返します。

text （内容文字列）幅を取得したいテキスト。C で *PDF_stringwidth()* を使うときは、*text* は null 終了と前提されているので、null バイトを含んでいてはいけません。null バイトを含む可能性のある文字列に対しては *PDF_stringwidth2()* を使います。

len （C 言語バインディングのみ）*text* の長さ（バイト単位）。*len=0* にすると null 終了文字列を与える必要があります。

font *PDF_load_font()* によって返されたフォントハンドル。

fontsize フォントのサイズを、ユーザー座標の単位で測ったもの。

戻り値 *PDF_load_font()* で選んでいるフォントと、与えた *fontsize* での、*text* の幅。返される幅は負の値になることもあります（負の横伸縮を設定しているとき等）。縦書きのときは、もっとも幅の広いグリフの幅が返されます（テキストの実際の高さを知るには *PDF_info_textline()* を使います）。
字間が指定されているときは、それは最後のグリフにも適用されます（この動作は *PDF_info_textline()* とは異なります）。

詳細　幅の計算にあたっては、次のテキストオプション（*PDF_set_text_option()* で設定できます）のカレント値が考慮されます：*horizscaling*・*kerning*・*charspacing*・*wordspacing*。

スコープ　**オブジェクト**以外任意

バインディング　*PDF_stringwidth2()* は C でのみ得られます。なぜなら他のすべてのバインディングでは、任意の文字列内容を *PDF_stringwidth()* に与えることができるからです。

4.4 ユーザー定義 Type 3 フォント

C++ Java C# ***void begin_font(String fontname,***
double a, double b, double c, double d, double e, double f, String optlist)

Perl PHP ***begin_font(string fontname,***
float a, float b, float c, float d, float e, float f, string optlist)

C ***void PDF_begin_font(PDF *p, const char *fontname, int reserved,***
double a, double b, double c, double d, double e, double f, const char *optlist)

Type 3 フォントの定義を開始します。

fontname （名前文字列）フォントを登録して、以後の *PDF_load_font()* で使いたい名前。

reserved （C 言語バインディングのみ）予約済。0 にする必要があります。

a・b・c・d・e・f （Type 3 フォントのサブセット化の場合のパス 2 では無視されます）フォント行列の各要素。この行列は、グリフが描かれる座標系を定義します。この 6 個の値は PDF の行列を構成します。*a*×*d* を *b*×*c* に等しくしてはいけません。代表的な 1000×1000 の座標系に対するフォント行列は *[0.001, 0, 0, 0.001, 0, 0]* です。

optlist （Type 3 フォントのサブセット化の場合のパス 2 では無視されます）表 4.7 に従ったオプションリスト。次のオプションが使えます：*colorized*・*defaultcmyk*・*defaultgray*・*defaultrgb*・*familyname*・*stretch*・*weight*・*widthsonly*

詳細 このフォントには任意の数のグリフを入れることができ、*encoding=unicode* で使用できます。なお、Type 3 フォントは PDF 内では 8 ビット指定が必要です。生成されたフォントは、それを囲う**文書**スコープが終わるまで使えます。

サブセット化なしの Type 3 フォントは、ただ 1 回のパスで生成され、その際にはすべての引数とオプションに従います。Type 3 フォントに対するサブセット化は、2 回のパスを必要とします：

- パス 1 では、*PDF_begin_font()* の *widthsonly* オプションを指定し、メトリクスと Unicode 情報だけを与えます。パス 1 は、フォントとグリフのメトリクスだけを定義します：*PDF_begin_font()* でフォントマトリックスを与え、かつ、*PDF_begin_glyph_ext()* で *wx* とグリフ外接枠を与える必要があります。それぞれのグリフに対しては、*PDF_begin_glyph_ext()* と *PDF_end_glyph()* だけが必要であり、これ以外の、実際のグリフの形状を定義する呼び出しは一切必要ありません。1 つのグリフ記述の始まりと終わりとの間で他のメソッドを呼び出しても、そのメソッドは無視されます。
- パス 2 は、そのフォントを使用したすべてのテキストを生成した後に行う必要があります。そこでグリフのアウトラインかビットマップを定義します。行列値とオプションは無視されます。最終ページが生成された後、PDFlib は、その文書の中でどのグリフが使われたかを知り、それに必要なグリフ定義だけを埋め込むことによってフォントのサブセットを構築します。

  使われていないグリフを記述した API メソッド呼び出しは、警告なしに無視されます。エラーコードを返しうるメソッド（*PDF_load_image()* など）はエラー値を返しますので、アプリケーションはそれを無視する必要があります。グリフが使われていないのか、それとも本当のエラーなのかを区別するために、*PDF_get_errnum()* が返すエラーメッセージ番号をチェックする必要があります：それがゼロの場合には、グリフは無視されており、すなわち、本当のエラーは起こっていません。

両方のパスにおいて、同一のグリフの集合を定義する必要があります。

スコープ **オブジェクト**以外任意。このメソッドは**フォント**スコープを開始させます。対応する *PDF_end_font( )* と必ずペアにして呼び出す必要があります。

表 4.7 PDF_begin_font( ) のオプション（Type 3 フォントのサブセット化の場合のパス 2 では無視されます）

オプション	説明
colorized	（論理値。廃止）PDF_begin_glyph_ext( ) の colorized オプションのデフォルトを設定。デフォルト：false
defaultgray ***defaultrgb*** ***defaultcmyk***	（ICC ハンドルまたはキーワード）与えられた ICC プロファイルハンドルに従って、そのフォントの中のグリフ記述群に対するデフォルトのグレー・RGB・CMYK のうちのいずれかの色空間を設定します。オプション defaultrgb ではキーワード srgb も使えます。
familyname	（文字列。PDF 1.5）フォントファミリーの名前
stretch	（キーワード。PDF 1.5）フォントの幅の値：ultracondensed・extracondensed・condensed・semicondensed・normal・semiexpanded・expanded・extraexpanded・ultraexpanded。デフォルト：normal
weight	（整数またはキーワード。PDF 1.5）フォントの太さ。とりうる数値または同等のキーワード：100=thin、200=extralight、300=light、400=normal、500=medium、600=semibold、700=bold、800=extrabold、900=black。デフォルト：normal
widthsonly	（論理値）値 true にすると、Type 3 フォントのサブセット化の場合のパス 1 が開始されます。そこではフォントとグリフのメトリクスだけを定義します。PDF_begin_glyph_ext( ) と PDF_end_glyph( ) の間で、他の API メソッドを呼び出してはいけません。他のメソッドをたとえ呼び出しても、PDF 出力には何ら効果を与えず、例外も発生しません。widthsonly=false のときは、Type 3 フォントのサブセット化の場合のパス 2 が開始されます。そこではグリフのアウトラインかビットマップを定義できます。デフォルト：false

C++ Java C# ***void end_font( )***
Perl PHP ***end_font( )***
C ***void PDF_end_font(PDF *p)***

Type 3 フォントの定義を終了させます。

スコープ **フォント**。このメソッドは**フォント**スコープを終了させます。対応する *PDF_begin_font( )* と必ずペアにして呼び出す必要があります。

C++ Java C# ***void begin_glyph_ext(int uv, String optlist)***
Perl PHP ***begin_glyph_ext(int uv, string optlist)***
C ***void PDF_begin_glyph_ext(PDF *p, int uv, const char *optlist)***

Type 3 フォントのためのグリフ定義を開始させます。

uv グリフの Unicode 値。各 Unicode 値は、1 個のグリフ記述に対してのみ与えることができます。Unicode 値 0 を持つグリフには、そのグリフが指定されたか否かにかかわらず、グリフ ID 0 とグリフ名 *.notdef* が割り当てられます。

uv=−1 の場合、その Unicode 値は、*glyphname* オプションから、PDFlib の内部グリフ名リストに従って導き出されます。グリフ名が未知の場合には、PUA 値（U+E000 から始まる）

が順次割り当てられます。この値は、*PDF_info_font()* を用いてクエリーすることができます。

optlist （Type 3 フォントのサブセット化の場合のパス 2 では無視されます）表 4.8 に従ったオプションリスト。以下のオプションが使えます：***boundingbox*** • ***colorized*** • ***glyphname*** • ***width***

詳細 グリフ記述は、テキスト・グラフィック・画像メソッドを使って定義することができます。このメソッドは、すべてのテキスト・グラフィック・色ステートパラメーター群を、それらのデフォルト値へリセットします。

デフォルトでは、グリフ記述はそのグリフの形状のみを指定することができ、その色を指定することはできません。グリフは、それが配置された時点でのカレントの塗り色で着色されます。またグリフ記述は、透過を設定するための操作をすべて避ける必要があり、かつ、マスク以外の画像を配置してはいけません。***colorized*** オプションを ***true*** にすると、これらの制約はなくなり、そのグリフ記述においてグリフの形状だけでなく色・透過を指定することも可能になります。

スコープ **ページ・フォント**。このメソッドは**グリフ**スコープを開始させます。対応する *PDF_end_glyph()* と必ずペアにして呼び出す必要があります。*PDF_begin_font()* で ***widthsonly=true*** にしているときは、*PDF_begin_glyph_ext()* と *PDF_end_glyph()* の間の API メソッドの呼び出しはすべて無視されます。

表 4.8 PDF_begin_glyph_ext( ) のオプション

オプション	説明
bounding-box	（4 個の float のリスト。オプショナルですが、推奨）グリフの外接枠の左下隅と右上隅の座標。デフォルト：{0 0 0 0}
colorized	（論理値）true にすると、このグリフ記述で色・透過を指定することが可能。false にすると、このグリフはその形状しか指定できず、それが配置される時点でのカレントの塗り色か描線色で描かれます。デフォルト：PDF_begin_font( ) の colorized オプションが指定されているならその値、ないなら false
glyphname	（文字列。uv=-1 の場合には必須。uv=0 の場合には無視されます）グリフの一意名。Unicode 0 を持つグリフ 0 の名前は強制的に .notdef となります。デフォルト：与えられた Unicode 値に従った AGL 名
width	（float。必須）グリフの幅を、そのフォントマトリックスによって指定された通りのグリフ座標系で表したもの。

C++ Java C# ***void end_glyph()***
Perl PHP ***end_glyph()***
C ***void PDF_end_glyph(PDF *p)***

Type 3 フォントのためのグリフ定義を終了させます。

スコープ **グリフ**。このメソッドは**グリフ**スコープから**フォント**スコープへ移行します。対応する *PDF_begin_glyph_ext()* と必ずペアにして呼び出す必要があります。

4.5 ユーザー定義 8 ビットエンコーディング

C++ Java C# ***void encoding_set_char(String encoding, int slot, String glyphname, int uv)***

Perl PHP ***encoding_set_char(string encoding, int slot, string glyphname, int uv)***

C ***void PDF_encoding_set_char(PDF *p, const char *encoding, int slot, const char *glyphname, int uv)***

カスタムの 8 ビットエンコーディングに、グリフ名か Unicode 値またはその両方を追加します。

encoding エンコーディングの名前。これは、*PDF_load_font()* で使う必要のある名前です。これは、あらゆる組み込みエンコーディングとも、以前に使ったすべてのエンコーディングとも異なっている必要があります。

slot 定義したいキャラクターの位置を *0* ≦ *slot* ≦ *255* で指定します。1 つのエンコーディング内では、各スロットはそれぞれ 1 回だけ定義することができます。

glyphname キャラクターの名前。

uv キャラクターの Unicode 値。

詳細 このメソッドは、非標準 8 ビットエンコーディングを処理しなければならない特殊な応用においてのみ必要となります。複数回呼び出して、エンコーディング内の最大 256 個のキャラクタースロットを定義することができます。そのエンコーディングを初めて使う前であれば、キャラクターを追加していくことができます。すべてのコード点を指定する必要はなく、未定義のスロットには *.notdef* と U+0000 が書かれます。

1 つのエンコーディングの中で、各グリフ名 /Unicode 値を 1 度だけ与えることを強く推奨します（*.notdef*/U+0000 を除いて）。スロット 0 が使用される場合には、それは *.notdef* キャラクターを内容とするべきです。

定義したエンコーディングは、カレントの**オブジェクト**スコープが終わるまで使えます。

スコープ 任意

5 テキストと表の組版

この章の API メソッド：

- *PDF_fit_textline()*
- *PDF_info_textline()*
- *PDF_add_textflow()*
- *PDF_create_textflow()*
- *PDF_fit_textflow()*
- *PDF_info_textflow()*
- *PDF_delete_textflow()*
- *PDF_add_table_cell()*
- *PDF_fit_table()*
- *PDF_info_table()*
- *PDF_delete_table()*

5.1 テキスト行による一行テキスト

C++ Java C# ***void fit_textline(String text, double x, double y, String optlist)***

Perl PHP ***fit_textline(string text, float x, float y, string optlist)***

C ***void PDF_fit_textline(PDF*p, const char *text, int len, double x, double y, const char *optlist)***

一行のテキストを位置 *(x, y)* に、さまざまなオプションに従って配置します。

text （内容文字列）ページ上に配置したいテキスト。

len （C 言語バインディングのみ）*text* が UTF-16 文字列のときの長さ（バイト単位）。*len=0* にすると null 終了文字列を与える必要があります。

x・y テキストをさまざまなオプションに従って配置したい参照点の座標を、ユーザー座標系で指定します。はめ込みアルゴリズムの説明は 135 ページ「6.1 オブジェクトのはめ込み」を参照。

optlist フォント・テキスト・組版オプションを指定したオプションリスト。以下のオプションが使えます：

- 一般オプション：*errorpolicy*（表 1.5 参照）
- 表 4.1 で暗黙的フォント読み込み（すなわち、テキスト書式グループの *font* オプションを与えない）の場合に従ったフォント読み込みオプション：
  ascender・autosubsetting・capheight・colormode・descender・embedding・encoding・fallbackfonts・fontname・fontstyle・keepnative・linegap・metadata・readfeatures・replacementchar・subsetlimit・subsetminsize・subsetting・unicodemap・vertical・xheight
- 表 4.5 に従ったテキストフィルターオプション：
  actualtext・charref・escapesequence・glyphcheck・normalize・textformat
- 表 4.6 に従ったテキスト書式オプション：
  charspacing・dasharray・decorationabove・fakebold・fillcolor・font・fontsize・gstate・horizscaling・inittextstate・italicangle・kerning・leading・overline・shadow・

strikeout・*strokecolor*・*strokewidth*・*textrendering*・*textrise*・*underline*・*underlineposition*・*underlinewidth*・*wordspacing*

- 表 5.1 に従ったテキスト行組版のためのオプション：*justifymethod*・*leader*・*textpath*・*xadvancelist*
- 表 5.4 に従ったシェーピング・タイポグラフィーオプション：*features*・*language*・*script*・*shaping*
- 表 6.1 に従ったはめ込みオプション：*alignchar*・*blind*・*boxsize*・*fitmethod*・*margin*・*matchbox*・*orientate*・*position*・*rotate*・*stamp*・*showborder*・*shrinklimit*

詳細　*inittextstate=false*（これがデフォルトです）にすると、カレントのテキスト・グラフィックステートオプション群が、オプションで明示的にオーバーライドしない限り、テキスト出力の書式の制御に用いられます。

inittextstate=true にすると、テキスト・グラフィックステートオプション群のデフォルト値が、オプションで明示的にオーバーライドしない限り、テキスト出力の書式の制御に用いられます。テキスト行オプション群は、今回の *PDF_fit_textline()* への呼び出しより後に生成する出力では効力を持ちません。

カレントのテキスト・グラフィックステートは、このメソッドによって変更を受けません（特に、カレントフォントは影響を受けません）。ただし、*textx/texty* オプションは、生成したテキスト出力の末尾位置へ変更されます。その末尾グリフの後では *charspacing* パラメーターが考慮されません。

PDF_continue_text() に対する参照点は、テキストの先頭には設定されません。*PDF_fit_textline()* の後に *PDF_continue_text()* を使うためには、開始点を *PDF_info_textline()* と *startx/starty* キーワードで取得して、テキスト位置を *PDF_set_text_pos()* で設定する必要があります。

スコープ　ページ・パターン・テンプレート・グリフ。

表 5.1　PDF_fit_textline( ) の追加オプション

オプション	説明
justifymethod	（キーワードのリスト。fitmethod=auto と stamp=none の場合にのみ意味を持ちます。boxsize を必要とします。縦書きでは無視されます）テキストがはめこみ枠からはみ出さないようにするために、文字サイズを変えずに 1 個ないし複数の組版方式を適用します。以下のキーワードのうちの 1 個ないし複数を与えることができます。複数のキーワードが存在する場合には、調整は以下の順に適用されます：wordspacing、charspacing、horizscaling（デフォルト：none）： ***charspacing*** 適切な charspacing 値を用いて調整。 ***horizscaling*** 適切な horizscaling 値を用いて調整。 ***none*** 調整なし ***wordspacing*** 適切な wordspacing 値を用いて調整。テキストが空白キャラクターを含んでいない場合には、wordspacing 調整は適用されません。
leader	（オプションリスト。boxsize を指定していない場合、または枠の幅が 0 の場合には無視されます）隙間埋めテキスト（点リーダ等）と組版オプション群。リーダは、テキスト枠の端とテキストの間に繰り返し挿入されます。 使えるサブオプションの一覧は表 5.3 を参照。デフォルト：リーダなし

表 5.1 PDF_fit_textline( ) の追加オプション

オプション	説明
textpath	（オプションリスト）テキストをパスに沿って描きます。パスの終端からはみ出したテキストは表示されません。showborder=true にすると、パスがカレントの線幅と描線色で描かれます。表 5.2 に挙げるオプションに加え、PDF_draw_path( ) の以下のオプションを使えます： align・attachmentpoint・boxsize・fitmethod・orientate・scale（表 6.1 参照） close・round・subpaths（表 7.7 参照） bboxexpand・boundingbox（表 7.7 参照） PDF_fit_textline( ) の以下のオプションは、パス上テキストについては意味が変わります： ***matchbox*** 各グリフに対して個別の枠が生成されます。 ***position*** 1 番目の値は、パスの長さに対するテキストの相対的な開始位置（left/center/right）を指定します。テキストがパスより長い場合は、テキストはつねに startoffset から始まります。2 番目の値は、パスに対する各グリフの相対的な縦位置を、すなわちグリフ枠のどの部分がパスに接するか（bottom/center/top）を指定します。 ***rotate*** 各グリフの回転角。 以下のはめ込み枠関連のオプションは無視されます： boxsize・margin・fitmethod・orientate・alignchar・showborder・stamp・leader カーニングと、日中韓レガシーエンコーディングによるテキストは、パス上テキストでは対応していません。
xadvancelist	（float のリスト。shaping=true の場合には無視されます）テキスト内のグリフ群の変位幅。リスト長は、テキスト内のグリフの数以下かそれに等しくする必要があります。この xadvance 値は標準のグリフ幅をオーバーライドします。多くの場合、字間の変更は charspacing オプションを用いるほうが容易に実現できます。

表 5.2 PDF_fit_textline( ) の textpath オプションの追加サブオプション

オプション	説明
path	（パスハンドル。必須）テキスト出力のためのベースラインとして使いたいパス。デフォルトでは、テキストはパスの左側に配置され、パスはテキストのベースラインとなります。しかし、position オプションの 2 番目のキーワードを top にすると、テキストはパスの反対側に配置され、テキストの上端がパスに接します。PDF_fit_textline( ) の引数 x・y がパスの参照点として用いられます。
startoffset	（float またはパーセント値）テキストのパス上の開始点の変位を、ユーザー座標で、またはパスの長さに対するパーセント値で指定します。デフォルト：0
tolerance	（float またはパーセント値）パスの末尾グリフがどのくらいパスからはみ出てもよいかを指定します。値は、ユーザー座標で、または文字サイズに対するパーセント値で指定します。デフォルト：25%

表 5.3 PDF_fit_textline( ) ・ PDF_add/create_textflow( ) と、PDF_create_textflow( ) のインラインオプションの、leader オプションのサブオプション

オプション	説明
フォント読み込みオプション群	フォントを暗黙的に（すなわち、font オプションではなく fontname オプションを用いて）指定するときは、表 4.1 に従ったすべてのフォント読み込みオプションをサブオプションとして与えることができます。
alignment	（キーワード 1 個または 2 個）1 番目のキーワードは、はめ込み枠左端とテキスト行の間のリーダの整列を指定します。2 番目のキーワードは、テキスト行とはめ込み枠右端の間のリーダの整列を指定します。キーワードを 1 個だけ指定すると、テキスト行とはめ込み枠右端の間のリーダに対して使われます。使えるキーワード（テキスト行の場合のデフォルト：{none grid}。テキストフローの場合のデフォルト：grid）： ***center*** テキスト行：リーダは、テキスト行とはめ込み枠端の間で両端揃えされます。テキストフロー：リーダは、最後の部分テキスト（あるいはテキストがないときは行の先頭）とタブ位置（あるいはタブがないときは行の末尾）との間で中央揃えされます。 ***grid*** PDFlib は、テキスト行の左または右へ、リーダテキストの幅の半分の倍数ごとにグリッドを仮想し、リーダテキストの位置を、その次のグリッドに吸着させます。これにより、テキスト行とリーダテキストの間に、リーダテキストの幅の最大 50% のアキが生じます。 ***justify*** テキスト行：リーダは、適切な字間を適用することにより、テキスト行とはめ込み枠端の間で両端揃えされます。テキストフロー：リーダは、適切な字間を適用することにより、最後の部分テキスト（あるいはテキストがないときは行の先頭）とタブ位置（あるいはタブがないときは行の末尾）との間で両端揃えされます。 ***left*** それぞれリーダは、はめ込み枠左端から、またはテキスト行末尾から開始して繰り返されます。これを指定するとそれぞれ、テキスト行先頭に、またははめ込み枠右端に隙間が生じる可能性があります。 ***none*** リーダなし ***right*** それぞれリーダは、はめ込み枠右端から、またはテキスト行先頭から開始して繰り返されます。これを指定するとそれぞれ、テキスト行末尾に、またははめ込み枠左端に隙間が生じる可能性があります。
fillcolor	（色）リーダの色。デフォルト：テキスト行の色
font	（フォントハンドル）リーダに対して使いたいフォントのハンドル。デフォルト：テキスト行のフォント
fontsize	（文字サイズ）リーダのサイズ。デフォルト：テキスト行の文字サイズ
text	（内容文字列）リーダに使いたいテキスト。デフォルト：U+002E「.」（ピリオド）
yposition	（float またはキーワード）リーダの縦位置を、ベースラインに対して相対的に、数値で、またはキーワード fontsize・ascender・xheight・baseline・descender・textrise のうちのいずれか 1 つで。デフォルト：baseline

C++ Java C# ***double info_textline(String text, String keyword, String optlist)***

Perl PHP ***float info_textline(string text, string keyword, string optlist)***

C ***double PDF_info_textline(PDF *p, const char *text, int len, const char *keyword, const char *optlist)***

テキスト行の組版を、出力を生成せず仮想的に行なって、その結果のメトリックを取得します。

text （内容文字列）テキスト行の内容。

表 5.4 PDF_fit/info_textline( )・PDF_add/create_textflow( )・PDF_fill_textblock( ) に対するシェーピング・タイポグラフィーオプション

オプション	説明
features	（キーワードのリスト）script・language オプションに従って、OpenType フォントのどのタイポグラフィー機能がテキストに適用されるかを指定します。フォント内に存在しない機能は無視されます： ***_none*** フォント内の機能を一切適用しません。 **<名称>** 機能を有効化するために、その4文字のOpenType名を与えます。よく使われる機能名は aalt・liga・ital・tnum・smcp・swsh・zeroです。対応しているすべての機能の名前と説明の完全な一覧がPDFlibチュートリアルにあります。 ***no*****<名称>** 機能名の前に接頭辞noを付けると（例：noliga）この機能が無効化されます。 以下の機能はデフォルトで有効化されます：calt・ccmp・clig・liga・locl。機能 vrt2・vert は、縦書きのフォントに対しては自動的に有効化されます。
language	（キーワード。script を与えている場合にのみ意味を持ちます）指定した言語に従ってテキストを処理。features・shaping オプションに対して意味を持ちます。キーワードの完全な一覧は PDFlib チュートリアルにあります。例：ARA（アラビア語）・JAN（日本語）・HIN（ヒンディー語）。デフォルト：_none（言語未定義）
script	（キーワード。shaping=true の場合には必須）指定した用字系に従ってテキストを処理。features・shaping・advancedlinebreak オプションに対して意味を持ちます（デフォルト：_none）： ***_auto*** （シェーピングに対してのみ意味を持ちます）テキストの多数が属する用字系を選択。ただし latn・_none は無視されます。 ***_none*** 未定義の用字系。 複雑用字系のシェーピングに対しては以下の用字系を使えます： ***arab・hebr*** 中東用字系 ***sinh・tibt*** 中央アジア用字系 ***khmr・thai*** 東南アジア用字系 ***hang・hani*** 東アジア用字系 ***beng, deva, gujr, knda, mlym, orya, guru, taml, telu*** インド用字系 このほか、OpenType 機能の処理を制御するために、latn・cyrl・grek など OpenType 仕様に従ったスクリプトタグを使うこともできます。 使える用字系キーワードの一覧が PDFlib チュートリアルにあります。
shaping	（論理値）true にすると、script・language オプションに従って、複雑用字系のシェーピングと双方向組版がテキストに適用されます。script オプションが _none 以外の値を持つ必要があり、かつフォントが特定の条件に従っている必要があります（PDFlib チュートリアル参照）。シェーピングは、テキストフロー内の右書きテキストでは利用できません（テキスト行内のみ）。デフォルト：false

len （C 言語バインディングのみ）テキストの長さをバイト単位で表すか、または null 終了文字列では 0。

keyword ほしい情報を指定したキーワード：

- 表 6.3 に従った、オブジェクトはめ込みの結果をクエリーするためのキーワード：
  boundingbox・fitscalex・fitscaley・height・objectheight・objectwidth・width
- 表 5.5 に従ったさらなるキーワード：
  angle・ascender・capheight・descender・endx・endy・missingglyphs・pathlength・

perpendiculardir・*replacedchars*・*righttoleft*・*scriptlist*・*startx*・*starty*・*textwidth*・*textheight*・*unmappedchars*・*wellformed*・*writingdirx*・*writingdiry*・*xheight*

optlist PDF_fit_textline( ) のオプションを指定したオプションリスト。要求したキーワードに関係のないオプションは、警告を出さずに無視されます。

戻り値 *keyword* で要求した何らかのテキストメトリック値の値。

詳細 このメソッドは、与えたオプションに従ってテキストを配置するために必要な計算をすべて行いますが、実際にページ上に出力を作成はしません。テキスト参照点は {0 0} と見なされます。

errorpolicy=return の場合、このメソッドはエラー時に 0 を返します。*errorpolicy=exception* の場合、このメソッドはエラー時に例外を発生させます（*wellformed* キーワードに対しても）。

スコープ **オブジェクト**以外の任意

表 5.5 PDF_info_textline( ) のキーワード

キーワード	説明
angle	ベースラインの回転角を度単位で表したもの、すなわちテキストの回転
ascender ***capheight*** ***descender***	アセンダー・キャップハイト・ディセンダー。それぞれ、同名のタイポグラフィー特性をユーザー座標で表したもの
endx*・*endy	論理的なテキスト終了位置の x・y 座標をユーザー座標で表したもの
missingglyphs	（パス上テキストでのみ可）パス上に配置できなかったグリフの数。
pathlength	（パス上テキストでのみ可）テキストで覆われているパスの、その開始点から終了点までの長さ。この値は、PDF_fit_textline( ) が blind モードで呼びだされた場合でもクエリーできます。この値は、パスをさらなるテキストでラベリング継続するために PDF_fit_textline( ) の startoffset オプションに対して使用することもできます。
perpendiculardir	writingdir に垂直な単位ベクトル。標準的な横書きテキストならこれは (0, 1)、縦書きテキストなら (1, 0) でしょう
replacedchars	カレントエンコーディング内のコードに、またはフォント内のグリフにマップできなかったので、タイポグラフィー的に類似のキャラクター群の内蔵リスト内のわずかに異なるグリフに、または予備フォント内のグリフに置き換えられたキャラクターの数。この値が 0 以外になるのは、glyphcheck=replace にしたときだけです。
righttoleft	テキストのグローバルな出力方向が右書きなら 1、左書きまたは縦書きテキストならば 0。このグローバル方向は先頭キャラクター群に基づいて、かつ、テキスト内に方向マーカーが存在していれば（例：U+202D または &LRO;、左書き上書き）それにも基づいて決定されます。
scriptlist	テキスト内のすべての用字系の名前のスペース区切りのリストを内容として持つ文字列。これはテキストシェーピングを用意するのに有用でしょう。用字系名群は頻度順に降順で並べ替えられています。用字系 _none と _latn は、シェーピングに関係がないので無視されます。テキスト内に _none と _latn のキャラクターだけが存在しているときは、−1 が返されます。
startx*・*starty	論理的なテキスト開始位置の x・y 座標をユーザー座標系で表したもの
textwidth*・*textheight	テキストの幅と高さ。この高さは範囲枠の boxheight の定義に従います。その定義のデフォルトは {capheight none} です。パス上テキストではどちらの値も 0 です。

表 5.5 PDF_info_textline( ) のキーワード

キーワード	説明
unknownchars	glyphcheck=none の場合：スキップされたキャラクターの数。この数の中には、解決できなかった文字参照と、カレントエンコーディング内のコードに、またはフォント内のグリフにマップできなかったキャラクターとが含まれています。 glyphcheck=replace の場合：指定した置き換えキャラクター（replacementchar オプション）で置き換えられたキャラクターの数。この数の中には、カレントエンコーディング内のコードに、またはフォント内のグリフにマップできなかったキャラクターと、タイポグラフィー的に類似のキャラクターに置き換えることができなかったキャラクターとが含まれています。
unmappedchars	スキップされたか置き換えられたキャラクターの数。すなわち、replacedchars と unknownchars の合計。
wellformed	選択されたフォント・エンコーディング（適用可能なら textformat も）に従ってテキストが整えられているなら 1、そうでないなら 0。
writingdirx ***writingdiry***	(startx, starty) から (endx, endy) への単位ベクトルを記述した、優勢な筆記方向（すなわち行内のテキスト進行方向）の x・y 座標、左書きの横書きテキストなら値 は (1, 0)、縦書きテキストなら (0, -1)、右書きの横書きテキストなら (-1, 0)。筆記方向は、shaping・vertical オプションと、テキストの方向性特性群にもとづいて決定されます。
xheight	x ハイトをユーザー座標で表したもの

5.2 テキストフローによる複数行テキスト

C++ Java C# ***int add_textflow(int textflow, String text, String optlist)***

Perl PHP ***int add_textflow(int textflow, string text, string optlist)***

C ***int PDF_add_textflow(PDF *p, int textflow, const char *text, int len, const char *optlist)***

テキストフローオブジェクトを作成するか、または既存のテキストフローにテキストと明示オプションを追加します。

textflow 以前の *PDF_create_textflow()* または *PDF_add_textflow()* への呼び出しによって返されたテキストフローハンドルか、または -1（PHP では 0）で新規テキストフローを作成します。

text （内容文字列）テキストフローの内容。*inlineoptions* オプションを設定している場合には、テキストにインラインオプションを入れることもできます。このテキストは空にしてはいけません。

len （C 言語バインディングのみ）テキストの長さをバイト単位で表すか、または null 終了文字列では 0。

optlist 以下のテキストフローオプションを指定したオプションリスト：

- 一般オプション：*errorpolicy*（表 1.5 参照）
- 表 4.1 で暗黙的フォント読み込み（すなわち、テキスト書式グループの *font* オプションを与えない）の場合に従ったフォント読み込みオプション：
  ascender・*autosubsetting*・*capheight*・*colormode*・*descender*・*embedding*・*encoding*・*fallbackfonts*・*fontname*・*fontstyle*・*keepnative*・*linegap*・*metadata*・*readfeatures*・*replacementchar*・*subsetlimit*・*subsetminsize*・*subsetting*・*unicodemap*・*vertical*・*xheight*
- 表 4.5 に従ったテキストフィルターオプション：
  charref・*escapesequence*・*glyphcheck*・*normalize*・*textformat*
- 表 4.6 に従ったテキスト書式オプション：
  charspacing・*dasharray*・*decorationabove*・*fakebold*・*fillcolor*・*font*・*fontsize*・*gstate*・*horizscaling*・*inittextstate*・*italicangle*・*kerning*・*leading*・*overline*・*shadow*・*strikeout*・*strokecolor*・*strokewidth*・*textrendering*・*textrise*・*underline*・*underlineposition*・*underlinewidth*・*wordspacing*
- 表 5.4 に従ったシェーピング・タイポグラフィーオプション：
  features・*language*・*script*・*shaping*
- 表 5.6 に従ったテキストフロー組版のためのオプション：
  alignment・*avoidemptybegin*・*fixedleading*・*hortabmethod*・*hortabsize*・*lastalignment*・*leader*・*leftindent*・*minlinecount*・*parindent*・*rightindent*・*ruler*・*tabalignment*
- 表 5.7 に従った改行アルゴリズムを制御するためのオプション：
  adjustmethod・*advancedlinebreak*・*avoidbreak*・*locale*・*maxspacing*・*minspacing*・*nofitlimit*・*shrinklimit*・*spreadlimit*
- 表 5.8 に従ったコマンドオプション：
  comment・*inlineoptions*・*mark*・*matchbox*・*nextline*・*nextparagraph*・***restore***・*resetfont*・*return*・*save*・*space*

- 表 5.9 に従ったテキスト意味付けオプション：
  charclass・*charmapping*・*hyphenchar*・*tabalignchar*
- 表 5.10 に従ったタグ付けオプション：*tagbegin*・*tagend*

戻り値 テキストフローハンドル。テキストフロー関連のメソッドへの呼び出しで使えます。ハンドルは、カレントの**文書**スコープの終わりか、またはこのハンドルを指定して *PDF_delete_textflow()* を呼ぶまで有効です。

textflow 引数を −1（PHP では 0）にすると、新規テキストフローが作成されて、そのハンドルが返されます。そうでないなら、*textflow* 引数で与えたハンドルが返されます。デフォルトではこのメソッドは、エラーが起きたときは −1（PHP では 0）を返します。しかしこの動作は、*errorpolicy* オプションで変えることもできます。エラーが起きたときは、*textflow* 引数で与えたハンドルは、以後のメソッドへの呼び出しではもう使うことができません（−1 以外だったときの *PDF_delete_textflow()* を除き）。

詳細 このメソッドは、与えられたテキストを処理して、そこから内部データ構造を作成します。以後にフォーマッターが使うテキストの各部分（単語等）を決定し、テキストを可能なら Unicode へ変換し、改行可能位置を決定し、フォントとテキストのオプション群にもとづいてテキストの各部分の幅を算出します。

PDF_create_textflow() の場合は、1 度の呼び出しでテキスト内容とオプションをすべて与える必要がありますが、このメソッドはそれとは異なり、何度かの呼び出しに分けてテキスト内容とオプションを与えたいときに有用です。与えられた *text* と *optlist* を、新規または既存のテキストフローに追加します。*optlist* で指定されたオプションは、*text* を処理する前に評価されます。

textflow=-1（PHP では 0）にすると、このメソッドはほとんど *PDF_create_textflow()* と同等になります。ただし *PDF_create_textflow()* とは違って、このメソッドは *text* の中のインラインオプションを検索しません。ですので、インラインオプションリストの開始キャラクターを再定義したり、インラインオプションのテキストの長さを指定したりする必要はありません（非 Unicode テキスト・UTF-16 テキストの場合であっても）。

このメソッドは、与えたテキストとオプション群を前処理しますが、生成 PDF 文書に出力を作成せず、ただテキストを用意します。出力を作成するには、*PDF_fit_textflow()*・*PDF_fit_table()*・*PDF_fill_textblock()* のいずれかを使って、この前処理したテキストフローのハンドルを指定します。

デフォルトでは、キャラクター U+000B（VT）・U+2028（LS）・U+000A（LF）・U+000D（CR）・CRLF・U+0085（NEL）・U+2029（PS）・U+000C（FF）は、ニューラインを強制します。これらの制御キャラクターは、*encoding=builtin* で読み込んだ記号フォントに対しては解釈されません。これらは VT・LS を除いてすべて、改段落を強制します（すなわち、そこで *parindent* オプションが効きます）。FF はただちに、カレントはめ込み枠へのテキストのはめ込み処理を中止させます（メソッド *PDF_fit_textflow()* は文字列 *_nextpage* を返します）。

水平タブキャラクター（HT）は、後続するテキストに対して新しい開始位置を設定します。これの詳細は、*hortabmethod*・*hortabsize* オプションで制御されます。

ソフトハイフンキャラクター（SHY）は、そのソフトハイフンの後に改行が来るときは、*hyphenchar* オプションで指定されているキャラクターに置き換えられます。

縦書きには対応していません。

スコープ **オブジェクト以外の任意**

表 5.6 PDF_add/create_textflow( ) と、PDF_create_textflow( ) のインラインオプションと、テキストフローブロックに対する PDF_fill_textblock( ) の、追加の組版オプション

オプション	説明
alignment	（キーワード）段落内の行の整列（デフォルト：left）： ***left*** 左揃え。leftindent+parindent（段落の先頭行）と、leftindent（それ以外のすべての行）から ***center*** 中央揃え。leftindent から rightindent まで ***right*** 右揃え。rightindent まで ***justify*** 両端揃え 値 alignment=justify は、nextline オプションを含む行に対しては無視されます。nextparagraph を含む行の整列は、alignment オプションによって制御されず、オプション lastalignment によって制御されます。
avoid-emptybegin	（論理値）true にすると、はめ込み枠の先頭の空行群は削除されます。デフォルト：false
fixedleading	（論理値）true にすると、それぞれ行の中で見つかった最初の行送り値が使われます。そうでなければ、行の中のすべての行送り値のうち最大のものが使われます。PDF_fit_textflow( ) の wrap オプションを、または matchbox オプションの createwrapbox サブオプションを使ってテキストを輪郭に回り込ませる場合は、fixedleading は true を強制されます。デフォルト：false
hortabmethod	（キーワード）テキストの水平タブの処理。算出される位置が、カレントテキスト位置より左のときは、そのタブは無視されます（デフォルト：relative）： ***relative*** 位置を、hortabsize で指定している量だけ進めます。 ***typewriter*** 位置を、hortabsize の次の倍数まで進めます。 ***ruler*** 位置を、ruler オプションの n 番目のタブ値まで進めます。ここで n は、その行の中でこれまでに見つかったタブの数です。n がタブ位置の数より大きいときは、relative 方式を適用します。
hortabsize	（float またはパーセント値）水平タブの幅[1]。その解釈は、hortabmethod オプションに依存します。デフォルト：7.5%
lastalignment	（キーワード）段落内の最終行の整列。alignment オプションのすべてのキーワードのほか、以下のキーワードも使えます（デフォルト：auto）： ***auto*** alignment オプションの値を使います。ただしそれが justify のときは left が使われます。
leader	（オプションリスト）繰り返し挿入したい隙間埋めテキスト（点リーダ等）。リーダは、次のタブ位置か、またはタブが得られないときは行末まで挿入されつづけます。リーダは複数行にわたることはありません。使えるサブオプションの一覧は表 5.3 を参照。デフォルト：リーダなし
leftindent	（float またはパーセント値）テキスト行の左インデント[1]。leftindent を行内で指定しているときは、決定される位置がカレントテキスト位置より左なら、このオプションはこの行では無視されます。デフォルト：0
minlinecount	（整数）はめ込み枠の最後の段落の最少行数。行数がこれより少ないときは、その段落は次のはめ込み枠に配置されます。値 2 にすれば、段落の 1 行だけがはめ込み枠の最後に来てしまう状態（「オーファン」）を防げます。デフォルト：1
parindent	（float またはパーセント値）段落の先頭行の左インデント[1]。この値が leftindent に加算されます。このオプションを行内で指定すると、タブのように動作します。デフォルト：0
rightindent	（float またはパーセント値）テキスト行の右インデント[1]。デフォルト：0
ruler	（float かパーセント値のリスト）hortabmethod=ruler の場合の絶対タブ位置のリスト[1]。リストには、最大 32 個の非負の項目を昇順で入れることができます。デフォルト：hortabsize の整数倍

表 5.6 PDF_add/create_textflow( ) と、PDF_create_textflow( ) のインラインオプションと、テキストフローブロックに対する PDF_fill_textblock( ) の、追加の組版オプション

オプション	説明
tabalignment	（キーワードのリスト。hortabmethod=ruler の場合にのみ可）タブ位置の整列。このリストは 32 項目までを内容とすることができます。テキスト内で行あたり 32 個を超える水平タブが現れる場合には、このリストは最後の値を用いて拡張されます。リスト内の各項目はそれぞれ、ruler オプションの各項目の整列を定義します。デフォルト：left。 ***center*** テキストをタブ位置で中央揃えします。 ***decimal*** 最初に現れる tabalignchar をタブ位置で左揃えします。tabalignchar が見つからないときは、右揃えにします。 ***left*** テキストをタブ位置で左揃えします。 ***right*** テキストをタブ位置で右揃えします。

1. ユーザー座標で、またははめ込み枠の幅に対するパーセント値で指定します

表 5.7 PDF_add/create_textflow( ) と、PDF_create_textflow( ) のインラインオプションと、テキストフローブロックに対する PDF_fill_textblock( ) の、改行アルゴリズムを制御するための追加のオプション

オプション	説明
adjustmethod	（キーワード）minspacing・maxspacing オプションで指定している制限の中で単語間を縮めたり拡げたりしても、部分テキストが行に収まりきらないときに、行の調整に使わせたい方式。デフォルト：auto。 ***auto*** 次の方式を順に適用します：shrink・spread・nofit・split。 ***clip*** nofit において、はめ込み枠の右端（rightindent オプションを考慮）からはみ出した部分を切り落とします。 ***nofit*** 最後の単語を次の行へ送ります。ただし、残される（短い）行が、nofitlimit オプションで指定しているパーセント値よりも短くならない場合に限ります。両端揃えの段落でも若干がたついて見えることがあります。 ***shrink*** 単語が行に収まりきらないときに、テキストを shrinklimit の制限内で圧縮します。それでも収まらないときは、nofit 方式を適用します。 ***split*** 最後の単語を次の行へ送らずに、枠内の最後のキャラクターの後で強制的に分割します。テキストフォントの場合はハイフンキャラクターを挿入しますが、記号フォントの場合か、または hyphenchar=none のときは挿入しません。 ***spread*** 最後の単語を次の行へ送り、残された（短い）行を両端揃えするよう、単語内の字間を spreadlimit の制限内で拡げます。それでも両端揃えできないときは、nofit 方式を適用します。
advanced-linebreak	（論理値）複雑用字系に対して必要な高度な改行アルゴリズムを有効にします。これはタイ文字のように、単語間の境界を表すのにスペースキャラクターを用いない用字系での改行に必要です。locale・script オプションは効力を持ちます。デフォルト：false
avoidbreak	（論理値）true にすると、avoidbreak を false にリセットするまで、改行機会（スペースキャラクター等での）が無視されます。強制的な改行（ニューライン等での）と、adjustmethod によって定義される方式は、この場合にも実行されます。特に、adjustmethod=split はこの場合にもハイフネーションを生成します。デフォルト：false

表 5.7 PDF_add/create_textflow( ) と、PDF_create_textflow( ) のインラインオプションと、テキストフローブロックに対する PDF_fill_textblock( ) の、改行アルゴリズムを制御するための追加のオプション

オプション	説明
locale	（キーワード）advancedlinebreak=true の場合に、用字系特有の改行方式で用いられるロケール。キーワードは、以下の 1 個ないし複数の構成要素から成り、オプショナルな構成要素は下線キャラクター「_」で区切られます（その文法は、NLS/POSIX のロケール ID とは若干異なっています）。 ▸（必須）ISO 639-2 に従った、小文字 2 文字または 3 文字の言語コード。例：en（英語）・de（ドイツ語）・ja（日本語）。これは language オプションとは異なっています。 ▸（オプション）ISO 3166 に従った、大文字 2 文字の国コード。例：DE（ドイツ）・CH（スイス）・GB（イギリス）。 キーワード _none は、ロケール独自の処理が行われないことを指定します。 ロケールを指定することは、いくつかの用字系では、高度な改行のために必要です。デフォルト：_none 例：tha・de_DE・en_US・en_GB
maxspacing ***minspacing***	（float またはパーセント値。行がスペースキャラクター U+0020 を少なくとも 1 個含み、かつ alignment=justify の場合にのみ意味を持ちます）単語間隔の最大値と最小値（ユーザー座標で、またはスペースキャラクターの幅に対するパーセント値で指定します）。算出される単語間隔が、与える値で制限されます（ただし wordspacing オプションはさらに加算されます）。デフォルト：minspacing=50%、maxspacing=500%
nofitlimit	（float またはパーセント値。alignment=justify の場合にのみ意味を持ちます）nofit 方式にしているときの、行の長さの下限[1]。デフォルト：75%。
shrinklimit	（パーセント値）adjustmethod=shrink の場合の、テキストを縮める下限。算出される縮小率が、与える値で制限されますが、horizscaling オプションを掛け算されます。デフォルト：85%
spreadlimit	（float またはパーセント値）spread 方式の場合の字間の上限[1]。算出される字間が、charspacing オプションの値に加算されます。デフォルト：0

1. ユーザー座標で、またははめ込み枠の幅に対するパーセント値で指定します

表 5.8 PDF_add/create_textflow( ) と、PDF_create_textflow( ) のインラインオプションと、テキストフローブロックに対する PDF_fill_textblock( ) の、追加のコマンドオプション

オプション	説明
comment	（文字列）無視される任意のテキスト。オプションリストかマクロに注釈を付けるのに有用です
inlineoptions	（論理値。PDF_create_textflow( ) の中と、PDF_add_textflow( ) のインラインオプションの中では無視されます）true にすると、与えたテキストの中でインラインオプションが検索されます。デフォルト：false
mark	（整数）与える番号を、マークとして内部的に格納します。もっとも最近に格納したマークは、後で PDF_info_textflow( ) と lastmark キーワードで取得することができます。これは、テキストのどの部分がページにもう配置されたかを知るのに有用でしょう。
matchbox	（オプションリスト）表 6.4 に従った、範囲枠を作成するためのオプションリスト
nextline	（論理値）改行を強制します。キャラクター U+000B・U+2028 のいずれか 1 つと等価です。オプション alignment=justify と lastalignment は、この nextline オプションを含む行にはいかなる影響も及ぼしません。このオプションを、matchbox サブオプション end と一緒のインラインオプションリストの中で使用するべきではありません。
nextparagraph	（論理値）改段落を強制します。キャラクター U+000A、U+000D、U+000D と U+000A、U+0085、U+2029、U+00FF のいずれか 1 つと等価です。この nextparagraph オプションを含む行の整列は、オプション lastalignment によって決定され、オプション alignment は無視されます。

表 5.8 PDF_add/create_textflow( ) と、PDF_create_textflow( ) のインラインオプションと、テキストフローブロックに対する PDF_fill_textblock( ) の、追加のコマンドオプション

オプション	説明
resetfont	（論理値）font と fontsize を、カレントの設定と違っていた（フォントか文字サイズのどちらかが）もっとも最近の値へ戻します。これは、イタリックのテキストのように一時的にフォントを変えたのを、元に戻したいときに便利でしょう。font オプションはこのオプションよりも優先されます。このコマンドは、任意のフォント関連オプションについて最初の設定と違う設定を初めて行なった後にのみ意味を持ち、そうでないときは無視されます。
restore	（論理値）true にすると、対応する save コマンドによって保存されたすべてのテキスト・テキストフローオプションの値が復帰されます。保存 / 復帰ペア内で作成された範囲枠は、復帰後、保持されます。デフォルト：false
return	（文字列。先頭にアンダースコアキャラクター _ をつけてはいけません）与える文字列を戻り値として、PDF_fit_textflow( ) を抜けます。新たな行が自動的に作成されます。
save	（論理値）true にすると、タグ付けオプションと非ステートオプション charmapping・nextline・nextparagraph・resetfont・return・space・textlen 以外のすべてのテキスト・テキストフローオプションの値が保存されます。保存 / 復帰ペアは、任意の深さにネストすることができます。デフォルト：false
space	（float またはパーセント値）テキスト位置を、与えた値だけ進めます[1]。

1. ユーザー座標で、または文字サイズに対するパーセント値で指定します

表 5.9 PDF_add/create_textflow( ) と、PDF_create_textflow( ) のインラインオプションと、テキストフローブロックに対する PDF_fill_textblock( ) の、追加のテキスト意味付けオプション

オプション	説明
charclass	（ペアのリスト。それぞれのペアの 1 番目の要素はキーワードで、2 番目の要素は Unichar または Unichar のリスト。この Unichar は＜ 0xFFFF である必要があります。advancedlinebreak=true のときは無視されます）指定する Unichar 群を、その改行動作を決定するために指定するキーワードに分類します。 ***letter*** a B 等の文字のように動作します ***punct*** + / ; : 等の句読点のように動作します ***open*** [等の開きカッコのように動作します ***close***] 等の閉じカッコのように動作します ***default*** すべてのキャラクター分類を、PDFlib の内蔵のデフォルトにリセットします 例：charclass={ close » open « letter {/ : =} punct & }
charmapping	（ペアのリスト。それぞれのペアは 2 個の Unichar か、または 1 番目の要素が Unichar で、2 番目の要素が Unichar と整数のリスト。この Unichar は＜ U+FFFF である必要があります）個々のキャラクターを、別の 1 個ないし複数のキャラクターへ置き換えます。オプションリストには 1 個ないし複数の Unichar のペアを入れます。それぞれのペアの 1 番目のキャラクターが、2 番目のキャラクターに置き換わります。一対一対応でなく、それぞれのペアの 2 番目の要素をオプションリストにして、Unichar と個数を入れることもできます。 ***個数＞o*** 置き換えキャラクターをその個数並べます。 ***個数＜o*** キャラクターが複数個並んでいるとき、指定値の固定個数に減らします。 ***個数＝o*** キャラクターを削除します。 例： charmapping={ hortab space CRLF space LF space CR space } charmapping={ shy {shy 0} } charmapping={ hortab {space 4} }

表 5.9 PDF_add/create_textflow( ) と、PDF_create_textflow( ) のインラインオプションと、テキストフローブロックに対する PDF_fill_textblock( ) の、追加のテキスト意味付けオプション

オプション	説明
hyphenchar	（Unichar < 0xFFFF またはキーワード）改行位置でソフトハイフンを置き換えたいキャラクター。値 0 かキーワード none にすると、ハイフンが完全になくなります。このハイフンキャラクターはつねにメインフォントから採られ、フォールバックフォントは考慮されません。デフォルト：U+00AD（ソフトハイフン）、ただしこれがフォントにないときは U+002D（ハイフン・マイナス）
tabalignchar	（Unichar < 0xFFFF）小数点タブを整列させたいキャラクター。デフォルト：U+002E「.」

表 5.10 PDF_add/create_textflow( ) と、PDF_create_textflow( ) のインラインオプションと、テキストフローブロックに対する PDF_fill_textblock( ) の、追加のタグ付けオプション

オプション	説明
tagbegin[1]	（オプションリスト。タグ付き PDF でのみ可）構造エレメントを開きます。表 14.2 に従った短縮構造エレメントタグ付けのためのオプションをすべて指定できます。指定するタグは、適切にネストされる必要があり、かつ、構造エレメントに対するネスト化規則に従う必要があります。
tagend[1]	（論理値。タグ付き PDF でのみ可）テキストフロー内で一番最近に開かれた構造エレメントを閉じます。

1. 短縮タグ付けのための tag オプションが PDF_fit_textflow( ) に与えられている場合にはこのオプションは無視されます。

テキストフローオプションのマクロ　テキストフローのオプションリストには（*PDF_create_textflow( )*・*PDF_add_textflow( )* の *optlist* 引数でも、あるいは *PDF_create_textflow( )* に与えるテキストにインラインでも）、表 5.11 に従って、マクロの定義と、マクロの呼び出しを入れることができます。マクロは、フォント名やインデント量など、何度も使うオプション値を 1 回の定義にまとめたいときに便利でしょう。オプションリストが解析される前にはまず、その中に入っているマクロが、各マクロの定義で与えられているオプションリストの内容に置き換えられます。その結果できたオプションリストが、その後に解析されます。以下に、2 個のマクロのマクロ定義の例を示します。

```
<macro {
        comment { 以下のマクロは段落スタイルとして利用されます }
        H1 {fontname=Helvetica-Bold fontsize=14 }
        body {fontname=Helvetica fontsize=12 }
}>
```

これらのマクロは、オプションリストの中で以下のように利用できるでしょう。

```
<&H1>Chapter 1
<&body>This chapter talks about...
```

マクロの定義と使用には、以下の規則があります。

- マクロは、任意の深さにネストさせることができます（マクロの定義の中で、別のマクロを呼び出すことが可能）。
- マクロは、それを定義しているのと同じオプションリストの中で使うことはできません。*PDF_create_textflow( )* の場合は、マクロを定義しているインラインオプションリストを終わらせた直後に、そのマクロを使う新しいインラインオプションリストを始めればよいでしょう。*PDF_add_textflow( )* を使っている場合は、メソッドを一度呼び出してマクロを定義した後、それを使うにはもう一度呼び出す必要があります（*PDF_add_textflow( )* は一度に 1 つのオプションリストしか受け付けないので）。

- マクロの名前は大文字・小文字を区別します。
- 未定義のマクロは例外を発生させます。
- マクロはいつでも再定義することができます。

表 5.11 PDF_add/create_textflow( )・PDF_fit_textflow( ) と、テキストフローブロックに対する PDF_fill_textblock( ) のオプションリストマクロの定義と呼び出し

オプション	説明
macro	（ペアのリスト）それぞれのペアは、マクロの名前と定義を以下のように記述します（なお、このマクロ名とその定義の間には等号「=」があってはいけません）： ***name*** （文字列）マクロの名前。以後これを使ってマクロを呼び出すことができます。すでに定義してあるマクロを後から定義しなおすこともできます。特殊名 comment は無視されます。 ***suboptlist*** マクロが呼び出された時にマクロ名をリテラルに置き換えるオプションリスト。最初と最後のホワイトスペースは無視されます。
&name	指定名のマクロを展開し、マクロ名（キャラクター & を含め）を、そのマクロの内容、すなわち、そのマクロに対して定義しておいた suboptlist（両端の括弧は含まず）で置き換えます。マクロ名は、ホワイトスペース・{・}・=・& のいずれかで終了します。ですから、これらのキャラクターをマクロ名の中で使ってはいけません。 ネストされたマクロは、ネスト制限なしに展開されます。文字列オプションの中に入れたマクロも展開されます。マクロを置き換えたら有効なオプションリストになるようにする必要があります。

C++ Java C# ***int create_textflow(String text, String optlist)***

Perl PHP ***int create_textflow(string text, string optlist)***

C ***int PDF_create_textflow(PDF *p, const char *text, int len, const char *optlist)***

テキスト内容・インラインオプション・明示オプションからテキストフローオブジェクトを作成します。

text （内容文字列）テキストフローの内容。テキストと、マクロと（112 ページ「テキストフローオプションのマクロ」参照）、表 5.6・表 5.12 に従ったインラインオプションリスト（115 ページ「テキストフローのインラインオプションリスト」も参照）を入れることができます。*text* を空文字列にしても、有効なテキストフローハンドルが返されます。

len （C 言語バインディングのみ）テキストの長さをバイト単位で表すか、または null 終了文字列では 0。

optlist テキストフローオプションを指定したオプションリスト。*optlist* で指定するオプションは、*text* の中のインラインオプションリストで指定するオプションよりも前に評価されるので、*optlist* 引数で与えるオプションよりもインラインオプションのほうが優先されます。以下のオプションが使えます：

- *PDF_add_textflow( )* のすべてのオプション
- インラインオプションリストの処理を表 5.12 に従って制御するオプション：
  begoptlistchar・*endoptlistchar*・*fixedtextformat*・*textlen*

戻り値 テキストフローハンドル。*PDF_add_textflow( )*・*PDF_fit_textflow( )*・*PDF_info_textflow( )*・*PDF_delete_textflow( )* への呼び出しで使えます。ハンドルは、カレントの**文書**スコープを終えるか、または *PDF_delete_textflow( )* でこのハンドルを指定して呼び出すまで有効で

す。デフォルトでは、このメソッドはエラーが起きたときは -1（PHP では 0）を返します。この動作は、*errorpolicy* オプションで変えることもできます。

詳細 このメソッドは、オプションとテキストを受け付けて、テキストフローを作ります。*PDF_add_textflow()* 関数とは異なり、テキストにインラインオプションを入れることもできます。インラインオプションリストの検索は、*optlist* 引数で *textlen* オプションを与えれば、テキストの一部分または全体で無効にすることもできます（115 ページ「テキストフローのインラインオプションリスト」参照）。

このメソッドは、生成 PDF 文書に出力を作成せず、ただ与えられたオプションに従ってテキストを用意します。出力を作成するには、*PDF_fit_textflow()* を使って、この前処理したテキストフローのハンドルを指定します。

特殊キャラクターや改行などの情報については、詳しくは *PDF_add_textflow()* の**詳細**の項を参照してください。

スコープ オブジェクト以外の任意

表 5.12 PDF_create_textflow( ) と、オプション inlineoptions を指定した PDF_add_textflow( ) と、テキストフローブロックに対する PDF_fill_textblock( ) のインラインオプションリスト処理のための追加オプション

オプション	説明
begoptlistchar	（Unichar < 0xFFFF またはキーワード）オプションリストを開始させたいキャラクター。デフォルトのキャラクターがテキストにリテラルに現れるときは、これを別のキャラクターに替えると便利かもしれません（115 ページ「テキストフローのインラインオプションリスト」参照）。textlen を指定していないときは、テキストの中の begoptlistchar キャラクターは、先行するテキストと同じテキスト形式とエンコーディングでエンコードしておく必要があります。ということは、先行するテキストのエンコーディングに begoptlistchar が含まれているように、その Unicode 値を選ぶ必要があります。キーワード none を使うと、オプションリストの検索を完全に無効にすることができます。デフォルト：U+003C（<）
endoptlistchar	（Unichar < 0xFFFF。U+007D「}」は使えません）インラインオプションリストを終了させたいキャラクター。デフォルト：U+003F（>）
fixedtext-format	（論理値。C の場合と、Perl・PHP・Ruby で stringformat=legacy の場合にのみ可。このオプションは、インラインオプションリストでは意味を持たず、optlist 引数でのみ使えます）true にすると、部分テキストもインラインオプションリストも、すべて同じ textformat を使います。これは、utf8・utf16・utf16be・utf16le のうちのいずれかにする必要があります。これは、テキストとインラインオプションの取得元が同じときに有用です。 false にすると、テキストの形式にかかわらず、インラインオプションリストは、区切りキャラクターを含め、textformat=bytes でエンコードされていなければなりません。これを使うと、たとえば UTF-16 のテキストと、ASCII エンコーディングのインラインオプションリストを組み合わせること（テキストは Unicode のデータベースから取って、インラインオプションはアプリケーションの中で ASCII テキストとして作る場合など）が可能です。デフォルト：false
textlen	（整数またはキーワード。encoding=glyphid を用いた部分テキストの場合と、Unicode 非対応言語で fixedtextformat=false かつ textformat=utf16xx を用いた部分テキストの場合には必須。PDF_add_textflow( ) で inlineoptions=true の場合には許されません）次のインラインオプションリストの前の（115 ページ「テキストフローのインラインオプションリスト」参照）符号単位の数（たとえば、stringformat=utf8 を用いている言語バインディングならバイト数。Java か .NET なら UTF-16 単位数。UTF-32 モードの C++ でも UTF-16 符号単位数と見なされます）。符号単位は、文字参照が解決される前に数えられます。例：<textlen=8>①<...>。キーワード all を指定すると、残りのテキストすべてを意味します。デフォルト：次に begoptlistchar が現れるまでテキストが検索されます。

テキストフローのインラインオプションリスト　*PDF_create_textflow()* の *text* 引数で、または *PDF_add_textflow()* で *inlineoptions=true* を指定して与える内容の中には、テキストフローオプションを表 5.6 に従って指定した、任意の数のオプションリスト（インラインオプション）を入れることができます。あるいは、これらのオプションはすべて、*PDF_create_textflow()*・*PDF_add_textflow()* の *optlist* 引数の中で与えることもできます。1 つのオプションリストの中で同じオプションを複数回指定することも可能です。その場合、最後に指定したそのオプションだけが考慮されます。

インラインオプションリストは、*begoptlistchar* と *endoptlistchar* オプションで指定するキャラクター（デフォルトでは < と >）で囲う必要があります。当然、インラインオプションの開始に使っているキャラクターを、テキスト本体でも使わなければならないときは、衝突が起こります。この衝突を解決するにはいくつかの方法があり、テキストにインラインオプションリストを入れるかどうかによって方法が決まります。*PDF_add_textflow()* の場合は先述のとおり、テキストとオプションを完全に分離していますので、衝突は起こりません。

テキストにインラインオプションを全く入れないときは、以下のいずれかの方法で、インラインオプションリストの検索を完全に無効にすることもできます。

- *PDF_create_textflow()* の *optlist* 引数で *begoptlistchar=none* を設定。
- *PDF_create_textflow()* の *optlist* 引数で *textlen* オプションにテキスト全体の長さを設定。

テキストにインラインオプションを入れるときは、以下のいずれかの方法を使えば、テキストの内容と、インラインオプションを始める *begoptlistchar* との衝突を避けることができます。

- テキスト内に現れる < キャラクターをすべて、その数値または文字実体参照（*<* か *<*）で置き換えて、インラインオプションリストをリテラルな < キャラクターで開始。

  ```
  A&lt;B<fontname=Helvetica>
  ```

  ただしこの方式は、*encoding=builtin* によるフォントでは動作しません。
- *PDF_create_textflow()* の *optlist* 引数かインラインオプションリストで *begoptlistchar* オプションに、テキストで使っていないキャラクターを設定し（*$* 等）、そのキャラクターを使ってインラインオプションリストを開始。

  ```
  <begoptlistchar=$>A<B$fontname=Helvetica>
  ```

- 先行するインラインオプションリストで *textlen* オプションを使って、次の部分テキスト（次のインラインオプションリストの開始までの）の長さを指定。

  ```
  <textlen=3>A<B<fontname=Helvetica>
  ```

- *begoptlistchar* をエスケープシーケンスとして指定し、*escapesequence* グローバルオプションを *true* に設定。ただし、エスケープシーケンスは、*endoptlistchar* を含んだインラインオプションリスト内では動作しません。

注　複数のインラインオプションリストを間に何のテキストもなく与えることは避けるべきです。すべてのオプションをただ 1 つのオプションリストの中へまとめることを推奨します。インラインオプションリストの直後にまたオプションリストを与えると、間に長さゼロの部分テキストをはさんでいると見なされます。1 番目のオプションリストで textlen オプションを与えるときはこれは重要です。

C++ Java C# **String fit_textflow(int textflow, double llx, double lly, double urx, double ury, String optlist)**

Perl PHP **string fit_textflow(int textflow, float llx, float lly, float urx, float ury, string optlist)**

C **const char *PDF_fit_textflow(PDF *p,**
int textflow, double llx, double lly, double urx, double ury, const char *optlist)

テキストフローの次の部分を組版します。

textflow PDF_create_textflow() か PDF_add_textflow() を呼び出して返されたテキストフローハンドル。

llx・lly・urx・ury 対象にしたい長方形（**はめ込み枠**）の左下隅と右上隅の *x・y* 座標を、ユーザー座標で指定します。この 2 隅は逆の順に指定することもできます。長方形でない輪郭へ流し込みを行うには、*wrap* オプションを使います。

optlist 処理オプションを指定したオプションリスト。以下のオプションが使えます：

- 表 5.13 に従ったテキストフローオプション：
  avoidwordsplitting・blind・createfittext・createlastindent・exchangefillcolors・exchangestrokecolors・firstlinedist・fitmethod・fontscale・lastlinedist・linespreadlimit・maxlines・minfontsize・orientate・returnatmark・rewind・rotate・showborder・showtabs・stamp・truncatetrailingwhitespace・verticalalign・wrap
- 表 6.1 に従った範囲枠オプション：*matchbox*
- 表 14.4 に従った、短縮構造エレメントタグ付けのためのオプション（**ページ**スコープでのみ可）：*tag*

戻り値 メソッドから戻った原因を示す文字列。

- *_stop*：テキストフローの全テキストの処理が完了。テキストが空のときは、*return* または *mark/returnatmark* オプションを与えていても、つねに *_stop* が返されます。
- *_nextpage*：次のページを待ち（フォームフィードキャラクター U+000C が原因）。また PDF_fit_textflow() を呼び出して残りのテキストを処理する必要があります。
- *_boxfull*：はめ込み枠内にテキストをいくらか配置してもうゆとりがないか、または最大行数（*maxlines* オプションで指定している）をはめ込み枠に配置してしまったか、または *fitmethod=auto* と *minfontsize* を指定しているがテキストがはめ込み枠に収まらなかった。また PDF_fit_textflow() を呼び出して残りのテキストを処理する必要があります。
- *_boxempty*：処理後、枠にテキストがまったく入っていない。これは、はめ込み枠の大きさが小さすぎてテキストが入らないときに、または回り込み枠がはめ込み枠よりも大きいときに起きることがあります。無限ループを避けるため、同じはめ込み枠を指定してまた PDF_fit_textflow() を呼び出すべきではありません。
- *_mark#*：*returnatmark* オプションが番号 # で指定されており、このオプションで指定された番号のマークが配置された。
- その他の任意文字列：インラインオプションリストで *return* コマンドに与えた文字列。

戻った理由が同時に複数あるときは、上記の一覧で（上から下へ）最初のものが報告されます。返された文字列は、次にこのメソッドを呼び出すまで有効です。

詳細 カレントテキスト・グラフィックステートは、このメソッドが作成するテキスト出力では効力を持ちません（この点は PDF_fit_textline() と異なります）。PDF_create_textflow()・PDF_add_textflow() でテキストの書式を制御するには、*fillcolor・strokecolor* などのテキスト書式オプションを使います（表 5.2 参照）。このメソッドから返った後で、テキストス

テータスは変更されていません。ただし、*textx/texty* オプションは、生成したテキスト出力の末尾の点へ移動します（*blind* オプションを *true* に設定していなければ）。

スコープ　ページ・パターン・テンプレート・グリフ

表 5.13 PDF_fit_textflow( ) と、テキストフローブロックに対する PDF_fill_textblock( ) のオプション

オプション	説明
avoidword-splitting	（論理値）true かつ fitmethod=auto の場合には、テキストフローは、文字サイズを下げて単語分割を避けることによって、テキストをはめ込み枠内へ完全に収めようとします（adjustmethod 参照）。
blind	（論理値）true にすると、出力が生成されず、しかし計算はすべて行われ、組版結果を PDF_info_textflow( ) で調べることができます。デフォルト：false
createfittext	（論路値）true にすると、カレントはめ込み枠に配置されたテキストがメモリーに保存されて、以後、キーワード fittext を付けた PDF_info_textflow( ) への呼び出しによって取得できるようになります。デフォルト：true
createlast-indent	（オプションリスト）はめ込み枠の末尾行の末尾にいくらかのアキをとります。また、範囲枠を生成してそのアキに入れることもできます。このアキは、テキストの末尾に、続きがあることを示す点群や画像や、続きのテキストへのリンクなどを追加するために有用でしょう： ***rightindent***（float またはパーセント値）はめ込み枠の末尾行の追加右インデントを、ユーザー座標で、またははめ込み枠の幅に対するパーセント値で指定します。この値は、PDF_add/create_textflow( ) の rightindent オプションの値に追加されます。デフォルト：0 ***matchbox***（表 6.4 に従ったオプションリスト）末尾行の末尾に範囲枠を生成します。範囲枠オプション boxwidth を指定しないときは、rightindent の値が枠の幅として用いられます。boxwidth=0 にすると枠は生成されません。
exchange-fillcolors	（色偶数個のリスト）リスト内のそれぞれのペアで、元の塗り色と、置き換え色を指定します。はめ込み枠内でテキストフローが元の塗り色を指定している部分がすべて、指定した置き換え色で置き換えられます。これは、色を背景に応じて調節するために有用でしょう。キャラクター「*」を元の色として指定すると、残りのすべての色が置き換え色で置き換えられます。例： exchangefillcolors={{gray 0} white Orchid DeepPink {rgb 1 0 1} MediumBlue}
exchange-strokecolors	（色偶数個のリスト）リスト内のそれぞれのペアで、元の描線色と、置き換え色を指定します。はめ込み枠内でテキストフローが元の描線色を指定している部分がすべて、指定した置き換え色で置き換えられます。これは、色を背景に応じて調節するために有用でしょう。キャラクター「*」を元の色として指定すると、残りのすべての色が置き換え色で置き換えられます。
firstlinedist[1]	（float・パーセント値・キーワードのいずれか）はめ込み枠上端とテキスト先頭行ベースラインの間隔を、ユーザー座標で、または関連文字サイズ（fixedleading=true にしているときは行の先頭の文字サイズ、そうでなければ行内のすべての文字サイズのうちの最大値）に対するパーセント値で、あるいはキーワードで指定します（デフォルト：leading）： ***leading*** 先頭行で決定された行送り値。Ł等の、読み分け記号つきの代表的なキャラクターがはめ込み枠上端に接します。 ***ascender*** 先頭行で決定されたアセンダー値。d や h 等の、大きなアセンダーを持つ代表的なキャラクターがはめ込み枠上端に接します。 ***capheight*** 先頭行で決定されたキャップハイト値。H 等の、代表的な大文字がはめ込み枠上端に接します。 ***xheight*** 先頭行で決定された x ハイト値。x 等の、代表的な小文字がはめ込み枠上端に接します。 fixedleading=false にしているときは、先頭行の中で見出されたすべての leading・ascender・xheight・capheight 値のうちの最大値が使われます。

表 5.13 PDF_fit_textflow( ) と、テキストフローブロックに対する PDF_fill_textblock( ) のオプション

オプション	説明
fitmethod	（キーワード）テキストをはめ込み枠内にはめ込むのに使いたい方式： ***auto*** テキストがはめ込み枠に収まるまで、文字サイズを下げたり、その他のフォント関連のオプション群を変えたりしながら（fontscale 参照）、PDF_fit_textflow( ) をブラインドモードで繰り返し呼び出します（ただし minfontsize オプションも参照）。 ***clip*** テキストをはめ込み枠の下端で切り落とします。 ***nofit*** テキストを、はめ込み枠の下端から（verticalalign=top の場合）、またははめ込み枠の上端から（verticalalign=bottom の場合）、または両方から（verticalalign=center の場合）、はみ出させることもあります。 PDF_fit_textflow( ) の場合、デフォルトは clip です。テキストフローブロックに対する PDF_fill_textblock( ) の場合、デフォルトは、オプションを与えていれば textflowhandle であり、そうでなければ auto です。
fontscale	（正の float またはパーセント値）fontsize の値と、leading・minspacing・maxspacing・spreadlimit・space の絶対値（パーセント値は無視）に、与える倍率またはパーセント値を掛け算します。デフォルト：rewind=0 にしているときは 1、それ以外にしているときはその PDF_fit_textflow( ) への呼び出しで与えていた値。
gstate	（PDF_create_gstate( ) に対するオプションのオプションリスト、またはグラフィックステートハンドル）グラフィックステートオプション群またはハンドル。このグラフィックステートが、このメソッドで生成されるすべてのテキストに対して効力を持ちます。別のグラフィックステートを PDF_add/create_textflow( ) に与えていたときは、グラフィックステートがマージされます。デフォルト：グラフィックステートなし、すなわち、カレント設定が用いられます
lastlinedist[1]	（float・パーセント値・キーワードのいずれか。fitmethod=nofit の場合には無視されます）テキスト最終行ベースラインとはめ込み枠下端の間隔を、ユーザー座標で、または文字サイズ（fixedleading=true の場合には行の先頭の文字サイズ、そうでない場合には行内のすべての文サイズのうちの最大値）に対するパーセント値で、あるいはキーワードで指定します。デフォルト：0、すなわちはめ込み枠下端をベースラインとして使い、代表的なディセンダーがはめ込み枠の下に出ます。使えるキーワード： ***descender*** 最終行で決定されたディセンダー値。g や j 等の、ディセンダーを持つ代表的なキャラクターがはめ込み枠下端に接します。fixedleading=false にしているときは、最終行の中で見出されたすべての descender 値のうちの最大値が使われます。
linespread-limit	（float またはパーセント値。verticalalign=justify の場合にのみ可）縦揃えで行送りを増やせる最大値を、ユーザー座標で、または行送りに対するパーセント値で指定します。デフォルト：200%
maxlines	（整数またはキーワード）はめ込み枠内の最大行数か、または、できるだけ多くの行をはめ込み枠内に配置させたいときはキーワード auto を指定します。最大行数を配置したときは、PDF_fit_textflow( ) は文字列 _boxfull を返します。デフォルト：auto
minfontsize	（float またはパーセント値）特に fitmethod=auto にしているときに、テキストを縮小してはめ込み枠に収めるときの、許容できる最小文字サイズ。この制限値は、ユーザー座標系か、またははめ込み枠の高さに対するパーセント値で指定します。制限を超えてもテキストが枠に収まりきらないときは、文字列 _boxfull が返されます。デフォルト：0.1%
mingapwidth	（float またはパーセント値）複数の輪郭の間に（例：複数の回り込み輪郭の間に）テキストをはめ込むための最小横幅を、ユーザー座標で、または文字サイズに対するパーセント値で指定します。これは、複数の回り込み輪郭の間に狭い隙間しかない場合に組版結果が醜くならないようにするために有用でしょう。デフォルト：10%

表 5.13 PDF_fit_textflow( ) と、テキストフローブロックに対する PDF_fill_textblock( ) のオプション

オプション	説明
orientate	（キーワード）テキストを配置する時に向けたい向き（デフォルト：north）： ***north*** 直立 ***east*** 右倒し ***south*** 上下逆さま ***west*** 左倒し
returnatmark	（整数）指定する番号で定義された mark オプションのあるテキスト位置で、PDF_fit_textflow( ) は不完全なまま返ります。戻り理由文字列は _mark# となります。ここで # はこのオプションで指定した番号です。
rewind	（−2・−1・0・1 のいずれかの整数。テキストフローがインラインタグを含んでいる場合には不可）与えるテキストフローの状態が、同じテキストフローハンドルを指定して PDF_fit_textflow( ) を呼び出したいずれかの時の前の状態へリセットされたのちに、そのテキストフローが通常どおり配置されます（デフォルト：0）： ***1*** PDF_fit_textflow( ) を最初に呼び出した時の前の状態へ巻き戻し。 ***0*** テキストフローをリセットしません。 ***-1*** PDF_fit_textflow( ) を直前に呼び出した時の前の状態へ巻き戻し。 ***-2*** PDF_fit_textflow( ) を最後から 2 番目に呼び出した時の前の状態へ巻き戻し。
rotate	（float）はめ込み枠の左下隅を中心に、指定する値を度単位の回転角として、座標系を回転させます。結果として、はめ込み枠とテキストが回転します。テキストの配置が完了した時点で回転はリセットされます。デフォルト：0
showborder	（論理値）true にすると、はめ込み枠の辺が描線されます（カレントグラフィックステートを使って）。これは開発やデバッグに便利でしょう。デフォルト：false
showtabs	（キーワード）デバッグの補助のために、タブ位置と左インデントを縦線で視覚表示します。線は、PDF_fit_textflow( ) を呼び出す前に有効だったグラフィックステートに従って描かれます（デフォルト：none）： ***none*** 線を描きません ***fitbox*** 線をはめ込み枠の高さいっぱいに描きます ***validarea*** 線をそれが有効な縦領域にだけ描きます
stamp	（キーワード）このオプションを使うと、範囲枠内に対角線上のスタンプを生成することができます。スタンプテキストの改行は明示的に指定する必要があります（すなわち、ニューラインキャラクターかニューラインオプションを用いて）。テキストの中に明示的な改行が全くないときは、1 行のスタンプが生成されます。生成されるスタンプテキストは可能な限り大きくされますが、しかし指定した文字サイズよりは大きくなりません。使えるキーワード（デフォルト：none）： ***ll2ur*** スタンプが左下隅から右上隅への対角線上に配置されます。 ***ul2lr*** スタンプが左上隅から右下隅への対角線上に配置されます。 ***none*** スタンプは生成されません。
truncate-trailing-whitespace	（論理値）末尾空白のみを内容とするはめ込み枠、すなわち、はめ込み枠が空白で始まり、かつテキストフローの末尾まで空白しかない場合の処理を制御します。このオプションが true の場合、末尾空白は除去され、すなわち、そのはめ込み枠は空として扱われ、その戻り値は _stop となります。このオプションを false にすると、この空白は通常のテキストと同様に処理され、すなわち、このメソッドは _stop 以外の値（末尾空白の量に依存）を返し、PDF_info_textflow( ) の textendx/y やその他のキーワードはこの空白を考慮に入れます。truncatetrailingwhitespace=false は、オリジナルのテキストを空白除去なしに処理する必要がある場合に有用でしょう。デフォルト：true

表 5.13 PDF_fit_textflow( ) と、テキストフローブロックに対する PDF_fill_textblock( ) のオプション

オプション	説明
verticalalign[1]	（キーワード）はめ込み枠内のテキストの縦揃え。firstlinedist・lastlinedist オプションを適切に考慮されます（デフォルト：top）： ***top*** 組版を先頭行から始めて、下へ進行。テキストがはめ込み枠を満たさないときは、テキストの下に空白ができます。 ***center*** テキストをはめ込み枠の縦中央に置きます。テキストがはめ込み枠を満たさないときは、テキストの上と下の両方に空白ができます。 ***bottom*** 組版を最終行から始め、上へ進行。テキストがはめ込み枠を満たさないときは、テキストの上に空白ができます。 ***justify*** テキストをはめ込み枠の上端と下端に整列させます。そのために行送りを、linespreadlimit で指定している制限内で増やします。先頭行の高さは、firstlinedist=leading にしているときだけ増やします。
wrap	（表 5.14 に従ったオプションリスト）テキストが、表 5.14 に示すサブオプション群で指定する輪郭を回り込みます。これを使うと、テキストフローの中にグラフィックを配置して、テキストをその回りに回り込ませたり、あるいは任意の輪郭の中へテキストを流し込んだりすることができます。はめ込み枠への流し込みは、fillrule オプションに従って行われ、はめ込み枠の辺から開始されます。 デフォルトでは、指定する領域に一切テキストが入りません（領域どうしが重なり合う場合を除き）。すなわち、テキストは輪郭を回り込みます。addfitbox・inversefill オプションを使うと、これと逆の効果が得られます：すなわち、指定する領域へテキストが流し込まれ、その外側の、はめ込み枠と輪郭の間の領域は空のままになります。これを利用すれば、任意の輪郭（llx/lly/urx/ury 引数で与える長方形にかぎらず）へテキストを流し込むことができます。 絶対座標値と相対座標値はユーザー座標系で解釈されます。相対座標は、直前の絶対座標に追加されます。最大 256 個の値を相対値として与えることができます。パーセント値は、はめ込み枠座標系で、すなわちはめ込み枠の左下隅を (0, 0)、右上隅を (100, 100) として解釈されます（上から下への座標系においても）。最大 256 個の値をパーセント値として与えることができます。例： 枠を相対座標で除外：wrap={ boxes={{120r 340r 50r 60r}} } （wrap={ boxes={{120 340 170 400}} } と同等） はめ込み枠の右上 4 分の 1 部分を除外：wrap={ boxes={50% 50% 100% 100%} } 三角形の輪郭へ流し込み：wrap={ addfitbox polygons={{50% 80% 30% 40% 70% 40% 50% 80%}} } image1 という範囲枠をつけた画像の領域を回り込み：wrap={ usematchboxes={{ image1 }}}

1. firstlinedist・lastlinedist・verticalalign オプションは、たとえ回り込み要素が存在していても、つねにはめ込み枠からの指定になります。すなわちテキストフローは、テキストとはめ込み枠の辺との距離を、および verticalalign オプションに従ってテキスト枠の位置を決定するにあたり、回り込み要素群の境界枠を用いません。このことはとりわけ、反転流し込み、すなわち回り込み要素群にテキストを流し込む場合に重要になります。このせいで、とくに回り込み要素の外辺がはめ込み枠に接していない場合には、結果が思い通りにならないかもしれません。この影響は、はめ込み枠に接するような回り込み要素を与えることでほぼ完全に避けることができます。

表 5.14 PDF_fit_textflow( ) と、テキストフローブロックに対する PDF_fill_textblock( ) の wrap オプションのサブオプション

オプション	説明
addfitbox	（論理値）はめ込み枠が回り込み領域に追加されます。結果として、他の回り込みオプション群で指定する輪郭について、テキストがその輪郭を回り込むのではなく、輪郭へテキストが流し込まれます。デフォルト：false
beziers	（ベジエ曲線 2 個以上のリスト）回り込み領域に追加したいベジエ曲線群。
boxes	（長方形のリスト）回り込み領域に追加したい 1 個ないし複数の長方形。
circles	（円のリスト）回り込み領域に追加したい 1 個ないし複数の円。

表 5.14 PDF_fit_textflow( ) と、テキストフローブロックに対する PDF_fill_textblock( ) の wrap オプションのサブオプション

オプション	説明
creatematch-boxes	（オプションリストのリスト）boxes オプション内の長方形 1 個ないし複数から範囲枠を生成します。それぞれのオプションリストが、boxes オプション内の項目 1 個に対応し（順序は意味を持ちます）、範囲枠 1 個の生成を制御します。表 6.4 のすべての関連する範囲枠オプションが使えます。サブオプションリストは空にすることもでき、その場合は、対応する回りこみ枠に対する範囲枠は生成されません。
fillrule	（キーワード）重なり合う回り込み輪郭の内側を決定するための方式（デフォルト：evenodd）。詳しくは表 7.1 を参照： ***evenodd*** 偶奇規則を用います。 ***winding*** 非ゼロ巻数規則を用います。重なり合う円の内側を処理したいとき（すなわち、「ドーナツの穴」を避ける）や、重なり合う輪郭の和（共通部でなく）を処理したいときは、この規則を用います。
inversefill	（論理値）true にすると、回り込み輪郭の処理は、テキスト行とはめ込み枠内の回り込み要素の辺との最初の共通部から始まります。false にすると、処理ははめ込み枠の辺から始まります。fillrule=evenodd の場合、inversefill=true オプションは addfitbox=true と同じ効果を持ちます。fillrule=winding の場合、addfitbox=true オプションならばはめ込み枠は空か満たされるかのどちらかになります（それぞれ inversefill=false・true のとき）。
lineheight	（要素 2 個のリスト。各要素は正の float かキーワード）回り込み要素との交わりを算出するために使われる、テキスト行の縦範囲を定義します。テキストのベースラインの上方と下方の範囲について、キーワード 2 個か float 2 個を指定することができます。使えるキーワード： none（範囲なし）・xheight・descender・capheight・ascender・fontsize・leading・textrise デフォルト：{ascender descender}
usematch-boxes	（文字列のリストのリスト）それぞれのリストの 1 番目の要素は、範囲枠を指定する名前文字列です。2 番目の要素は、示したい長方形の番号を指定する整数か、または選んでいる範囲枠を参照しているすべての長方形を指定したいときはキーワード all も使えます。2 番目の要素を指定しないと、デフォルトとして all と見なされます。それぞれの長方形の外接枠が、テキストの回り込みのための輪郭として使われます。
offset	（float またはパーセント値）テキストと回り込み領域の輪郭との横間隔を、ユーザー座標で、またははめ込み枠の幅に対するパーセント値で指定します。これを使うと、回り込み領域を横へ拡げることができます。デフォルト：0
paths	（オプションリストのリスト）回り込み領域に追加したい 1 個ないし複数のパス： ***path*** （パスハンドル。必須）回り込み領域に追加したいパスのハンドル。 ***refpoint*** （float 2 個かパーセント値 2 個のリスト）パスに対する参照点の座標を、ユーザー座標で、またははめ込み枠の幅と高さに対するパーセント値で指定します。デフォルト：{0 0} PDF_draw_path( ) の以下のオプション（表 6.1・表 7.7 参照）も使えます： align・attachmentpoint・boxsize・close・fitmethod・orientate・position・round・scale・subpaths
polygons	（折れ線のリスト）回り込み領域に追加したい 1 個ないし複数の折れ線。閉じていなくてもかまいません）。

C++ Java C# **double info_textflow(int textflow, String keyword)**

Perl PHP **float info_textflow(int textflow, string keyword)**

C **double PDF_info_textflow(PDF *p, int textflow, const char *keyword)**

テキストフローの、*PDF_fit_textflow()* を呼び出した後のカレントステータスを取得します。

textflow *PDF_add/create_textflow()* か *PDF_fill_textblock()* で *textflowhandle* オプションを指定して呼び出して返されたテキストフローハンドル。

keyword ほしい情報を表 5.15 に従って指定したキーワード。

戻り値 *keyword* で要求した何らかのテキストフロー特性の値。このメソッドは、ブラインドモードでも正しい位置情報を返します（*textx/texty* オプションとは異なり）。要求されたキーワードがテキストを生み出す場合には、文字列番号が返され、その対応する文字列を、*PDF_get_string()* を用いて取得する必要があります。

スコープ **オブジェクト**以外の任意

表 5.15 PDF_info_textflow( ) のキーワード

キーワード	説明
boundingbox	テキストフローの外接枠をユーザー座標で表したものを内容として持つパスのハンドル、または -1（PHP では 0）。firstlinedist と lastlinedist が考慮されます。
boxlinecount	最後のはめ込み枠の中の行数
firstparalinecount	はめ込み枠の最初の段落の行数
firstlinedist	テキストの先頭ベースラインと、上方の空想のベースライン（verticalalign=top ならこれがはめ込み枠の上端になる）の間隔
fittext	PDF_fit_textflow( ) への直前の呼び出しで配置されたテキストに対する文字列番号。これを利用すると、はめ込み枠内に配置することができたテキストの量を知ることができます。文字列は以下のように正規化されます：エンコーディングは、Unicode 対応言語では UTF-16 になり、それ以外では（EBCDIC-）UTF-8 になり、改行は U+000A でマークされ、水平タブはスペースキャラクター U+0020 で置き換えられます。
fontscale	PDF_fit_textflow( ) を fitmethod=auto でもっとも最近に呼び出した後の fontscale の値
lastfont	はめ込み枠の末尾テキスト行の中で使われているフォントのハンドル
lastfontsize	はめ込み枠の末尾テキスト行の中で使われている文字サイズ
lastmark	最後のはめ込み枠の中にあるテキストフローの処理済みの部分で見つかった最後のマークの番号（マークは mark オプションで設定することができます）
lastlinedist	テキストの最終ベースラインと、行送りの変更がなかったとしたときの下方の空想のベースライン（verticalalign=bottom ならこれがはめ込み枠の下端になる）の間隔
lastparalinecount	はめ込み枠の最後の段落の行数
leading	テキストフローの中のテキストとオプションによって決定される、leading オプションのカレント値
leftlinex[1] ・ ***leftliney***[1]	もっとも最近に流し込みが行われたはめ込み枠の中の、もっとも左で始まる行の x・y 座標を、カレントユーザー座標系で表したもの
maxlinelength	もっとも最近に流し込みが行われたはめ込み枠の中の、もっとも長いテキスト行の長さ

表 5.15 PDF_info_textflow( ) のキーワード

キーワード	説明
maxliney[1]	もっとも最近に流し込みが行われたはめ込み枠の中の、もっとも長いテキスト行のベースラインの y 座標を、カレントユーザー座標系で表したもの
minlinelength	もっとも最近に流し込みが行われたはめ込み枠の中の、もっとも短いテキスト行の長さ
minliney[1]	もっとも最近に流し込みが行われたはめ込み枠の中の、もっとも短いテキスト行のベースラインの y 座標を、カレントユーザー座標系で表したもの
returnreason	もっとも最近の PDF_fit_textflow( ) への直接または間接の呼び出しの戻り原因に対する文字列番号（表 2.1 参照）。取得される戻り原因は、PDF_fit_textflow( ) が返す文字列のいずれかと同じになります。これは、PDF_fill_textblock( ) が内部的に行う間接的なテキストフロー呼び出しの結果を取得したいときに有用です。
rightlinex[1] ・ ***rightliney***[1]	もっとも最近に流し込みが行われたはめ込み枠の中の、もっとも右で終わる行の x・y 座標を、カレントユーザー座標系で表したもの
split	最後のはめ込み枠の中で単語の分割が起きたかどうかを示します。 ***0*** 単語を分割する必要はなかった。 ***1*** 少なくとも 1 個の単語を分割する必要があった。
textendx・textendy	もっとも最近にはめ込み枠へ流し込みが行われた後の、カレントテキスト位置の x・y 座標を、カレントユーザー座標系で表したもの
textheight	テキスト全体の外接枠の高さを（firstlinedist と lastlinedist を考慮して）、カレントユーザー座標系で表したもの
textwidth	テキスト全体の外接枠の幅を、カレントユーザー座標系で表したもの
used	現時点で配置済みのテキストの割合を、パーセント値で表したもの（0 ～ 100）
x1・y1・...x4・y4	テキスト全体の外接枠の座標をカレントユーザー座標系で表したもの。firstlinedist と lastlinedist が考慮されます。x1, y1 は左下、x2, y2 は右下、x3, y3 は右上、x4, y4 は左上を表します。

1. rotate が 0 以外のときは、この値は回転後の系を参照します。

C++ Java C# ***void delete_textflow(int textflow)***

Perl PHP ***delete_textflow(int textflow)***

C ***void PDF_delete_textflow(PDF *p, int textflow)***

テキストフローと、関連するすべてのデータ構造を削除します。

textflow *PDF_create_textflow( )* か *PDF_add_textflow( )* を呼び出して返されたテキストフローハンドル。

詳細 このメソッドで削除していないテキストフローは、カレントの**文書**スコープを終える時に自動的に削除されます。しかし、多くのテキストフローを生成するときは、*PDF_delete_textflow( )* を呼び出さないと、アプリケーションはかなり遅くなります。

スコープ 任意

5.3 表の組版

C++ **int add_table_cell(int table, int column, int row, string text, string optlist)**
Perl PHP **int add_table_cell(int table, int column, int row, string text, string optlist)**
C **int PDF_add_table_cell(PDF *p,**
int table, int column, int row, const char *text, int len, const char *optlist)

新規または既存の表にセルを追加します。

table 以前に *PDF_add_table_cell()* を呼び出して取得した有効な表ハンドルか、または新規表を開始したいときは -1（PHP では 0）。この表ハンドルは、まだ *PDF_fit_table()* への呼び出しでは使っていないものでなければなりません。すなわち、表の内容はすべて、表をページに配置する前に定義する必要があります。

column・row セルを入れたい列と行の番号。セルが複数の列・行にわたるときは、いちばん左の列といちばん上の行を与える必要があります。最初の列・行の番号は 1 です。

text （内容文字列）セルに入れたいテキスト。*text* を空以外にすると、*PDF_fit_textline()* を使ってセル内に書き込まれます。

len （C 言語バインディングのみ）*text* の長さ（バイト単位）。*len=0* にすると null 終了文字列を与える必要があります。

optlist 表セルの組版の詳細を指定したオプションリスト：

- 一般オプション：*errorpolicy*（表 1.5 参照）
- 表 5.16 に従った表列・表行定義オプション：*colwidth*・*colscalegroup*・*minrowheight*・*return*・*rowheight*・*rowjoingroup*・*rowscalegroup*
- 表 5.16 に従ったセル特性オプション：
  avoidwordsplitting・*colspan*・*margin*・*marginleft*・*marginbottom*・*marginright*・*margintop*・*rowspan*
- 表 5.16 に従ったセル内容組版オプション：*continuetextflow*・*repeatcontent*
- 表 5.16 に従った静的セル内容：
  fitgraphics・*fitimage*・*fitpath*・*fitpdipage*・*fittextflow*・*fittextline*・*graphics*・*image*・*matchbox*・*path*・*pdipage*・*textflow*
- 表 5.18 に従ったインタラクティブセル内容（**ページ**スコープでのみ可）：
  annotationtype・*fieldname*・*fieldtype*・*fitannotation*・*fitfield*
- 表 14.4 に従った、短縮構造エレメントタグ付けのためのオプション：*tag*

戻り値 表ハンドル。以後の表関連の呼び出しで使えます。*errorpolicy=return* の場合、戻り値 -1（PHP では 0）はエラーを表すので、そうでないかを呼び出し側で調べる必要があります。エラーが起きたときは、最後のセル定義だけが破棄されます。表には内容は追加されませんが、表ハンドルは有効なままです。返された表ハンドルは、複数の PDF 出力文書にわたって再利用することはできません。

詳細 表セルには、画像・グラフィック・取り込み PDF ページ・パスオブジェクト・フォームフィールド・注釈・テキストフロー・テキスト行を入れることができます。1 つのセルに対して、複数の内容種別を、1 回のメソッド呼び出しで指定することもできます。

表の組版アルゴリズムと、幅・高さ計算の解説は PDFlib チュートリアルを参照。

PDF/UA *path* か *image* オプションを用いて与えられたベクトルグラフィックとラスタ画像は、*tag* オプションを用いて *Artifact* か *Figure* としてタグ付けする必要があります。

スコープ **オブジェクト**以外の任意

表 5.16 PDF_add_table_cell( ) の組版オプション

オプション	説明
avoidword-splitting	（論理値。テキストフローセルに対してのみ意味を持ちます）true にすると、表組版機能は、そのテキストフローが、そのテキストをその表セル内へはめ込む際に、少なくとも 1 つの強制的な単語分割を必要とするかどうかをチェックします。もしそうであれば、そのセル幅を拡げて、単語分割を避けようとします。デフォルト：true
colscale-group[1]	（文字列）列を追加したい列グループの名前。長いテキストをまるごと収めるために、グループの 1 つの列を拡げる必要があるときは、そのグループのすべての列が統一的に拡大されます。セルが複数の列にわたるときは、それらの列は自動的に伸縮グループを形成します。
colspan	（整数）セルがわたる列の数。デフォルト：1
colwidth[1]	（float またはパーセント値）column 引数で指定している列の幅。この幅は、ユーザー座標で[2]、または表の最初のはめ込み枠（PDF_fit_table( ) 参照）の幅に対するパーセント値で指定することができます。ユーザー座標とパーセント値を混在させてはいけません。すなわち、1 つの表では、すべての列幅定義に、ユーザー座標かパーセント値のどちらかを使う必要があります。テキストを内容とするセル群にわたる列の場合には、その列幅は自動的に増やされることがあります。表セル内に画像・グラフィック・PDF ページがあっても列幅には一切影響を与えません。デフォルト：PDF_fit_table( ) のオプション colwidthdefault を参照
continue-textflow	（論理値。テキストフローに対してのみ意味を持ちます）true にすると、textflow オプションで指定したテキストフローの内容を、もしそれと同じテキストフローハンドルで別のセルへの流し込みを行い、かつそのセルでも continuetextflow=true にするならば、その別のセルへ続けさせることができます。テキストフローの各部分は、セルが追加された順番に配置されていきます。PDFlib は、テキストフロー全体に対するセルサイズの調整は行わず、また、avoidwordsplitting オプションは無視されます。ですので、適切なセルサイズを定義する必要があります。このオプションは、複数の表行または表列にわたっていない複数の表セルの中にテキストフローを配置したいときに便利です。
margin ***marginleft*** ***marginbottom*** ***marginright*** ***margintop***	（float またはパーセント値）セルの左・下・右・上余白を、ユーザー座標系か（0 以上にする必要があります）、またはセルの幅か高さに対するパーセント値（100% 未満にする必要があります）で指定します。指定する余白は、セルの内容に対するはめ込み枠としてはたらくセル内枠を定義します。margin に対するデフォルト：0。それ以外のすべてに対するデフォルト：margin
minrow-height[1]	（float またはパーセント値。廃止）このオプションは、表インスタンスの下端に収まりきらない表行を 2 つの部分へ分割できることを表します。カレントの表インスタンスに収める 1 個目の部分の高さの最小値を、ユーザー座標系で、あるいは表行高さに対する割合で指定できます。次の表インスタンスは 2 個目の部分から始まります。このオプションは注意深く使用するべきであり、かつ、はめ込み枠の高さに比べて非常に高いテキストフローを表行が内容として持つ場合にのみ使用するべきです。このオプションの値は、セル内にテキスト行が少なくとも 1 行は配置できるよう推算されるべきです。そうでないとエラーが発生します。デフォルト：100%、すなわち分割なし

表 5.16 PDF_add_table_cell( ) の組版オプション

オプション	説明
repeatcontent	（論理値）セルまたは表行が複数の表インスタンスに分割されたときに表セルの内容を繰り返すかどうかを指定します。デフォルト：true セルの分割：セルが複数表行にわたっている場合に、その末尾表行（群）がはめ込み枠に収まらないときは、そのセルは分割されます。repeatcontent=true にすると、テキストフロー（繰り返されません）の場合を除き、そのセル内容は次の表インスタンス内で繰り返されます。そうでないときは繰り返されません。 表行の分割：末尾本体行がはめ込み枠に収まらないときは、通常、その表行は分割されずに次の表インスタンスへまるごと配置されます。minrowheight 値を小さくすることによって、末尾本体行を分割させ、その表行の内容のうちのある割合を最初のインスタンスに、残りの部分を次のインスタンスに配置させることもできます。repeatcontent=true にすると、テキストフロー（繰り返されません）の場合を除き、そのセル内容は次の表インスタンス内で繰り返されます。そうでないときは繰り返されません。
return[1]	（文字列）PDF_fit_table( ) は、指定行を配置した後に停止し、指定文字列を返します。文字列は、頭にアンダースコアキャラクター「_」をつけてはいけません。指定行が連動グループに含まれているときは、それはそのグループの最後の行である必要があります。そうでなければエラーが起こります。
rowheight[1]	（float またはパーセント値）row 引数で指定している表行の高さ。この高さは、ユーザー座標で[2]、または表の最初のはめ込み枠（PDF_fit_table( ) 参照）の高さに対するパーセント値で指定することができます。ユーザー座標とパーセント値を混在させてはいけません。すなわち、1 個の表では、すべての表行高さ定義に、ユーザー座標かパーセント値のどちらかを使う必要があります。テキストを内容とするセル群にわたる表行の場合には、その表行高さは自動的に増やされることがあります。表セル内に画像・グラフィック・PDF ページがあっても表行高さには一切影響を与えません。デフォルト：PDF_fit_table( ) のオプション rowheightdefault を参照
rowscale-group[1]	（文字列）表行を追加したい表行グループの名前。長いテキストをまるごと収めるために、グループの 1 個の表行を拡げる必要があるときは、そのグループのすべての表行が統一的に拡大されます。セルが複数の表行にわたるときは、それらの表行は自動的に伸縮グループを形成します。
rowjoin-group[1]	（文字列）表行を追加したい表行グループの名前。グループの表行はすべて、1 つの表インスタンスにまとまります。グループの表行は連続している必要があります。セルが複数の表行にわたっていても、それらの表行は自動的には連動グループを形成しません。
rowspan	（整数）セルがわたる表行の数。デフォルト：1

1. このオプションの最後の指定が優先されます。すなわち、同じ行・列に対するそれ以前の指定は無視されます。
2. より厳密には、最初の表インスタンスを配置するために PDF_fit_table( ) を呼び出した時に効いている座標系。

表 5.17 静的セル内容のための、PDF_add_table_cell( ) のオプションと、PDF_fit_table( ) の caption オプションのサブオプション

オプション	説明
fitgraphics	（オプションリスト。グラフィックでのみ意味を持ちます）PDF_fit_graphics( ) に対するオプションリスト。このオプションリストは、graphics オプションを用いて与えられたグラフィックをセル内に配置するために適用されます。そのはめ込み枠の左下隅が参照点として用いられます。 デフォルト：fitmethod=meet position=center。このオプションリストが、ユーザーが与えるオプション群の頭に付加されます。[1]
fitimage	（オプションリスト。画像とテンプレートに対してのみ意味を持ちます）PDF_fit_image( ) に対するオプションリスト。このオプションリストは、image オプションを用いて与えられた画像またはテンプレートをセル内に配置するために適用されます。そのはめ込み枠の左下隅が参照点として使われます。 デフォルト：fitmethod=meet position=center。このオプションリストが、ユーザーが与えるオプション群の頭に付加されます。[1]

表 5.17 静的セル内容のための、PDF_add_table_cell( ) のオプションと、PDF_fit_table( ) の caption オプションのサブオプション

オプション	説明
fitpath	（オプションリスト。パスオブジェクトに対してのみ意味を持ちます）PDF_draw_path( ) に対するオプションリスト。このオプションリストは、path オプションを用いて指定されたパスオブジェクトがその外接枠内に入ったものをセル内に配置するために適用されます。そのはめ込み枠の左下隅が参照点として用いられます。 デフォルト：fitmethod=meet position=center。このオプションリストが、ユーザーが与えるオプション群の頭に付加されます。[1]
fitpdipage	（オプションリスト。PDI ページに対してのみ意味を持ちます。PDI が利用可能な場合にのみ可）PDF_fit_pdi_page( ) に対するオプションリスト。このオプションリストは、pdipage オプションを用いて与えられたページをセル内に配置するために適用されます。そのはめ込み枠の左下隅が参照点として用いられます。 デフォルト：fitmethod=meet position=center。このオプションリストが、ユーザーが与えるオプション群の頭に付加されます。[1]
fittextflow	（オプションリスト。テキストフローに対してのみ意味を持ちます）PDF_fit_textflow( ) に対するオプションリスト。このオプションリストは、textflow オプションで与えられたテキストフローをセル内に配置するために適用されます。そのはめ込み枠がはめ込み枠として用いられます。 デフォルト：verticalalign=center lastlinedist=descender。このオプションリストが、ユーザーが与えるオプションリストの頭に付加されます。
fittextline	（オプションリスト。テキスト行に対してのみ意味を持ちます）PDF_fit_textline( ) に対するオプションリスト。このオプションリストは、text 引数を用いて与えられたテキストをセル内に配置するために適用されます。そのはめ込み枠の左下隅が参照点として使われます。指定していないオプションは、それぞれのデフォルトに置き換えられます。カレントテキストステータスは考慮されません。 デフォルト：fitmethod=nofit position=center。このオプションリストが、ユーザーが与えるオプション群の頭に付加されます。[1]
graphics	（グラフィックハンドル）このハンドルに紐付けられたグラフィックがこのはめ込み枠内に配置されます。
image	（画像ハンドル）ハンドルに関連づけられた画像が、セル内枠の中に配置されます。
matchbox	（オプションリスト）範囲枠の詳細を表 6.4 に従って入れたオプションリスト
path	（パスハンドル）パスオブジェクトがその外接枠内に入ったものが、fitpath オプションに従ってセル内枠内に配置されます。
pdipage	（ページハンドル）ハンドルに関連づけられた取り込み PDF ページが、セル内枠内に配置されます。
text	（内容文字列）PDF_fit_textline( ) を用いて、オプション fittextline に従って配置したいテキスト。PDF_add_table_cell( ) では、このオプションの値をメソッド引数 text を通じて与えることもできます。
textflow	（テキストフローハンドル）ハンドルに関連づけられたテキストフローが、はめ込み枠内に配置されます。continuetextflow オプションが、複数のセルで使われるテキストフローハンドルに対する動作を制御します。1 個のテキストフローハンドルは表の外で使ってはいけません。

1. 枠の大きさは自動的に算出されます。オプションリストで boxsize オプションを与えても無視されます。

表 5.18 インタラクティブなセル内容のための、PDF_add_table_cell( ) のオプションと、caption オプションのサブオプション（ページスコープでのみ）

オプション	説明
annotation-type	（文字列）表 12.2 に従った、表セル内に挿入したい注釈の種別。
fieldname	（ハイパーテキスト文字列）fieldtype に対するフォームフィールド名。
fieldtype	（文字列）表 12.5 に従った、表セルに挿入したいフォームフィールドの種類。フォームフィールドグループは表の外で定義する必要があります。
fitannotation	（オプションリスト）に対する、表 12.3 に従った注釈オプション群。
fitfield	（オプションリスト）に対する、表 12.6 に従ったフォームフィールドオプション群。

C++ Java C# ***String fit_table(int table, double llx, double lly, double urx, double ury, String optlist)***

Perl PHP ***string fit_table(int table, float llx, float lly, float urx, float ury, string optlist)***

C ***const char *PDF_fit_table(PDF *p,***
int table, double llx, double lly, double urx, double ury, const char *optlist)

表の全部ないし一部分をページに配置します。

table *PDF_add_table_cell( )* を呼び出して取得した有効な表ハンドル。

llx・lly・urx・ury 表インスタンスを配置したい長方形（はめ込み枠）の左下隅と右上隅の座標を、ユーザー座標で指定します。この 2 個の隅は逆の順に指定することもできます。

optlist 配置の詳細を表 5.19 に従って指定したオプションリスト。以下のオプションが使えます：

- 一般オプション：*errorpolicy*（表 1.5 参照）
- 表 6.1 に従ったはめ込みオプション：*fitmethod*・*position*・*showborder*
- 一般表オプション：
  blind・*colwidthdefault*・*horshrinklimit*・*rewind*・*rowheightdefault*・*vertshrinklimit*
- 表内容：***caption***・*header*・*footer*
- 表装飾：*fill*・*firstdraw*・*gstate*・*round*・*stroke*
- 開発・デバッグの視覚支援：*debugshow*・*showcells*・*showgrid*
- 表 14.4 に従った、短縮構造エレメントタグ付けのためのオプション（**ページ**スコープでのみ可）：*tag*。このオプションを使うと、自動表タグ付けを引き起こすことができます（詳しくは PDFlib チュートリアルを参照）。

戻り値 メソッドから戻った原因を示す文字列。

- *_stop*：表のすべての表行の処理が完了。
- *_boxfull*：配置するべき表行はまだあるが、表のはめ込み枠に充分は余地が得られない。もう一度 *PDF_fit_table( )* を呼び出して残りの表行を処理する必要があります。
- *_error*：エラーが発生。問題に関する詳細を得るには、*PDF_get_errmsg( )* を呼び出します。その問題を視覚化させるには *debugshow=true* と設定します。
- それ以外の文字列：*PDF_add_table_cell( )* への呼び出しで *return* オプションに与えた文字列。

エラー動作は、*errorpolicy* オプションで変えることができます。

詳細　表をページに配置します。これ以前に *PDF_add_table_cell( )* を呼び出して表にセルを入れておく必要があります。表全体がはめ込み枠に収まりきらないときは、最初の表インスタンスが配置されます。以後このメソッドを呼び出して、さらなる表インスタンスを配置していくことができます。表セルの内容は、以下の順序で配置されます。

- 塗り：*fill*　オプションで指定する領域は、次の順序で塗られます：*table*・*colother*・*colodd*・*coleven*・*col#*・*collast*・*rowother*・*rowodd*・*roweven*・*row#*・*rowlast*・*header*・*footer*。
- 範囲枠の塗り：*matchbox* 定義で定義した領域群。
- 内容：指定したセル内容は、次の順序で配置されます：画像、グラフィック、取り込み PDF ページグラフィック、パスオブジェクト、テキストフロー、テキスト行、注釈、フォームフィールド。
- 範囲枠の罫線：*matchbox* 定義で定義した領域群。
- 罫線：*stroke* オプションで指定する線は、*stroke* オプションの *linecap*・*linejoin* サブオプションに従って、次の順序で描線されます：*other*・*horother*・*hor#*・*horlast*・*vertother*・*vert#*・*vertlast*・*frame*（横線と縦線の順序は、*firstdraw* オプションで変えることができます。）。複数の行・列にわたるセルの中では、罫線は引かれません。同様に、枠線の装飾を指定している範囲枠をつけているセルのまわりには、線は描線されません（その範囲枠がセル内枠を使っているときを除き）。表の枠線 *verto*・*horo*・*vertN*・*horN* は、*frame* を指定しているときは無視されます。
- 名前付き範囲枠：これらには、表メソッド以外の、注釈・フォームフィールド・画像・グラフィックなどをつけることができます。

自動表タグ付け：オプション *tag={tagname=Table}* を用いると、表行・表セルのための構造エレメント群を自動生成することができます（PDFlib チュートリアル参照）。

スコープ　一般には**ページ・パターン・テンプレート・グリフ**。ただし、表がフォームフィールドか注釈かタグを含んでいる場合には、**ページ**スコープのみ許容されます。

表 5.19　PDF_fit_table( ) **のオプション**

オプション	説明
blind	（論理値）`true` にすると、すべての計算が行われますが、出力は作成されません。組版の結果は、PDF_info_table( ) で調べることができます。デフォルト：`false`

表 5.19 PDF_fit_table( ) のオプション

オプション	説明
caption	（オプションリスト）算出されたはめ込み枠からの相対位置にキャプションのためのはめ込み枠を作成し、それにさまざまな種別の内容をはめ込みます。以下のオプションを与えることができます（デフォルト：キャプションなし）： ***fitbox*** （4 個の float かパーセント値で絶対または相対座標を表したもの。必須）枠の対角線上の 2 隅の座標をユーザー座標で表したもの。値がパーセント値か相対値の場合には、それは表インスタンスの対応する隅 {llx lly urx ury} からのオフセットを示します。llx か urx に対応するパーセント値は、表インスタンス幅に対するパーセント値であり、lly か ury に対応するパーセント値は、表インスタンス高さに対するパーセント値です。このはめ込み枠は、その内容のサイズに合わせて自動的に調整されません。指定された範囲枠はそのはめ込み枠を記述します。これを使って、このキャプションはめ込み枠を描いたり、PDF_info_matchbox( ) を用いてその範囲枠を取得したりすることができます。 この fitbox オプションの使用例： 表インスタンスの上端の、高さ 20 のはめ込み枠：fitbox={0r 100% 0r 20r} 表インスタンスの右側の、幅 20、下端からのオフセット 20% のはめ込み枠：fitbox={100% 20% 20r 0r} これに加えて、以下のオプションを使えます： ▸ 表 5.17 に従った、静的セル内容のためのオプション：fitgraphics・fitimage・fitpath・fitpdipage・fittextflow・fittextline・graphics・image・matchbox・path・pdipage・***text***・textflow ▸ 表 5.18 に従った、インタラクティブセル内容のためのオプション（ページスコープでのみ可）：annotationtype，fieldname・fieldtype・fitannotation・fitfield ▸ 表 14.4 に従った、短縮構造エレメントタグ付けのためのオプション：tag。これを使うと、キャプション内容の親エレメントや、キャプション内容を構成する複数のエレメントのためのコンテナとしてグループ化エレメントを挿入することができます。
colwidth-default	（float またはキーワード。最初の PDF_fit_table( ) への呼び出しでのみ意味を持ちます）テキスト行セルもテキストフローセルも含まない、かつ PDF_add_table_cell( ) の colwidth オプションが指定されなかった列に対するデフォルト幅。このデフォルト幅は、絶対値として、またはキーワードとして指定することができます。値 0（ゼロ）はキーワード distribute と同等です。以下のキーワードを使えます（デフォルト：auto）： ***auto*** 幅が指定されていない、かつテキスト行セルのみを内容とする表列は、そのテキストの幅を持ちます。はめ込み枠の残りの幅が、幅を指定されていない、非テキスト行セルを持つすべての表列に分配されます。そのような表列が存在している場合には、表ははめ込み枠の幅いっぱいに広がります。 ***distribute*** はめ込み枠の幅が、幅が指定されていない、かつテキスト行セルを含んでいないすべての表列に分配されます。そのような表列が存在している場合には、表ははめ込み枠の幅いっぱいに広がります。 ***minimum*** 幅が指定されていない、テキスト行セルのみを内容とする表列は、そのテキストの幅、すなわち、そのテキストを保持できる可能なかぎり小さな幅を持ちます。 最小限の幅を持つ列を生成するには、小さな値を与えることができます（0.1 など）。テキスト行かテキストフローを含むすべての列の幅は自動的に調整されます（PDFlib チュートリアル参照）。
debugshow	（論理値）true にすると、表が高すぎか、または幅が広すぎか、またはそのセルが小さすぎのエラーがすべて出なくなり、かわりにログ記録されます。できあがる表インスタンスは、表としてはおかしくなりますが（たとえば表が不完全になったり、正しくなかったり）、デバッグの助けとして作成されます。デフォルト：false

表 5.19 PDF_fit_table( ) のオプション

オプション	説明
fill	（オプションリストのリスト）このオプションを使うと、行・列に色を塗ることができます（1 個のセルに色を塗るには、matchbox オプションが使えます。143 ページ「6.2 範囲枠」参照）： ***area*** （キーワード）塗りたい表の領域。 ***col#*** 表の列番号 # ***collast*** 最後の列 ***coleven*** 偶数番号の列すべて（PDF_add_table_cell( ) の col に従って） ***colodd*** 奇数番号の列すべて ***colother*** 未指定の列すべて ***row#*** 表の行番号 # ***rowlast*** 表インスタンスの最後の本体行 ***roweven*** 偶数番号の表行すべて（PDF_add_table_cell( ) の row に従って） ***rowodd*** 奇数番号の表行すべて ***header*** ヘッダーグループの表行すべて ***footer*** フッターグループの表行すべて ***rowother*** 未指定の本体行すべて ***table*** 表領域全体（すなわち表の表行すべて） 表 7.1 に従った以下のグラフィック書式オプションも使えます： ***fillcolor・shading*** 表領域の塗りを行わないためには fillcolor=none を使用します。 例： 表の表行すべてを赤く塗る：fill = { {area=table fillcolor=red} } 奇数番号の表行を緑で、偶数番号の表行を赤で塗る： fill = { {area=rowodd fillcolor=green} {area=roweven fillcolor=red} }
firstdraw	（キーワード）横線と縦線を生成する順序（デフォルト：vertlines）： ***horlines*** 横線をまず生成します。 ***vertlines*** 縦線をまず生成します。
footer	（整数）表の定義における、終了部（フッター）表行の数。各表インスタンスの下端に現れます。デフォルト：0（フッター行なし）
gstate	（PDF_create_gstate( ) に対するオプションのオプションリスト、またはグラフィックステートハンドル）表装飾（セル内容は除く）のためのグラフィックステートオプション群またはハンドル。デフォルト：グラフィックステートなし、すなわち、カレント設定が用いられます
header	（整数）表の定義における、開始部（ヘッダー）表行の数。各表インスタンスの上端に現れます。デフォルト：0（ヘッダー行なし）
horshrinklimit	（float またはパーセント値）表のはめ込み枠に収めるために表を縮めるときに使われる横縮小倍率の下限か（パーセント値を与えるとき）、または表の幅とはめ込み枠の幅の差の絶対指定（float を与えるとき）。デフォルト：50%
rewind	（−1・0・1 のいずれかの整数）表の状態が、同じ表ハンドルを指定して PDF_fit_table( ) を呼び出したいずれかの時の前の状態へリセットされたのちに、その表が通常どおり配置されます（デフォルト：0）。 ***1*** PDF_fit_table( ) を最初に呼び出した時の前の状態へ巻き戻し。 ***0*** 表をリセットしません。 ***-1*** PDF_fit_table( ) を直前に呼び出した時の前の状態へ巻き戻し。
round	（float）この値が 0 でないときには、表の長方形の fill・stroke オプションに対する隅は、指定した半径の円弧で丸められます。半径が負値の場合には円弧は内側へへこみます。デフォルト：0、すなわち丸めなし

表 5.19 PDF_fit_table( ) のオプション

オプション	説明
rowheight-default	(float またはキーワード。最初の PDF_fit_table( ) への呼び出しでのみ意味を持ちます) PDF_add_table_cell( ) の rowheight オプションが指定されなかった表行に対するデフォルト高さ。このデフォルト高さは、絶対値として、またはキーワードとして指定することができます。float 値が指定された場合には、それがテキスト枠高さよりも小さな場合を除き、それはデフォルト表行高さとして用いられます。値 0 (ゼロ) はキーワード distribute と等価です。以下のキーワードを使えます (デフォルト : auto) : ***auto*** 高さを指定されていない、テキスト行セルのみを内容とする表行は、テキスト枠の高さの 2 倍の高さを持ちます。はめ込み枠の残りの高さは、高さを指定されていない、非テキスト行セルを持つすべての表行に分配されます。そのような表行が存在している場合には、表ははめ込み枠の高さいっぱいに広がります。 ***distribute*** はめ込み枠の高さが、高さが指定されていない、テキスト行とその他のすべての表行に分配されます。そのような表行が存在している場合には、表ははめ込み枠の高さいっぱいに広がります。 ***minimum*** 高さが指定されていない、テキスト行セルのみを内容とする表行は、そのテキスト枠の高さ、すなわち、そのテキストを保持できる最も小さな高さを持ちます。テキスト行セルの高さを増やすには boxsize か margin オプションを使います。 最小限の高さを持つ表行を生成するには、小さな可能な値を与えることができます (1 など)。テキスト行かテキストフローを含むすべての表行の高さは自動的に調整されます (PDFlib チュートリアル参照)。
showcells	(論理値) true にすると、各セル内枠の辺がカレントグラフィックステートを使って描線されます。ページスコープでは、PDF/A がアクティブでない場合には、各セルはさらに、そのセルの内容を記述した注釈を用いて修飾されますので、これは表関連の諸問題を分析するために有用でしょう。デフォルト : false
showgrid	(論理値) true にすると、すべての列と行の縦・横の境界が描線されます。デフォルト : false
stroke	(オプションリストのリスト) このオプションを使うと、セルの辺に描線を作成することができます : ***line*** (キーワード) 描線したい表の線。 ***vert#*** 列番号 # の右の辺の縦線。vert0 は表の左の辺 ***vertfirst*** 最初の縦線 (vert0 と同義) ***vertlast*** 最後の縦線 ***vertother*** 未指定の縦線すべて ***hor#*** 表行番号 # の下の辺の横線。row0 は表の上の辺 ***horfirst*** 表インスタンスの最初の横線 ***horother*** 未指定の横線すべて ***horlast*** 表インスタンスの最後の横線 ***frame*** 表の外辺 ***other*** 指定していない線すべて 表 7.1 に従った以下のグラフィック書式オプションも使えます : ***dasharray・dashphase・linecap・linejoin・linewidth・strokecolor*** 表領域に対して描線を行わないためには strokecolor=none を使用します。 例 : すべての線を黒で線幅 1 で描線 : stroke = {line=other} 外辺の線を線幅 0.5 で描線 : stroke = { {line=frame linewidth=0.5} } 外辺の線を線幅 0.5 で、他の線すべてを線幅 0.1 で描線 : stroke = { {line=frame linewidth=0.5} {line=other linewidth=0.1} }
vertshrink-limit	(float またはパーセント値) 表のはめ込み枠に収めるために表を縮めるときに使われる縦縮小倍率の下限か (パーセント値を与えるとき)、または表の高さとはめ込み枠の高さの差の絶対指定 (float を与えるとき)。デフォルト : 90%

C++ Java C# **double info_table(int table, String keyword)**

Perl PHP **float info_table(int table, string keyword)**

C **double PDF_info_table(PDF *p, int table, const char *keyword)**

もっとも最近に配置した表インスタンスに関連する表情報を取得します。

table PDF_add_table_cell( ) を呼び出して取得した有効な表ハンドル。この表ハンドルは、少なくとも 1 回は PDF_fit_table( ) を呼び出すときに使っている必要があります。なぜなら戻り値は、表インスタンスをページに配置した後にのみ意味を持つからです。

keyword ほしい情報を指定したキーワード：

- 表 6.3 に従った、オブジェクトはめ込みの結果をクエリーするためのキーワード：
  boundingbox・fitscalex・fitscaley・height・objectheight・objectwidth・width・x1・y1・x2・y2・x3・y3・x4・y4
- 表 5.20 に従ったさらなるキーワード：
  firstbodyrow・horboxgap・horshrinking・lastbodyrow・returnreason・rowcount・rowsplit・tableheight・tablewidth・vertboxgap・vertshrinking・xvertline#・yhorline#

戻り値 *keyword* で要求した何らかの表特性の値。このメソッドはブラインドモードであっても、正しい位置情報を返します。要求されたキーワードがテキストを生み出す場合には、文字列番号が返され、その対応する文字列を、PDF_get_string( ) を用いて取得する必要があります。

スコープ **オブジェクト**以外の任意

表 5.20 PDF_info_table( ) のキーワード

キーワード	説明
firstbodyrow	もっとも最近に配置した表インスタンスの最初の本体行の番号
horboxgap	表インスタンスの幅とはめ込み枠の幅の差。表を縮めなければならなかったときは、この値ははめ込み枠の幅からの逸脱を示します（すなわち負の値）。
horshrinking	計算された表幅に対する横縮小倍率をパーセント値として表したもの。表を横に縮めなければならなかったときは、この値は縮小比を示しますが、そうでなければ 100 になります。
lastbodyrow	もっとも最近に配置した表インスタンスの最後の本体行の番号
returnreason	返った原因に対する文字列番号
rowcount	もっとも最近に配置した表インスタンスの表行の数（ヘッダー・フッターを含む）
rowsplit	最後の行を分割しなければならなかったときは 1、そうでなければ 0
tableheight tablewidth	表全体の幅と高さ
vertboxgap	表インスタンスの高さとはめ込み枠の高さの差。表を縮めなければならなかったときは、この値ははめ込み枠の高さからの逸脱を示します（すなわち負の値）。
vert-shrinking	計算された表の高さに対する縦縮小倍率をパーセント値として表したもの。表を縦に縮めなければならなかったときは、この値は縮小比を示しますが、そうでなければ 100 になります。
xvertline#	番号 # の縦線の x 座標。xvertline0 は表の左の辺。
yhorline#	番号 # の横線の y 座標。yhorline0 は表の上の辺。

C++ Java C# ***void delete_table(int table, String optlist)***

Perl PHP ***delete_table(int table, string optlist)***

C ***void PDF_delete_table(PDF *p, int table, const char *optlist)***

表と、関連するすべてのデータ構造を削除します。

table *PDF_add_table_cell( )* を呼び出して取得した有効な表ハンドル。

optlist クリーンアップオプションを表 5.21 に従って指定したオプションリスト。

詳細 このメソッドで削除しなかった表は、カレントの**文書**スコープを終えると自動的に削除されます。

スコープ 任意

表 5.21 PDF_delete_table( ) のオプション

オプション	説明
keephandles	（論理値）false にすると、PDF_add_table_cell( ) の textflow・image・graphics・pdipage オプションに与えたハンドルがすべて自動的に削除されます。デフォルト：false

6 オブジェクトのはめ込みと範囲枠

この章の API メソッド：

- *PDF_info_matchbox()*

6.1 オブジェクトのはめ込み

PDFlib のはめ込みアルゴリズムは、長方形のグラフィックオブジェクトを、点に、または横線か縦線に、あるいは長方形に相対的に配置します。このはめ込みアルゴリズムはいくつかのメソッドに実装されています：

- *PDF_fit_textline()*・*PDF_info_textline()*
- *PDF_fit_image()*・*PDF_info_image()*
- *PDF_fit_graphics()*・*PDF_info_graphics()*
- *PDF_fit_pdi_page()*・*PDF_info_pdi_page()*
- *PDF_draw_path()*・*PDF_info_path()*
- *PDF_add_table_cell()*（*fitgraphics*・*fitimage*・*fitpdipage*・*fitpath*・*fittextline* オプションに対するオプションリストを通じて）
- *PDF_fit_table()*
- *PDF_fill_*block()*

注 テキストフローに対するはめ込みオプション群は若干異なっていますので、ここではなく、106 ページ「5.2 テキストフローによる複数行テキスト」で記しています。

表 6.1 に、これらのはめ込みメソッドに与えることのできるはめ込みオプションを挙げます。すべてのオプションがすべてのメソッドで利用できるわけではなく、また、いくつかのオプションの動作はメソッドによって若干変わります。詳しくは表 6.1 を参照。以下のオプションがはめ込みオプションのグループを形成します：

alignchar・*boxsize*・*dpi*・*fitmethod*・*margin*・*matchbox*・*minfontsize*・*orientate*・*position*・*refpoint*・*rotate*・*scale*・*stamp*・*showborder*・*shrinklimit*

オブジェクト枠　はめ込みアルゴリズムは、配置するオブジェクトを囲む最小の長方形を算出します。この長方形を**オブジェクト枠**と呼びます。これは、オブジェクトの種類に応じて変えることができます：

- テキスト行（*PDF_fit/info_textline()*・一行テキストブロック・表セル）：その幅は、テキスト文字列の幅（横書きの場合）か、最も広いグリフの幅（縦書きの場合）です。テキスト枠のデフォルト高さは、選択したフォントの *capheight* です。これは、*matchbox* オプションの *boxheight* サブオプションで変えることもできます。字間は、最後のグリフの後には適用されません。
- 画像とテンプレート（*PDF_fit/info_image()*・イメージブロック・表セル）：*matchbox* オプションの *clipping* サブオプションを用いて、オブジェクトの一部分をオブジェクト枠として定義することができます。TIFF・JPEG 画像にクリッピングパスがある場合には、*matchbox* オプションの *innerbox* サブオプションを設定すると、それを囲む、辺が座標軸に平行な最小の長方形がオブジェクト枠として用いられます。*PDF_begin_template_ext()* の *transform* オプションが与えられている場合には、その指定された変形がテンプレートに適用されます。
- グラフィック（*PDF_fit/info_graphics()*）：*matchbox* オプションのサブオプション *clipping* を使って、オブジェクトのどこかの一部分をオブジェクト枠として定義することがで

きます。このオブジェクト枠は、SVG グラフィックの幅と高さによって、あるいは *forcedwidth* と *forcedheight* によって定義されます。これらの値が 0 の場合には、以下が成立します：もし *fitmethod* が *nofit* 以外であるか、またははめ込み枠が定義されていない場合には、オブジェクト枠のサイズは *fallbackwidth* と *fallbackheight* によって定義されます。もし *fitmethod=nofit* かつはめ込み枠が定義されている場合には、オブジェクト枠のサイズはそのはめ込み枠によって定義されます。

- 取り込み PDF ページ（*PDF_fit/info_pdi_page()*・PDF ブロック・表セル）：*PDF_open_pdi_page()* で用いられたオプション群に従います。*cloneboxes=true* の場合には、可視枠が使用されます（すなわち、CropBox が存在するならそれが、なければ MediaBox が）。*matchbox* オプションの *clipping* サブオプションを用いると、オブジェクトの一部分をオブジェクト枠として用いることができます。*PDF_begin_template_ext()* の *transform* オプションが与えられている場合には、その指定された変形がテンプレートに適用されます。
- パスオブジェクト（*PDF_draw/info_path()*・表セル）：パスを囲む、辺が座標軸に平行な最小の長方形がオブジェクト枠として用いられます。オブジェクト枠は、*boxsize*・*position* オプションの値がゼロでないときにのみ算出されます。*linewidth*・*miterlimit* オプションは無視されます。
- 表インスタンス（*PDF_fit_table()*）：表インスタンスを囲む、辺が座標軸に平行な最小の長方形がオブジェクト枠として用いられます。

参照点　参照点は、オブジェクト枠を配置するためのアンカーとして用いられます。これは以下のように定義されます：

- *PDF_fit_*()*・*PDF_draw_path()* の場合：メソッドの引数 *x*・*y*。
- *PDF_info_*()* の場合：点 *(o, o)*。
- *PDF_add_table_cell()*・*PDF_fit_table()*・*PDF_fill_*block()*：表セル・表インスタンス・PDFlib ブロックの左下隅。

これに加え、以下のメソッドでは *refpoint* オプションを用いて参照点をオーバーライドすることもできます：*PDF_fit_graphics()*・*PDF_info_path()*・*PDF_fill_*block()*。

はめ込み枠と参照線分　オブジェクト枠がその中に配置される長方形を、**はめ込み枠**と呼びます。これは参照点 *(x, y)* をその左下隅とし、その寸法は *boxsize* オプションの 2 個の値で指定されます：

```
lower left corner  = (x, y)
upper right corner = (x + boxsize[0], y + boxsize[1])        (topdown=falseのとき)
upper right corner = (x + boxsize[0], y - boxsize[1])        (topdown=trueのとき)
```

上述の定義に加えて、はめ込み枠は以下のように変更することもできます：

- テキスト行：はめ込み枠は *margin* オプションで縮めることもできます。
- 表セル：はめ込み枠は、セル内枠、すなわち、*margin** オプションによって変更を受けたセル枠によって定義されます。
- 表インスタンス：はめ込み枠は引数 *llx/lly/urx/ury* によって定義されます。
- PDFlibブロック：はめ込み枠はデフォルトではブロックの*Rect*プロパティによって定義されますが、*refpoint*・*boxsize* オプション（一方のみでも両方でも）で変更することもできます。

上記のうち後半 3 つの場合には、はめ込み枠はつねに得られます。それ以外の場合には、はめ込み枠は、*boxsize*オプションにゼロでない2個の値を指定したときにのみ得られます。

boxsize[0]=0 のとき、枠は 1 本の縦線へ縮退します。はめ込みアルゴリズムはオブジェクト枠を、この線分に対して相対的に配置します。同様に、*boxsize[1]=0* のとき、枠は、結果としてできる横線に対して相対的に配置されます。この縦線または横線を、**参照線分**と呼びます。

オブジェクト枠を配置　オブジェクト枠は、さまざまな方式で配置することができます：

- はめ込み枠が得られないとき、オブジェクトは参照点に対して相対的に配置されます（表セル・表インスタンス・PDFlib ブロックの場合を除く）：オブジェクト枠の左下隅が参照点の位置に置かれます。*position* オプションを使うと、オブジェクト枠内のそれ以外の点を選ぶこともできます。たとえば、*position=center* にすると、オブジェクト枠の中心点が参照点の位置に置かれます。
  scale オプションは、画像・グラフィック・テンプレート・パスオブジェクト・取り込み PDF ページに対して効力を持ちます。*dpi* オプションは、画像に対して効力を持ちます。*fitmethod* オプションはこの場合無視されます。
  パスオブジェクト：*position={0 0}* にすると、外接枠は算出されず、パスオブジェクトの原点が参照点の位置に置かれます。
- 参照線分に対して相対的に（表セル・表インスタンス・PDFlib ブロックの場合を除く）：これは、上述のようにオブジェクトを参照点に対して相対的に配置する場合と同様に動作します。これに加えて、*position* オプションも、参照点として作用する線分上の点を定義します。
- はめ込み枠に対して相対的に：*fitmethod* オプションは、オブジェクト枠をはめ込み枠内に強制的に収めるかどうか、およびどのように収めるかを指定します。*fitmethod=nofit* にすると、はめ込み枠へ収めるための動作は何も行われません。*fitmethod* のそれ以外の値は、表 6.2 に従って具体的なはめ込みアルゴリズムを定義します。
  この場合は、*scale*・*dpi* オプションは無視され、*margin*・*shrinklimit*・*showborder* オプションは効力を持ちます。
  オブジェクト枠の左下隅が、はめ込み枠の左下隅の位置に置かれます。*position* オプションを用いると、オブジェクト枠内のそれ以外の点と、同時にそれに対応するはめ込み枠内の点を選ぶことができます。たとえば、*position=center* は、オブジェクト枠の中心点をはめ込み枠の中心点の位置に置きます。

オブジェクトはめ込みの結果をクエリーするための共通キーワード　オブジェクトはめ込みの結果は、そのオブジェクトをページ上に実際に配置しないでクエリーすることもできます。これを使うと、実際にページ内容を作成する前に組版上の決定を行うことができます。組版結果をクエリーするためには、オブジェクトに対するはめ込みオプション群を、個別の *PDF_info_*()* メソッドに与えることができます。表 6.3 に、はめ込み結果をク

表 6.1 さまざまなメソッドのはめ込みオプション

オプション	説明
align	(float 2 個のリスト。パスオブジェクトのみ) パスオブジェクトの回転を定義する方向ベクトルの座標をユーザー座標で指定。パスオブジェクトの座標系の x 方向が、指定したベクトルと平行になります。座標を両方 0 にしてはいけません。ここで算出された回転が、orientate オプションで定義される回転に加えられます。デフォルト：{1 0}、すなわち追加の回転なし
alignchar	(Unichar < 0xFFFF またはキーワード。テキスト行でのみ可) 指定したキャラクターがテキスト内に見つかったときは、その左下隅が参照点の位置に置かれます。orientate=north または south の横書きテキストについては、position オプションで与える 1 番目の値が位置を定義します。orientate=west または east の縦書きテキストについては、position オプションで与える 2 番目の値が位置を定義します。 このオプションが存在するとき、組版されたテキストははめ込み枠からはみ出す可能性があります。このオプションは、指定した整列キャラクターがテキスト内に存在しないときは無視されます。指定したキャラクターがフォントまたはエンコーディング内に見つからないときは、glyphcheck=error であれば例外が発生します。glyphcheck がそれ以外の値であれば、alignchar オプションは、キャラクターが得られないときには警告を出さずに無視されます。 値 0 とキーワード none は整列キャラクターをなしにします。指定した fitmethod が適用されますが、ただし alignchar の強制位置合わせのために、テキストははめ込み枠内には配置できません。デフォルト：none
attachment-point	(文字列。パスオブジェクトでのみ可) 取付点の名前：パスオブジェクトは、指定した取付点が参照店の位置になるよう配置されます。fitmethod が nofit 以外の場合には、オブジェクトはまず、指定された方式に従ってはめ込み枠内に配置されます。デフォルト：パスオブジェクトの原点
blind	(論理値。テキストフローの場合は表 5.13 を参照) true にすると、出力は生成されませんが、すべての計算は実行され、その組版結果を適切なメソッド PDF_info_*( ) でチェックできます。デフォルト：false
boxsize	(2個のfloatのリスト。表では不可) オブジェクト (rotateオプションに従って回転されている場合もある) がそれに対して相対的に配置されるはめ込み枠の幅と高さ。はめ込み枠の左下隅が、参照点(x, y)の位置に置かれます。オブジェクトの配置は、position・fitmethodオプションによって制御されます。幅=0にすると高さのみが考慮され、高さ=0にすると幅のみが考慮されます。これらの場合にはfitmethodオプションは無視され、オブジェクトはpositionオプションに従って、(x, y)から(x, y+高さ) (あるいは上から下への座標系では(x, y-高さ)) の縦線に対して、または(x, y)から(x+幅, y)の横線に対して相対的に配置されます。 PDF_fill_*block( ) でのデフォルト：ブロックの Rect プロパティの幅と高さ。これらのメソッドは、絶対座標も受け付けますし、ブロックの元の幅か高さに対する相対座標も受け付けます。 それ以外のすべてのはめ込みメソッドでのデフォルト：{0 0}
dpi	(2 個の float かキーワードのリスト。画像でのみ可) 求める横方向と縦方向の画像解像度をインチあたりピクセル数で指定した 1 個か 2 個の値。このオプションは、画像内のピクセル数を変える (ダウンサンプリング) わけではありません。値を 1 個だけ指定すると、それは両方の方向に対して用いられます。値 0 にすると、画像の内部解像度がもし得られればそれが用いられ、そうでないなら 72 dpi となります。キーワード internal はゼロと同等です。このオプションによって生じる拡縮は、カレントユーザー座標系に対して相対的になります。すなわち、座標系が拡縮されているときは、最終的な物理解像度は与えた値とは異なるものになります。scale オプションが、この dpi 値に加えて適用されます。 fitmethod オプションにキーワード auto・meet・slice・entire のいずれかをつけて与えているときには、これらの dpi 値はその画像のアスペクト比のみを指定し、その絶対サイズを指定しません。。デフォルト：internal

表 6.1 さまざまなメソッドのはめ込みオプション

オプション	説明
fitmethod	（キーワード。テキストフローの場合は表 5.13 参照）指定したはめ込み枠内にオブジェクトを収めるために用いる方式。使えるキーワードは表 6.2 を参照してください。nofit 以外のキーワードは、はめ込み枠を指定していないときには無視されます。 デフォルト：表の場合は clip、パスオブジェクトの場合は meet、それ以外の場合は nofit
margin	（float のリスト。テキスト行でのみ可）はめ込み枠の横と縦の追加の縮小を記述した 1 個か 2 個の float 値。デフォルト：0
matchbox	（オプションリスト。パスオブジェクトでは不可。テキストフローの場合は表 5.13 参照）表 6.4 に従った、範囲枠を生成するためのオプションリスト
minfontsize	（float またはパーセント値。。テキストフローの場合は表 5.13 を参照）fitmethod=auto で shrinklimit を超えたときにテキストをはめ込み枠に収めるために縮小する際の最小許容文字サイズ。ユーザー座標で、またははめ込み枠の高さに対するパーセント値で指定します。この下限に達したときは、テキストは、指定した minfontsize を文字サイズとして生成されます。デフォルト：0.1%
orientate	（キーワードまたは float。表では不可。テキストフローの場合は表 5.13 を参照）オブジェクトの、カレント座標系に対する相対的な向き。デフォルト：north。 パスオブジェクトの場合には、任意の回転角（度単位で）を指定することもできますが、それ以外の種類のオブジェクトの場合にはできません。パスオブジェクトの境界枠は、そのパスオブジェクトを回転させた後に算出されます。すべてのメソッドで、以下のキーワードが使えます（それぞれ同等の角度を括弧内に示します）： ***north*** 直立（0） ***east*** 右倒し（270） ***south*** 倒立（180） ***west*** 左倒し（90）
position	（float かキーワードのリスト）参照点・参照線分・はめ込み枠のいずれかに対するオブジェクト枠の相対的な位置を指定した 1 個か 2 個の値。これらの値は、オブジェクト枠内の位置を指定します。横位置を枠の幅に対するパーセント値として（1 番目の値）、縦位置を枠の高さに対するパーセント値として（2 番目の値）定義します。この指定した位置が、参照点か、参照線分上の点か、はめ込み枠内の点の位置に置かれます。値はパーセント値を表していますが、パーセント記号なしで指定する必要があります。負の値も許されます。両方の値が等しいときは、値を 1 個だけ与えれば充分です。 デフォルト：表の場合は {0 100}、それ以外なら {0 0}。例： {0 0} オブジェクト枠の左下隅が、参照点か、参照線分の始点か、はめ込み枠の左下隅の位置に置かれます。 {100 100} オブジェクト枠の右上隅が、参照点か、参照線分の終点か、はめ込み枠の右上隅の位置に置かれます。 キーワード left・center・right（x 方向で）または bottom・center・top（y 方向で）を、値 0・50・100 と同等なものとして用いることができます。キーワードを 1 個だけ指定すると、もう 1 つの方向でそれに対応するキーワードが追加されます。例： {left center} または {0 50} 左揃え {center} または {50 50} 縦横中央揃え {right center} または {100 50} 右揃え テキスト行のみ：キーワード auto を、リストの 1 番目の値として用いることもできます。これは、テキストの筆記方向が右書きの場合（アラビア文字・ヘブライ文字等）には right を意味し、そうでない場合には left を意味します。

表 6.1 さまざまなメソッドのはめ込みオプション

オプション	説明
refpoint	（2 個の float のリスト。PDF_fit_graphics( ) ・PDF_fill_*block( ) ・PDF_info_path( ) でのみ可）ブロック内容またはパスをはめ込む際の参照点をユーザー座標で。 PDF_fill_*block( ) でのデフォルト：ブロックの Rect プロパティによって定義される長方形の左下隅。これらのメソッドは、絶対座標も受け付けますし、ブロックの左下隅に対する相対座標も受け付けます。相対座標の例：refpoint={20r 50r} PDF_info_path( ) でのデフォルト：{0 0} PDF_fit_graphics( )：x・y 引数
rotate	（float。表・パスオブジェクトでは不可。テキストフローの場合は表 5.13 参照）参照点を中心とし、指定した値を、回転角を度単位で表したものとして、座標系を回転します。これによって、はめ込み枠とオブジェクトが回転されます。この回転は、オブジェクトが配置された時点でリセットされます。デフォルト：0 表セル内のテキスト行：rotate オプションで 0 以外の値が指定された場合には、表エンジンは、回転されたテキストの外接枠をセル枠内へ、fitmethod と position オプションに従ってはめ込もうと試みます。fitmethod が auto 以外の場合には、セルは必要に応じて適切に拡大されます。
scale	（float のリスト。テキスト行では不可。fitmethod=meet の場合には無視されます）オブジェクトを、参照点を中心として、横方向と縦方向で指定した倍率（パーセント値ではなく）で拡縮します。両方の倍率が等しいときは、値を 1 個だけ指定すれば充分です。負の値の場合には鏡映となります。このオプションの絶対値は、fitmethod オプションにキーワード auto・meet・slice・entire のいずれかをつけて与えているときには無視されます。デフォルト：{1 1}
showborder	（論理値。テキストフローの場合は表 5.13 参照）true にすると、はめ込み枠の境界がカレントグラフィックステートを用いて描線されます。スタンプを生成しているときは、スタンプの境界枠も描線されます。これは開発やデバッグの際に有用でしょう。デフォルト：false
shrinklimit	（float またはパーセント値。テキスト行でのみ可）fitmethod=auto でテキストを収めるために適用される長体比率の下限。デフォルト：0.75
stamp	（キーワード。テキスト行でのみ可。boxsize を指定していない場合には無視されます。テキストフローの場合は表 5.13 参照）このオプションを使うと、boxsize オプションで指定する枠の対角線上に最大限の大きさのスタンプを生成することができます。より具体的には、このテキストははめ込み枠内に対角線上に配置されます。テキスト枠の大きさは、それがはめ込み枠を可能な限り覆いつつもそのテキスト枠の縦横比を保持するように選ばれます（すなわち、スタンプを構成するテキストは可能な限り大きくなります）。オプション fontsize・fitmethod・position は無視されます。オプション orientate=west と =east は何の意味も持ちません（north・south のみ）。使えるキーワード（デフォルト：none）： ***ll2ur*** スタンプは、左下隅から右上隅への対角線に沿って置かれます。 ***ul2lr*** スタンプは、左上隅から右下隅への対角線に沿って置かれます。 ***none*** スタンプは作成されません。

表 6.2 さまざまなメソッドの fitmethod オプションのキーワード。テキスト行に対する各キーワードの典型的効果をあわせて図示。fontsize オプションにはすべての図で同じ値を用いています。

キーワード	説明
auto	この方式は、オブジェクトをはめ込み枠内に自動的に収めようと試みます： オブジェクトがはめ込み枠に収まる場合には、動作は nofit 方式と同じです。すなわち、オブジェクトは拡縮されずに配置されます。 オブジェクトがはめ込み枠より大きい場合には、オブジェクトは以下のようにサイズが縮小されます： ▸ テキスト行：テキストを横に縮めて（つぶして）はめ込み枠に収められるような縮小倍率が算出されます。算出された倍率が shrinklimit オプションよりも小さいときは、meet 方式が適用され、すなわち、テキストが収まるまで、または文字サイズ 0.001 に達するまで文字サイズが縮小されます。 ▸ PDF_fit_table( )：表枠がはめ込み枠より狭い場合にはそれをはめ込み枠の幅へ拡げます。この点以外の動作は meet 方式と同じです。 ▸ その他のオブジェクト種別：動作は meet 方式と同じです。 言い換えれば、fitmethod=auto を用いると、オブジェクトの寸法は縮められることがありますが、拡げられることは一切ありません。
clip	オブジェクトを置き、はめ込み枠の辺で図形的に切り落とします。 PDF_fit_table( )：算出された表枠は、はめ込み枠の下辺で論理的に切り落とされ、次のはめ込み枠へ続くことができます。論理的切り落としは、PDF_fit_textflow( ) と同様であり、PDF_fit_image( ) 等での図形的切り落としとは異なります。表枠は、はめ込み枠内で position オプションに従って配置されます。
entire	オブジェクト枠を、はめ込み枠全体を覆うように拡縮します。一般に、この方式ではオブジェクトはつぶされます。position オプションは何ら効力を持ちません。 PDF_fit_table( )：clip に似ています。表枠がはめ込み枠よりも小さいときは、表枠がはめ込み枠全体を覆うまで、表枠のセル群が一様に拡大されます（セルの内容は拡大されません）。
meet	オブジェクトを position オプションに従って置き、はめ込み枠内にまるごと収まるよう、縦横比を保って拡縮します。一般に、オブジェクト枠の少なくとも 2 つの辺が、対応するはめ込み枠の辺と重なることになります。 PDF_fit_table( )：clip に似ています。表枠がはめ込み枠よりも小さいときは、表の横辺か縦辺がはめ込み枠に重なるまで、表枠のセル群が一様に拡大されます（セルの内容は拡大されません）。

表 6.2 さまざまなメソッドの fitmethod オプションのキーワード。テキスト行に対する各キーワードの典型的効果をあわせて図示。fontsize オプションにはすべての図で同じ値を用いています。

キーワード	説明
nofit	オブジェクトを置くだけです。scale オプションが画像とグラフィックに適用されます。画像には dpi オプションも適用されます。 PDF_fit_table( ) : 表が、無限の高さを持つ仮想的なはめ込み枠に対して算出されます。表枠ははめ込み枠内に、position オプションに従って配置されます。列と表行のデフォルトのサイズは、指定したはめ込み枠の高さに対して相対的になります。表を blind モードで組版するには fitmethod=nofit を推奨します。
slice	オブジェクトを position オプションに従って置き、はめ込み枠全体を覆うように、かつオブジェクトの少なくとも 1 つの方向ではめ込み枠内にちょうど収まるよう、縦横比を保って拡縮します。一般に、オブジェクトのもう 1 つの方向ではめ込み枠から一部分がはみ出すので、その分は切り落とされます。 PDF_fit_table( ) : clip に似ています。表枠がはめ込み枠よりも小さいときは、はめ込み枠全体が表枠によって覆われるまで、表枠のセル群が、縦横比を保って一様に拡大されます（セルの内容は拡大されません）。表枠ははめ込み枠内に、position オプションに従って配置されます。表枠のうち、はめ込み枠からはみ出した部分は、はめ込み枠の辺で図形的に切り落とされます。

エリーするためのキーワードを挙げます。*PDF_info_path( )* に対するはめ込み結果は、参照点からの相対位置として表されます。

表 6.3 PDF_info_image( ) ・ PDF_info_graphics( ) ・ PDF_info_path( ) ・ PDF_info_pdi_page( ) ・ PDF_info_table( ) ・ PDF_info_textline( ) を用いたオブジェクトはめ込みの結果をクエリーするための共通キーワード

キーワード	説明
boundingbox	オブジェクトの外接枠に対するパスハンドル
fitscalex ・ ***fitscaley***	オブジェクトを枠にはめ込んだ結果の拡縮倍率
height	オブジェクトの高さをユーザー座標で表したもの
objectheight ・ ***objectwidth***	オブジェクトを読み込むか作成するために関係したすべてのオプションを処理した後のオブジェクトの生のサイズ。このサイズがはめ込みアルゴリズムによって使用されます。
width	オブジェクトの幅をユーザー座標で表したもの
x1 ・ ***y1*** ・ ***x2*** ・ ***y2*** ・ ***x3*** ・ ***y3*** ・ ***x4*** ・ ***y4***	与えられたオプション群に従った、オブジェクトの外接枠の i 番目の長方形隅（i=1, 2, 3, 4）の位置をユーザー座標で表したもの。x1，y1 は左下、x2，y2 は右下、x3，y3 は右上、x4，y4 は左上を表します。

6.2 範囲枠

範囲枠は、長方形の領域であり、テキストや画像を配置した組版処理の結果を保持します。範囲枠を使って、その領域に囲み枠を描くこともでき、中を塗ることもできます。範囲枠の位置をクエリーして、それを他の目的に使うことも可能です。たとえば注釈を作成する際に使えます。範囲枠を使ってオブジェクトを切り抜くこともできます。

6.2.1 範囲枠を定義

範囲枠は、それ専用の API メソッドを用いて定義するのではなく、その対応する要素を作成する組版メソッドにおいて *matchbox* オプションを用いて定義します：

- テキスト行：*PDF_fit_textline()*・*PDF_fill_textblock()* で *textflow=false* を用いる：範囲枠はテキスト行の外接枠を記述します。
- テキストフロー：*PDF_add/create_textflow()*・*PDF_fill_textblock()* で *textflow=true* を用いる：範囲枠は、生成されるテキスト出力の外接枠を記述します。*PDF_fill_textblock()* における範囲枠指定は、インラインテキスト飾りに対する開始として用いることはできず、テキスト全体に対する範囲枠を作成するためにのみ用いることができます。
- 取り込み PDF ページ：*PDF_fit_pdi_page()*・*PDF_fill_pdfblock()* を用いる：範囲枠は、配置されたページの外接枠を記述します。
- 画像・テンプレート：*PDF_fit_image()*・*PDF_fill_imageblock()* を用いる：範囲枠は、配置された画像またはテンプレートの外接枠を記述します。
- グラフィック：*PDF_fit_graphics()*・*PDF_fill_graphicsblock()* を用いる：範囲枠は、配置されたグラフィックの外接枠を記述します。
- 表セル：*PDF_add_table_cell()*：範囲枠は、表セルの外接枠を記述します。

範囲枠は、これらのメソッドの *matchbox* オプションで定義され、その範囲枠が定義されたページの終了まで使用できます。範囲枠は、その名前によって識別され、1 つないし複数の長方形で構成されます。*matchbox* オプションには、以下のサブオプションが使えるオプションリストを与えます：

- 表 7.1 に従ったグラフィック書式オプション：
  borderwidth・*dasharray*・*dashphase*・*fillcolor*・*gstate*・*linecap*・*linejoin*・*shading*・*strokecolor*
- 表 6.4 に従った範囲枠制御オプション
- 表 14.4 に従った、短縮構造エレメントタグ付けのためのオプション（**ページ**スコープでのみ可）：*tag*

注 Matchboxes are not supported in blind mode, i.e. formatting with the blind option.

6.2.2 範囲枠を使う

範囲枠は以下の目的のために使えます：

- 装飾：範囲枠によって定義された長方形は、オプション *fillcolor* または *shading* が指定されている場合には塗られます。範囲枠の罫は、オプション *strokecolor* が指定されており、かつオプション *borderwidth* の値が > 0 の場合には描線されます。範囲枠の装飾はとりわけ表セルに対して有用です。
- 切り抜き：配置する画像・グラフィック・PDF ページの寸法を、*matchbox* オプションの *clipping* サブオプションを用いて変更できます。
- テキストの回り込み：範囲枠をテキストフローの回り込み枠として使うことができます。*PDF_fit_textflow()* でオプション *wrap*・サブオプション *usematchboxes* を用います。

- インタラクティブ要素：範囲枠を注釈かフォームフィールドのターゲット長方形として使うことができます。*PDF_create_annotation()* でオプション *usematchbox* を用います。
- パスオブジェクトを範囲枠内に収めます。*PDF_draw_path()* でオプション *usematchbox* を用います。
- 範囲枠に対応する長方形の詳細は、*PDF_info_matchbox()* で取得できます。これによって、たとえば組版操作の結果をクエリー — したりすることが可能です。

表 6.4 さまざまなメソッドの matchbox オプションのサブオプション

オプション	説明
boxheight	（要素 2 個のリスト。各要素は正の float か、文字サイズに対するパーセント値か、キーワード。テキスト行・テキストフローでのみ可）テキスト枠の縦の長さを定義します。ベースラインの上と下の長さについて、値 2 個を数値かキーワードで指定することができます： none（長さゼロ）・xheight・descender・capheight・ascender・fontsize・leading・textrise テキストフローの場合、範囲枠の先頭のテキストに対応する値が使われます。 デフォルト：{capheight none}
boxwidth	（float またはパーセント値。テキストフローでのみ可）範囲枠の幅を、ユーザー座標で、またははめ込み枠の幅に対するパーセント値で指定します。このオプションを与えると、matchbox オプションと、次の部分テキストか matchbox end 指定との間に、指定した幅の横アキが挿入されます。これは、テキストフローの中に、画像・テンプレート・PDF ページを挿入するためのアキを確保したいときに便利です。なお、alignment=justify の場合には、枠の幅がテキストと同様に圧縮されることがあります（オプション shrinklimit を参照）。デフォルト：0
clipping	（長方形、またはパーセント値 4 個。画像・グラフィック・取り込み PDF ページでのみ可。innerbox オプションを指定している場合には無視されます）画像かグラフィックかページの中のどの部分が見えるようにするかを指定する長方形の左下隅と右上隅の座標。その指定はオブジェクトによって異なります（デフォルト：{0% 0% 100% 100%}）： ▸ 画像の場合、この切り抜き長方形は、ピクセル単位で、または幅・高さに対するパーセント値として指定できます。 ▸ グラフィックの場合、この切り抜き長方形は、このグラフィックの座標で、またはそのグラフィックのオブジェクト枠の幅・高さに対するパーセント値として指定できます。 ▸ PDF ページの場合、この切り抜き長方形は、デフォルト単位で、またはそのページの CropBox の幅・高さに対するパーセント値として指定できます。
create-wrapbox	（論理値。テキストフローでのみ可）true にすると、範囲枠を構成する 1 個ないし複数の長方形が算出された後に、それらが回り込み枠としてテキストフロー内へ挿入されます。その範囲枠の入っている行の後に続く行群は、その長方形を回り込みます。デフォルト：false
doubleadapt	true にすると、2 番目の線の始点と終点が、1 番目の線に合わせて調整されます。そうでなければ、2 番目の線は、doubleoffset の分だけ短く、または長くなります。デフォルト：true
doubleoffset	（float）0 以外にすると、内側の範囲枠長方形の境界のまわりの線が二重になります。2 番目の線は、元の線に対して、指定した変位を持ちます。変位を正の値にすると、線は範囲枠長方形の外側に描かれ、負の値にすると内側に描かれます。デフォルト：0（すなわち一重線）
drawleft ***drawbottom*** ***drawright*** ***drawtop***	（論理値）true にすると、borderwidth を 0 より大きい値に設定しているなら、長方形のそれぞれの境界が描かれます。デフォルト：true

表 6.4 さまざまなメソッドの matchbox オプションのサブオプション

オプション	説明
end	（論理値。テキストフローでのみ可）範囲枠の終了。true にすると、カレントの matchbox 定義に対する他のすべてのオプションは無視されます。テキストフロー内の範囲枠はネストできません。テキストフローの範囲枠の幅は、boxwidth オプション（指定していれば）と、matchbox オプションと matchbox=end オプションではさんだテキストの視覚的長さによって定義されます。この end オプションを指定していないときは、範囲枠は、テキストフローの末尾キャラクターの後に終了します。
exceedlimit	（float またはパーセント値。テキストフローでのみ可）範囲枠がはめ込み枠の下辺または右辺からはみ出すことを許す上限を、ユーザー座標で、または範囲枠の高さに対するパーセント値で指定します。指定した上限を超えるときは、PDF_fit_textflow( ) は _boxfull を返します。この場合、残りのテキストと範囲枠は次のはめ込み枠へ続くことができます。デフォルト：0、すなわち範囲枠が枠内に完全に収まる必要があります。
innerbox	（論理値。表セルと TIFF・JPEG 画像でのみ可）表セル：true にすると、セルに対して定義している余白の分だけセル枠が縮小されます。そうでなければセル枠全体が用いられます。 TIFF・JPEG 画像：true にすると、画像にクリッピングパスがあるときは、画像全体でなく、そのクリッピングパスの外接枠が使われます。 デフォルト：false
margin	（float またはパーセント値）範囲枠の長方形に対する追加の余白を、ユーザー座標系で（0 以上にする必要があります）、または長方形の幅か高さに対するパーセント値で（100% 未満にする必要があります）指定します。このオプションは、offset* を指定している辺については無視されます。 デフォルト：0
name	（名前文字列）範囲枠の名前。この名前を付けた範囲枠がすでにあるときは、この範囲枠のための長方形がもう 1 個作られます。ですので、1 個の範囲枠が複数の長方形から成る場合がありえます。この名前は PDF_info_matchbox( ) で使えます。さまざまなメソッドで、usematchbox オプションを使って、範囲枠の長方形（群）を参照することができます。たとえば PDF_create_annotation( ) で注釈を追加することができます。範囲枠の名前は、カレントページを終えるまで使えます。名前「*」（アスタリスクキャラクター）を範囲枠の名前として用いてはいけません。デフォルト：名前なし
offsetleft ***offsetbottom*** ***offsetright*** ***offsettop***	（float またはパーセント値）算出された長方形から、求める枠への、左・右・下・上辺のユーザー定義の変位。値は、ユーザー座標で、または長方形の幅（offsetleft/offsetright の場合）か高さ（offsetbottom/offsettop の場合）に対するパーセント値で指定します。負の値も可能で、範囲枠を拡げるために用いることができます。offsetleft/offsetbottom のデフォルト：margin。 offsetright/offsettop のデフォルト：-margin
openrect	（論理値。テキストフロー・表セルでのみ可）テキストフロー：true にすると、範囲枠長方形が分割される必要がある（たとえばフォント変更や改行のために）とき、前の長方形の右辺と、後の長方形の左辺を描きません。表セル：true にすると、表行が次の表インスタンスへ分割されるとき、前の部分の下辺と、後の部分の上辺を描きません。デフォルト：false
round	（float）範囲枠長方形の隣りあう線の頂点が、それらの接合点で、指定した半径を持ち、それらの線分を接線とする円弧によって丸められます。負の半径を指定すると、角が円形にへこむように弧が切り取られ、その接線は枠の線分に対して垂直になります。デフォルト：0（丸めなし）

6.2.3 範囲枠をクエリー

C++ Java C# ***double info_matchbox(String boxname, int num, String keyword)***

Perl PHP ***float info_matchbox(string boxname, int num, string keyword)***

C ***double PDF_info_matchbox(PDF *p, const char *boxname, int len, int num, const char *keyword)***

カレントページ上の範囲枠に関する情報を取得します。

boxname （名前文字列）カレントページ上でこの名前で作成された範囲枠の名前。これは、その範囲枠が定義された際にその *matchbox* オプションの *name* サブオプションで作成されている必要があります。あるいは、名前「*」（アスタリスクキャラクター）を用いて、そのページ上のすべての範囲枠に関する情報をクエリーすることもできます。空の *boxname* を用いて、カレントページ上のすべての範囲枠長方形に関する情報をクエリーすることもできます。

len （C 言語バインディングのみ）*boxname* の長さ（バイト単位）。*len=0* にすると null 終了文字列を与える必要があります。

num 範囲枠または長方形の正の番号（1 から数える）。

keyword 求める情報を表 6.5 に従って指定したキーワード。

戻り値 *keyword* で要求された何らかの範囲枠特性の値。指定された名前の範囲枠、または指定された番号の範囲枠長方形が存在しないときは、戻り値は、キーワード *boundingbox*・*name*・*rectangle* に対しては -1（PHP では 0）になり、それ以外のすべてのキーワードに対しては 0 になります。要求されたキーワードがテキストを生み出す場合には、文字列番号が返され、その対応する文字列を、*PDF_get_string()* を用いて取得する必要があります。

カレントスコープに応じて、このメソッドは、カレントページ・パターン・テンプレート・グリフ記述上の範囲枠群に関する情報を返します。

詳細 テキストフロー内の名前付き範囲枠は、*PDF_fit_textflow()* を呼び出した後にのみ取得することができます。**ブラインドモードで作成された範囲枠をクエリーすることはできません。**

キーワード *boundingbox*・*exists*・*height*・*name*・*rectangle*・*width*・*x1*・*y1*・…・*x4*・*y4* に対する長方形は、以下のように選択されます：

- *boxname* が範囲枠の名前を内容としているとき：カレントページ上のその指定名の範囲枠の *num* 番目の矩形を選択します。
- *boxname=** のとき：カレントページ上の *num* 番目の名前付き範囲枠の最初の矩形を選択します。
- *boxname* が空のとき：カレントページ上の名前付き範囲枠によって作成された *num* 番目の長方形を選択します。

スコープ **文書・オブジェクト**以外任意

表 6.5 PDF_info_matchbox( ) のキーワード

キーワード	説明
activeitemid	範囲枠がタグ付きテキストフロー内で作成されている場合には構造エレメントのアイテム ID、そうでないなら -1。

表 6.5 PDF_info_matchbox( ) のキーワード

キーワード	説明
boundingbox	選択された長方形の外接枠をカレントユーザー座標系で表したものを内容とするパスオブジェクトのハンドル、あるいは指定された長方形が存在しない場合には -1（PHP では 0）。この外接枠は、範囲枠が回転されている場合には、その長方形とは異なります。
count	（num 引数は無視されます） boxname が範囲枠の名前を内容として持つとき：この範囲枠の長方形の数 boxname=* のとき：少なくとも 1 個の長方形を持つ範囲枠の数 boxname が空のとき：名前付き範囲枠群によって作成された長方形の総数
exists	選択された長方形が存在しているなら 1、そうでないなら 0。
height[1]	選択された長方形の高さをユーザー座標で表したもの
name	選択された長方形が作成された目的である範囲枠の名前の文字列番号。対応する文字列は、PDF_get_string( ) で取得できます。
rectangle	選択された長方形をユーザー座標で表したものを内容として持つパスオブジェクトのハンドル、、あるいはその長方形が見つからなかったときには -1（PHP では 0）
width[1]	選択された長方形の幅をユーザー座標で表したもの
x1・y1・... x4・y4[1]	選択された長方形の i 番目の隅（i=1, 2, 3, 4）の位置を、ユーザー座標で表したもの。それぞれのはめ込み要素（画像・テキスト等）の座標系で、x1，y1 は左上、x2，y2 は左下、x3，y3 は右下、x4，y4 は右上隅に対応します。

1. このキーワードは、boxname=* のときは無視されます

7 グラフィックメソッド

この章の API メソッド：

- *PDF_set_graphics_option()*
- *PDF_setlinewidth()*
- *PDF_save()*
- *PDF_restore()*
- *PDF_create_gstate()*
- *PDF_set_gstate()*
- *PDF_translate()*
- *PDF_scale()*
- *PDF_rotate()*
- *PDF_align()*
- *PDF_skew()*
- *PDF_concat()*
- *PDF_setmatrix()*
- *PDF_moveto()*
- *PDF_lineto()*
- *PDF_curveto()*
- *PDF_circle()*
- *PDF_arc()*
- *PDF_arcn()*
- *PDF_circular_arc()*
- *PDF_ellipse()*
- *PDF_elliptical_arc()*
- *PDF_rect()*
- *PDF_closepath()*
- *PDF_stroke()*
- *PDF_closepath_stroke()*
- *PDF_fill()*
- *PDF_fill_stroke()*
- *PDF_closepath_fill_stroke()*
- *PDF_clip()*
- *PDF_endpath()*
- *PDF_add_path_point()*
- *PDF_draw_path()*
- *PDF_info_path()*
- *PDF_delete_path()*

7.1 グラフィック書式オプション

グラフィック書式オプション　表 7.1 のグラフィック書式オプションは、以下のメソッドで使えます（なお、すべてのメソッドがすべてのオプションに対応しているわけではありません。詳しくは関数の説明を参照してください）：

- *PDF_set_graphics_option()*
- *PDF_create_gstate()*（*dasharray*・*dashphase*・*flatness*・*linecap*・*linejoin*・*linewidth*・*miterlimit* のみ）
- *PDF_add_path_point()*・*PDF_draw_path()*
- *PDF_fit_table()* の塗りオプション（*fillcolor*・*shading* のみ）と、*PDF_fit_table()* の描線オプション（*dasharray*・*dashphase*・*linecap*・*linejoin*・*linewidth*・*strokecolor* のみ）
- *PDF_shading_pattern()*
- さまざまなメソッドの *matchbox* オプション

表 7.1 グラフィック書式オプション

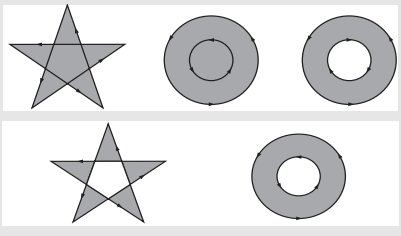
オプション	説明
cliprule	（キーワード）切り抜きのための領域の内面を決定する切り抜き規則。とりうるキーワードについては fillrule を参照してください。デフォルト：fillrule オプションの値
borderwidth	（float。範囲枠でのみ可）長方形の境界の線幅。borderwidth を 0 より大きい値に設定すると、すべての長方形の境界が描線されます。上・下・左・右の境界が描線されないようにするには、それぞれ drawtop・drawbottom・drawleft・drawright オプションを false に設定します。デフォルト：0
dasharray	（2 個の非負 float のリストかキーワード）描線されるパスの短線と間隙の長さ（ユーザー座標系で測ったもの）を交互に指定した 2 ～ 12 個の値のリスト。これらの配列値は負であってはいけません。これらは、パス全体が描線されるまで循環的に再利用されます。キーワード none を用いると、実線を作成することができます。デフォルト：none
dashphase	（float）破線パターン内で短線を開始するまでの距離。デフォルト：0
fillcolor[1]	（色）領域の塗り色。デフォルト：通常は {gray 0}（PDF/A モードの場合：{lab 0 0 0}）、ただし範囲枠の場合は none
fillrule	（キーワード）領域の、塗りと切り抜きの際の内側を決定する塗り規則（デフォルト：winding）： ***winding*** 非ゼロ巻数規則を用います。単純な形状の場合、塗りの結果は直感的見込みと一致します。複数のパスから成る形状の場合には、パスの方向が意味を持ちます。 ***evenodd*** 偶奇規則を用います。これは、単純な形状では winding と同じ結果を得ますが、より複雑な形状、とりわけ自己交差するパスでは異なる結果を生じます。 
flatness[2]	（float > 0）円弧または曲線と、線分群から成る近似表現との最大間隔を示した（デバイスピクセル単位で）正の数。デフォルト：1
gstate	（PDF_create_gstate( ) に対するオプションのオプションリスト、またはグラフィックステートハンドル）グラフィックステートオプション群またはハンドル。デフォルト：グラフィックステートなし、すなわち、カレント設定が用いられます
initgraphics-state[1]	（論理値。PDF_set_graphics_option( ) でのみ可）true にすると、すべてのグラフィック書式オプションがそのデフォルト値で初期化されます。カレントクリッピングパスは影響を受けません。false にすると、カレントグラフィックステート値群が用いられます。デフォルト：false

表 7.1 グラフィック書式オプション

オプション	説明
linecap	（整数またはキーワード）パスの終端の形状（デフォルト：*PDF_fit_table()* では projecting、それ以外では butt）： ***butt*** （同等な値：0）バット線端：パスの終端で描線を四角く切ります。 ***round*** （同等な値：1）丸型線端：終端を中心とし、線幅に等しい直径を持つ半円を描き、塗ります。 ***projecting*** （同等な値：2）突出線端：線の終端から、線幅の半分だけ描線を延長し、四角く切ります。
linejoin	（整数またはキーワード）パスの角の形状（デフォルト：miter）： ***miter*** （同等な値：0）マイター結合：2 本の線分の描線の外辺を、互いに出会うまで延ばします。そうするとマイターリミットによる指定よりも遠くまで延びてしまうときは、ベベル結合がかわりに用いられます。 ***round*** （同等な値：1）ラウンド結合：線分どうしの交点を中心とし、線幅に等しい半径の円弧を描き、塗ることで、角を丸くします。 ***bevel*** （同等な値：2）ベベル結合：2 本のパス線分をバット線端で描き（linecap の説明を参照）、終端どうしの間に生じる三角形の切れ込みを塗ります。
linewidth	（float > 0）線幅。デフォルト：1
miterlimit	（float ≧ 1）マイター結合によって形成されるくさび型を制御します（デフォルト：10。これは角度にしておよそ 11.5 度にあたります）。線結合スタイル linejoin を 0（マイター結合）に設定すると、2 本の線分が出会う角度が小さいときは、鋭いくさび型が形作られます。マイター長と線幅との比がマイターリミットを超えるとき、このくさび型を平坦な終端に置き換えます（すなわちマイター結合をベベル結合に変更します）。
shading	（表 8.4 に従ったオプションリスト。範囲枠・表でのみ可）範囲枠の長方形（群）または表領域のシェーディングを指定します。以下のオプションを使用可能です：antialias・domain・end・endcolor・N・start・startcolor・type
strokecolor[1]	（色）パスの描線色。デフォルト：通常は {gray 0}（PDF/A モードの場合：{lab 0 0 0}）、ただし範囲枠の場合は none

1. グリフスコープ内では、もしそのグリフの colorized オプションが false の場合にはこのオプションを使ってはいけません。
2. このデバイス依存なグラフィックステートオプションをパターンスコープ内で使ってはいけません。

C++ Java C# ***void set_graphics_option(String optlist)***

Perl PHP ***set_graphics_option(string optlist)***

C ***void PDF_set_graphics_option(PDF *p, const char *optlist)***

1 個ないし複数のグラフィック書式オプションを設定します。

optlist 表 7.1 に従ってグラフィック書式オプションを指定したオプションリスト。以下のオプションを使えます：

cliprule・dasharray・dashphase・fillcolor・fillrule・flatness・gstate・initgraphicsstate・linecap・linejoin・linewidth・miterlimit・strokecolor

詳細　グラフィック書式オプション群は、以下のメソッドのためのグラフィックステートを設定します：

- 明示的描画メソッド：*PDF_stroke()*・*PDF_fill()* など
- 暗黙的描画メソッド：*PDF_fit_textline()*・*PDF_fit_textflow()* の *showborder* オプションなど
- テキストオプション群を用いて色が設定されていない場合に、単純テキスト出力メソッド群を用いて作成されるテキスト出力：*PDF_show()* など

すべてのグラフィック書式オプションは、ページ・パターン・テンプレート・グリフ定義の開始でそれらのデフォルト値へリセットされ、カレント**ページ・ページ・テンプレート・グリフ**スコープの終了までそれらの値を保持します。ただし、このグラフィック書式オプション群を *initgraphicsstate* オプションを用いてリセットすることもできます。

以後の *PDF_setcolor()* への呼び出しは、*fillcolor* および / または *strokecolor* 値をオーバーライドします。以後の *PDF_setlinewidth()* への呼び出しは *linewidth* 値をオーバーライドします。

スコープ　**ページ・パターン・テンプレート・グリフ**（オプション *fillcolor*・*strokecolor*・*initgraphicsstate* は**グリフ**スコープではグリフの *colorized* オプションが *true* の場合にのみ使えます）。バイス依存なオプションは**パターン**スコープでは使ってはいけません。

7.2 グラフィックステート

C++ Java C# ***void setlinewidth(double width)***
Perl PHP ***setlinewidth(float width)***
C ***void PDF_setlinewidth(PDF *p, double width)***

カレント線幅を設定します。

width 線幅を、ユーザー座標系の単位で指定します。0 より大きくなければいけません。

詳細 このメソッドは、グラフィック状態の線幅（*PDF_set_graphics_option()* 参照）とテキスト状態の描線幅（*PDF_set_text_option()* 参照）を設定します。*width* 引数は各ページの開始時にデフォルト値 1 にリセットされます。

スコープ ページ・パターン・テンプレート・グリフ

C++ Java C# ***void save()***
Perl PHP ***save()***
C ***void PDF_save(PDF *p)***

カレントグラフィックステートをスタックに保存します。

詳細 グラフィックステートは、あらゆる種類のグラフィックオブジェクトを制御するオプション群を持っています。グラフィックステートの保存は、PDF によって必要とされているわけではなく、アプリケーションが後で、オプションをすべて明示的に設定しなおすことなしにいずれかのグラフィックステート（カスタムの座標系等）へ戻りたいときにのみ必要です。以下の項目が保存・復帰の対象になります。

- グラフィック書式オプション：
  クリッピングパス・座標系・カレント点・平坦度許容量・線端スタイル・破線パターン・線結合スタイル・線幅・マイターリミット。
- 色オプション：塗り・描線色。
- *PDF_set_gstate()* で明示グラフィックステートによって設定しているグラフィックオプション。
- テキスト位置と、以下のテキスト書式オプション：
  charspacing・decorationabove・fakebold・font・fontsize・horizscaling・italicangle・leading・strokewidth・textrendering・textrise・underlineposition・underlinewidth・wordspacing

PDF_save() と *PDF_restore()* のペアは、ネストさせることもできます。PDF 規格では、保存・復帰のペアのネスト階層の深さに制限はありませんが、アプリケーションではネスト階層の深さを 26 未満に抑えておくべきです。

多くのテキストオプションは、保存 / 復帰によって影響を受けます。上記の一覧を参照してください。以下のテキストオプションは保存 / 復帰に影響を受けません：*fillrule・kerning・underline・overline・strikeout*。

スコープ **ページ・パターン・テンプレート・グリフ**。対応する *PDF_restore()* と必ずペアにして呼び出す必要があります。*PDF_save()* と *PDF_restore()* への呼び出しは、ページ・パターン・テンプレート・グリフ記述ごとに同数にする必要があります。

C++ Java C# **void restore( )**

Perl PHP **restore( )**

C **void PDF_restore(PDF *p)**

もっとも最近にスタックに保存したグラフィックステートに復帰します。

詳細 そのグラフィックステートは、同じページかパターンかテンプレートで保存してある必要があります。

スコープ **ページ・パターン・テンプレート・グリフ**。対応する *PDF_save( )* と必ずペアにして呼び出す必要があります。*PDF_save( )* と *PDF_restore( )* への呼び出しは、ページ・パターン・テンプレート・グリフ記述ごとに同数にする必要があります。

C++ Java C# **int create_gstate(String optlist)**

Perl PHP **int create_gstate(string optlist)**

C **int PDF_create_gstate(PDF *p, const char *optlist)**

グラフィックステートオブジェクトを、さまざまなオプションに従って作成します。

optlist グラフィックステートオプションを指定したオプションリスト：

- 表 7.1 に従ったグラフィック書式オプション：
  dasharray・*dashphase*・*linecap*・*linejoin*・*linewidth*・*miterlimit*
- 表 7.2 に従ったグラフィックステートオプション：
  alphaisshape・*blendmode*・*halftoneorigin*・*opacityfill*・*opacitystroke*・*renderingintent*・*softmask*・*strokeadjust*・*textknockout*・*useblackptcomp*
- 表 7.3 に従った、デバイス依存なグラフィックステートオプション：
  overprintfill・*overprintmode*・*overprintstroke*・*smoothness*
- 表 7.1 に従った、デバイス依存なグラフィック書式オプション：*flatness*

戻り値 グラフィックステートハンドル。以後の *PDF_set_gstate( )* への呼び出しで、カレントの**文書**スコープが続く限り使えます。

詳細 オプションリストは、任意の数のグラフィックステートオプションを内容として持つことができます。指定されなかったオプションは、以前の値を保ちます。この以前の値というのは、デフォルト値か、以前に設定されていた他のグラフィックステートの中で指定された値です。

スコープ **オブジェクト**以外任意。表 7.3 のデバイス依存なグラフィックステートオプションは**パターン**スコープでは使ってはいけません。

表 7.2 PDF_create_gstate( ) のオプション

オプション	説明
alphaisshape	（論理値）カレントのソフトマスク・アルファ定数の供給元の扱いを、輪郭（true）か不透過（false）にします。デフォルト：false

表 7.2 PDF_create_gstate( ) のオプション

オプション	説明
blendmode[1]	（キーワード。PDF/A-1・PDF/X-3 では Normal のみ可）透過操作に対するブレンドモード（デフォルト：None）： Color・ColorDodge・ColorBurn・Darken・Difference・Exclusion・HardLight・Hue・Lighten・Luminosity・Multiply・None・Normal・Overlay・Saturation・Screen・SoftLight。 これらのブレンドモードの説明とその使用例については PDFlib チュートリアルを参照してください。
halftoneorigin	（2 個の float のリスト。PDF 2.0）ハーフトーン原点の座標をユーザー座標で
opacityfill[1]	（float かパーセント値。PDF/A-1・PDF/X-3 では値 1 にする必要があります）塗り操作での不透明度を、範囲 0 ～ 1 で指定します。値 0 は完全に透過を意味します。値 1 は完全に不透明を意味します。
opacitystroke[1]	（float かパーセント値。PDF/A-1・PDF/X-3 では値 1 にする必要があります）描線操作での不透明度を、範囲 0 ～ 1 で指定します。値 0 は完全に透過を意味します。値 1 は完全に不透明を意味します。
renderingintent	（キーワード）CIE ベースカラー変換のためのレンダリングインテント： Auto・AbsoluteColorimetric・RelativeColorimetric・Saturation・Perceptual

表 7.2 PDF_create_gstate( ) のオプション

オプション	説明
softmask[1]	（オプションリストまたはキーワード。PDF/A・PDF/X-3 では none のみ使用可能）透過画像処理のためのカレントのソフトマスク。グラフィックステートのソフトマスクは画像のソフトマスクによってオーバーライドされる場合があります。なぜなら PDF は同時に 2 つのソフトマスクには対応していないからです（回避策：画像オプション templateoptions={transparencygroup={isolated}} を使用）。使えるオプションとキーワード（デフォルト：none）： ***backdropcolor*** （キーワード、または float 1 個か 3 個か 4 個のリスト。type=luminosity の場合のみ可）ソフトマスクの初期背景（デフォルト：opaque）： ***opaque*** ソフトマスクが黒で初期化されることによってソフトマスクを不透明にします。すなわちテンプレート内で明るい（白い）内容が描かれていない領域ですべての色がブロックされます。 ***transparent*** ソフトマスクが白で初期化されることによってソフトマスクを透明にします。すなわちテンプレート内で暗い（黒い）内容が描かれていない領域でカレント色が使われます。 キーワードのかわりにカラー値群を与えることもできます。色要素の数は、ソフトマスクテンプレートを作成する際に用いた transparencygroup オプションの colorspace サブオプションに合致している必要があります（たとえば DeviceRGB であれば 3）。 ***invert*** （論理値）true にするとソフトマスクの効果が反転します。すなわち、テンプレートの暗い領域が透明になり、明るい領域が色をブロックします。このプションは、backdropcolor サブオプションに対するキーワードの効果をも反転させます。デフォルト：false ***none*** （キーワード）ソフトマスクなし。これは、直前に設定されたグラフィックステートから効力を持っているかもしれないソフトマスクを無効にするために必要です。 ***template*** （テンプレートハンドル。必須）PDF_begin_template_ext( ) と transparencygroup オプションを用いて作成された透過グループテンプレート。type=luminosity のときはこのテンプレートは colorspace サブオプションで none 以外の値を用いて作成されている必要があります。 ***type*** （キーワード。必須）透過グループテンプレートからマスク値を導出するための方式： ***alpha*** テンプレートのアルファ値群がマスクを定義します。色は無視されます。 ***luminosity*** テンプレートの色は輝度値（知覚されるグレーレベル）へ変換されてそれがマスクを定義します。
strokeadjust	（論理値）自動描線調整を適用するかどうか。デフォルト：false
textknockout	（論理値）色版合成に関して、テキスト内のグリフ群を、ばらばらに扱う（false）か、単体として扱う（true）かを指定します。デフォルト：true
useblackptcomp	（キーワード。PDF 2.0）CIE ベースカラー変換に対する黒点補償を制御します。使えるキーワード：off・on・default。デフォルト：default

1. この透過オプションを用いて作成したグラフィックステートは、グリフスコープでは、グリフの colorized オプションが false の場合には使ってはいけません。

表 7.3 PDF_create_gstate( ) のデバイス依存なオプション（パターンスコープでは使ってはいけません）

オプション	説明
overprintfill	（論理値）画像置換を含め、あらゆる非描線操作に対するオーバープリント動作：false にすると、任意の色空間において描画を行う際に、指定されていないインキの該当領域が消去されます。true にすると、出力デバイスがオーバープリンティングに対応している場合には、他のインキにおけるそれ以前のマーキングが変更されずに残されます。デフォルト：false
overprintmode	（整数）overprintfill か overprintstroke が true の場合の CMYK 要素値 0 のオーバープリント動作。このオーバープリントモードは、テキストとベクトル要素にのみ影響を与え、画像とシェーディングには影響しません。使用可能な値（デフォルト：0）： 0　（ゼロ）各色要素は、それ以前に配置されたマーク群を置換します（「前面色優先」）。 1　0 の色要素は、それ以前に描かれた色の対応する要素を変更せずに残します（「前面濃度値 0 を無視」）。言い換えれば、値 0 は無指定として扱われます。 PDF/A-2/3：もしもカレント色空間が ICC ベース CMYK であり、かつ overprintfill か overprintstroke が true の場合には、overprintmode=1 は指定不可です。
overprintstroke	（論理値）描線操作でのオーバープリント：false にすると、任意の色空間において描画を行う際に、指定されていないインキの該当領域が消去されます。true にすると、かつ出力デバイスがオーバープリンティングに対応している場合には、他のインキにおけるそれ以前のマーキングが変更されずに残されます。デフォルト：false
smoothness	（範囲 0 ～ 1 の float）シェーディングにおける色内挿の最大誤差

C++ Java C# ***void set_gstate(int gstate)***

Perl PHP ***set_gstate(int gstate)***

C ***void PDF_set_gstate(PDF *p, int gstate)***

グラフィックステートオブジェクトを呼び出します。

gstate *PDF_create_gstate( )* で取得したグラフィックステートオブジェクトのハンドル。

詳細 グラフィックステートオブジェクトが持つすべてのオプションが設定されます。このメソッドを複数回呼び出すと、グラフィックステートのオプションは蓄積されていきます。指定されなかったオプションは、以前の値を保ちます。この以前の値というのは、デフォルト値か、以前に設定されていた他のグラフィックステートの中で指定された値です。グラフィックステートのオプションはすべて、ページの始まりごとに、それぞれのデフォルト値にリセットされます。

スコープ **ページ・パターン・テンプレート・グリフ**。表 7.3 のデバイス依存なグラフィックステートオプションは**パターン**スコープでは使ってはいけません。

7.3 座標系の変換

変換メソッド（*PDF_translate()*・*PDF_scale()*・*PDF_rotate()*・*PDF_align()*・*PDF_skew()*・*PDF_concat()*・*PDF_setmatrix()*・*PDF_initgraphics()*、および、*PDF_set_graphics_option()* の *initgraphicsstate* オプション）はすべて、以後のオブジェクトを描くために使われる座標系を変更します。ページにすでに存在しているオブジェクトでは効力を持ちません。

C++ Java C# ***void translate(double tx, double ty)***
Perl PHP ***translate(float tx, float ty)***
C ***void PDF_translate(PDF *p, double tx, double ty)***

座標系の原点を平行移動させます。

tx・ty 座標系の新しい原点 *(tx, ty)* を、古い座標系で測ります。

スコープ ページ・パターン・テンプレート・グリフ

C++ Java C# ***void scale(double sx, double sy)***
Perl PHP ***scale(float sx, float sy)***
C ***void PDF_scale(PDF *p, double sx, double sy)***

座標系を拡縮します。

sx・sy *x*・*y* 方向の拡縮倍率。

詳細 このメソッドは、座標系を *sx* 倍・*sy* 倍します。負の倍率を使って、反転（鏡映）を行わせることもできます。新しい座標系における *x* 方向の 1 単位は、古い座標系における *x* 方向の *sx* 単位と等しくなります。*y* 座標についても同様です。

スコープ ページ・パターン・テンプレート・グリフ

C++ Java C# ***void rotate(double phi)***
Perl PHP ***rotate(float phi)***
C ***void PDF_rotate(PDF *p, double phi)***

座標系を回転させます。

phi 回転角を度単位で指定します。

詳細 角度は、カレント座標系の *x* 軸正の向きからの反時計回りで測ります。新しい座標軸は、古い座標軸を *phi* 度回転させることによって得られます。

スコープ ページ・パターン・テンプレート・グリフ

C++ Java C# ***void align(double dx, double dy)***

Perl PHP ***align(float dx, float dy)***

C ***void PDF_align(PDF *p, double dx, double dy)***

座標系の方向を相対ベクトルと同じにします。

dx・dy 方向ベクトルの座標。*dx* と *dy* を両方 0 にしてはいけません。

詳細 座標系を回転し、新しい座標系の *x* 軸の方向がベクトル *(dx, dy)* と同じになり、*y* 軸の方向が *(-dy, dx)* と同じになるようにします。これは *PDF_rotate( )* で *phi=180°/ pifkatan2(dy/dx)* とした場合と等価です。

スコープ ページ・パターン・テンプレート・グリフ

C++ Java C# ***void skew(double alpha, double beta)***

Perl PHP ***skew(float alpha, float beta)***

C ***void PDF_skew(PDF *p, double alpha, double beta)***

座標系を斜形化します。

alpha・beta *x・y* 方向の斜形化角を、度単位で指定します。

詳細 斜形化（シアーともいう）とは、座標系を *x・y* 方向へ指定の角度だけ歪めます。*alpha* は、カレント座標系の *x* 軸正の向きからの反時計回りで表され、*beta* は、*y* 軸正の向きからの時計回りで測ります。どちらの角度も、90° の奇数倍にしてはいけません。

スコープ ページ・パターン・テンプレート・グリフ

C++ Java C# ***void concat(double a, double b, double c, double d, double e, double f)***

Perl PHP ***concat(float a, float b, float c, float d, float e, float f)***

C ***void PDF_concat(PDF *p, double a, double b, double c, double d, double e, double f)***

カレント座標系に変換行列を適用します。

a・b・c・d・e・f 変換行列の各要素。6 個の値は、PDF と同じ方式で行列を構成します（各リファレンスを参照）。縮退変換を避けるため、*a*×*d* を *b*×*c* に等しくしてはいけません。

詳細 このメソッドを使うと、もっとも一般化された形で変換を行えます。変換行列の使い方がよくわからないなら、このメソッドでなく、*PDF_translate( )*・*PDF_scale( )*・*PDF_rotate( )*・*PDF_skew( )* の使用を推奨します。座標系は、ページの始まりごとにデフォルト座標系（すなわち、カレント変換行列が単位行列 *[1, 0, 0, 1, 0, 0]*）にリセットされます。

スコープ ページ・パターン・テンプレート・グリフ

C++ Java C# ***void setmatrix(double a, double b, double c, double d, double e, double f)***

Perl PHP ***setmatrix(float a, float b, float c, float d, float e, float f)***

C ***void PDF_setmatrix(PDF *p, double a, double b, double c, double d, double e, double f)***

カレント変換行列を明示的に設定します。

a・b・c・d・e・f *PDF_concat( )* 参照。

詳細 このメソッドは *PDF_concat( )* に似ています。ただし、カレント変換行列を捨てて、新しい行列に置き換えます。

スコープ **ページ・パターン・テンプレート・グリフ**

7.4 パス構築

注 この節のメソッドを使った後は、必ず 165 ページ「7.5 描画とクリッピング」のメソッドのいずれかを呼び出してください。そうでないと、作成したパスは何の効果もなく、その後の操作が例外を発生させることがあります。

PDF/UA ベクトルグラフィックは、*PDF_begin_item( )* への呼び出しを用いて *Artifact* か *Figure* としてタグ付けする必要があります。

C++ Java C# ***void moveto(double x, double y)***

Perl PHP ***moveto(float x, float y)***

C ***void PDF_moveto(PDF *p, double x, double y)***

グラフィック出力のためのカレント点を設定します。

x・y 新しいカレント点の座標。

詳細 カレント点は、ページの始まりごとに、デフォルト値である**未定義**に設定されます。グラフィックのカレント点と、カレントテキスト位置は、別々に保持されます。

スコープ **ページ・パターン・テンプレート・グリフ・パス**。このメソッドは**パス**スコープを開始させます。

C++ Java C# ***void lineto(double x, double y)***

Perl PHP ***lineto(float x, float y)***

C ***void PDF_lineto(PDF *p, double x, double y)***

カレント点から別の点へ線を描きます。

x・y 線の終点の座標。

詳細 このメソッドは、カレント点から *(x, y)* までの直線を、カレントパスに追加します。カレント点は、このメソッドを使う前に設定しておく必要があります。点 *(x, y)* が新たにカレント点になります。

この線は、「理想の線」を中心軸として描かれます。すなわち、2 個の端点を結ぶ線の両側に、線幅（*linewidth* オプションの値によって決定される）の半分ずつが描かれます。終端での動作は、*linecap* オプションによって決定されます。

スコープ **パス**

C++ Java C# ***void curveto(double x1, double y1, double x2, double y2, double x3, double y3)***

Perl PHP ***curveto(float x1, float y1, float x2, float y2, float x3, float y3)***

C ***void PDF_curveto(PDF *p, double x1, double y1, double x2, double y2, double x3, double y3)***

カレント点から、3 個の制御点を用いて、ベジエ曲線を描きます。

x1・y1・x2・y2・x3・y3 3 個の制御点の座標。

詳細 カレント点から *(x3, y3)* までのベジエ曲線を、*(x1, y1)* と *(x2, y2)* を制御点として使って、カレントパスに追加します。カレント点は、このメソッドを使う前に設定しておく必要があります。曲線の終点が新たにカレント点になります。

スコープ　パス

C++ Java C# ***void circle(double x, double y, double r)***

Perl PHP ***circle(float x, float y, float r)***

C ***void PDF_circle(PDF *p, double x, double y, double r)***

円を描きます。

x・y 円の中心の座標。

r 円の半径。

詳細 このメソッドは、円を完全なサブパスとして、カレントパスに追加します。点 ***(x + r, y)*** が新たにカレント点になります。描かれる形は、ユーザー座標系において円になります。座標系を、***x・y*** 方向で違う倍率で拡縮しているときは、描かれる曲線は楕円になります。この円は反時計回りの向きに作成されます。

スコープ **ページ・パターン・テンプレート・グリフ・パス**。このメソッドは**パス**スコープを開始させます。

C++ Java C# ***void arc(double x, double y, double r, double alpha, double beta)***

Perl PHP ***arc(float x, float y, float r, float alpha, float beta)***

C ***void PDF_arc(PDF *p, double x, double y, double r, double alpha, double beta)***

円弧を、反時計周りで描きます。

x・y 円弧の中心の座標。

r 円弧の半径。*r* は非負にする必要があります。

alpha・beta 円弧の開始角と終了角を、度単位で指定します。

詳細 このメソッドは、*alpha* 度から *beta* 度までの反時計回りの円弧を、カレントパスに追加します。*PDF_arc( )* でも *PDF_arcn( )* でも、角度は、カレントユーザー座標系における *x* 軸正の向きからの反時計回りで指定します。カレント点があるときは、カレント点から円弧の始点までの直線も描かれます。円弧の終点が新たにカレント点になります。

円弧は、ユーザー座標系において部分円になります。座標系を、*x・y* 方向で違う倍率で拡縮しているときは、描かれる曲線は部分楕円になります。

スコープ **ページ・パターン・テンプレート・グリフ・パス**。このメソッドは**パス**スコープを開始させます。

C++ Java C# ***void arcn(double x, double y, double r, double alpha, double beta)***

Perl PHP ***arcn(float x, float y, float r, float alpha, float beta)***

C ***void PDF_arcn(PDF *p, double x, double y, double r, double alpha, double beta)***

円弧を、時計回りで描きます。

詳細 描画方向以外は、このメソッドは *PDF_arc( )* とまったく同じ動作をします。特に角度は、この関数でも *x* 軸正の向きからの**反時計回り**で指定します。

C++ Java C# ***void circular_arc(double x1, double y1, double x2, double y2)***

Perl PHP ***circular_arc(float x1, float y1, float x2, float y2)***

C ***void PDF_circular_arc(PDF *p, double x1, double y1, double x2, double y2)***

3 個の点によって定義される円弧線分を描きます。

x1・y1 円弧線分上の任意の点の座標。

x2・y2 円弧線分の終点の座標。

詳細 このメソッドは、カレントパスに円弧線分を追加します。この円弧線分はカレント点を始点とし、*(x1, y1)* を通って、*(x2, y2)* を終点とします。このメソッドを使う前にカレント点を設定する必要があります。この曲線の終点が新たにカレント点となります。

この円弧線分はユーザー座標系において円形になります。座標系が *x* 方向と *y* 方向で異なる拡縮をされているときは、生成される曲線は楕円形になります。

スコープ **パス**

C++ Java C# ***void ellipse(double x, double y, double rx, double ry)***

Perl PHP ***ellipse(float x, float y, double rx, double ry)***

C ***void PDF_ellipse(PDF *p, double x, double y, double rx, double ry)***

楕円を描きます。

x・y 楕円の中心の座標。

rx・ry 楕円の *x* 半径と *y* 半径。

詳細 このメソッドは、カレントパスに楕円を、完全なサブパスとして追加します。点 *(x + rx, y)* が新たにカレント点となります。この楕円は反時計回りの向きに作成されます。

スコープ **ページ・パターン・テンプレート・グリフ・パス**。このメソッドは**パス**スコープを開始させます。

C++ Java C# ***void elliptical_arc(double x, double y, double rx, double ry, String optlist)***

Perl PHP ***elliptical_arc(float x, float y, double rx, double ry, string optlist)***

C ***void PDF_elliptical_arc(PDF *p, double x, double y, double rx, double ry, const char *optlist)***

カレント点から楕円弧を描きます。

x・y 楕円弧の終点の座標。

rx・ry 楕円の *x・y* 半径。これらの値のうち少なくとも 1 つを、カレント点と *(x, y)* との間の距離の半分よりも大きくする必要があります。

optlist 表 7.4 に従って楕円弧のための構築オプションを指定したオプションリスト。

詳細 このメソッドは、カレントパスに楕円弧を追加します。この楕円弧は、カレント点から開始して、*(x, y)* で終わります。カレント点は、このメソッドを使う前に設定されている必要があります。楕円弧の終点はが新たなカレント点となります。4 個の可能な楕円弧のうち、2 個は ≤ 180° の楕円弧（小さい楕円弧）を表し、もう 2 個は ≥ 180° の楕円弧（大きい楕円弧）を表します。

スコープ　**ページ・パターン・テンプレート・グリフ・パス**。このメソッドは**パス**スコープを開始させます。

表 7.4 PDF_elliptical_arc( ) に対するオプション

オプション	説明
clockwise	（論理値）true にすると、時計回り楕円弧のうちの 1 つが作成されます。そうでない場合は、反時計回り楕円弧が作成されます。デフォルト：false
largearc	（論理値）true にすると、大きい楕円弧のうちの 1 つが作成されます。そうでない場合は、小さい楕円弧が作成されます。デフォルト：false
rectify	（論理値）true にすると、小さすぎる半径は、楕円が構築できるよう変更されます。そうでない場合は、例外が発生します。デフォルト：false
xrotate	（float）楕円に対する回転角、すなわち、楕円の x 軸の、カレント座標系の x 軸に対する角度を、度単位で表したもの。楕円弧の開始点と終了点は固定されたままです。デフォルト：0

C++ Java C# ***void rect(double x, double y, double width, double height)***

Perl PHP ***rect(float x, float y, float width, float height)***

C ***void PDF_rect(PDF *p, double x, double y, double width, double height)***

長方形を描きます。

x・y　長方形の左下隅の座標。

width・height　長方形の寸法。

詳細　このメソッドは、長方形を完全なサブパスとして、カレントパスに追加します。このメソッドの前にカレント点を設定しておくことは、必須ではありません。点 ***(x, y)*** が新たにカレント点になります。線は、「理想の線」を中心軸として描かれます。すなわち、それぞれの端点を結ぶ線の両側に、線幅（***linewidth*** オプションの値によって決定される）の半分ずつが描かれます。この長方形は反時計回りの向きに作成されます。このメソッドは暗黙的にパスを閉じます。

スコープ　**ページ・パターン・テンプレート・グリフ・パス**。このメソッドは**パス**スコープを開始させます。

C++ Java C# ***void closepath( )***

Perl PHP ***closepath( )***

C ***void PDF_closepath(PDF *p)***

カレントパスを閉じます。

詳細　このメソッドは、カレントサブパスを閉じます。すなわち、カレント点からサブパスの開始点までの線を追加します。

スコープ　**パス**

7.5 描画とクリッピング

注 この節のメソッドの多くは、パスをクリアして、カレント点を未定義にします。ですので、これらのメソッドのいずれかを呼び出した後の描画操作では、カレント点を明示的に設定する必要があります（*PDF_moveto( )* を使う等）。

C++ Java C# ***void stroke( )***
Perl PHP ***stroke( )***
C ***void PDF_stroke(PDF *p)***

パスを、カレント線幅とカレント描線色で描線した後、クリアします。

スコープ **パス**。このメソッドは**パス**スコープを終了させます。

C++ Java C# ***void closepath_stroke( )***
Perl PHP ***closepath_stroke( )***
C ***void PDF_closepath_stroke(PDF *p)***

パスを閉じた後、描線します。

詳細 このメソッドは、カレントサブパスを閉じた後（カレント点からパスの開始点への線分を追加）、カレントパス全体を、カレント線幅とカレント描線色で描線します。

スコープ **パス**。このメソッドは**パス**スコープを終了させます。

C++ Java C# ***void fill( )***
Perl PHP ***fill( )***
C ***void PDF_fill(PDF *p)***

パスの内部を、カレント塗り色で塗ります。

詳細 このメソッドは、カレントパスの内部を、カレント塗り色で塗ります。パスの内部は、2種のアルゴリズムのいずれかによって決定されます（***fillrule*** オプション参照）。開いているパスは、塗る前に暗黙的に閉じられます。

スコープ **パス**。このメソッドは**パス**スコープを終了させます。

C++ Java C# ***void fill_stroke( )***
Perl PHP ***fill_stroke( )***
C ***void PDF_fill_stroke(PDF *p)***

パスを、カレント塗り色とカレント描線色で、塗り、描線します。

スコープ **パス**。このメソッドは**パス**スコープを終了させます。

C++ Java C# ***void closepath_fill_stroke( )***

Perl PHP ***closepath_fill_stroke( )***

C ***void PDF_closepath_fill_stroke(PDF *p)***

パスを閉じた後、塗り、描線します。

詳細 このメソッドは、カレントサブパスを閉じた後（カレント点からパスの開始点への線分を追加）、カレントパス全体を塗り、描線します。

スコープ **パス。**このメソッドは**パス**スコープを終了させます。

C++ Java C# ***void clip( )***

Perl PHP ***clip( )***

C ***void PDF_clip(PDF *p)***

カレントパスをクリッピングパスとして使って、パスを終了させます。

詳細 このメソッドは、カレントパスとカレントクリッピングパスの共通部分を、以後の操作に対するクリッピングパスとして使います。クリッピングパスは、ページの始まりごとに、デフォルト値であるページサイズと同じに設定されます。クリッピングパスは、*PDF_save( )*・*PDF_restore( )* の対象になります。クリッピングパスを大きくできるのは、*PDF_save( )*・*PDF_restore( )* を使ったときだけです。クリッピング領域は、*cliprule* オプションを用いて選択されたアルゴリズムに従って決定されます。

スコープ **パス。**このメソッドは**パス**スコープを終了させます。

C++ Java C# ***void endpath( )***

Perl PHP ***endpath( )***

C ***void PDF_endpath(PDF *p)***

カレントパスを、塗らずに、描線せずに、終了させます。

詳細 このメソッドは、ページに対して何の視覚的効力も持ちません。目に見えないパスをページに生成するだけです。

スコープ **パス。**このメソッドは**パス**スコープを終了させます。

7.6 パスオブジェクト

C++ Java C# ***int add_path_point(int path, double x, double y, String nametype, String optlist)***

Perl PHP ***int add_path_point(int path, float x, float y, string type, string optlist)***

C ***int PDF_add_path_point(PDF *p, int path, double x, double y, const char *type, const char *optlist)***

新規の、または既存のパスオブジェクトに点かパスを追加します。

path それ以前に *PDF_add_path_point( )* を呼び出して返された有効なパスハンドルか、または新規パスを作成するには -1（PHP では 0）。

x・y 新規カレント点の座標。*polar=false* にすると、この 2 個の数は、点のデカルト座標 *(x, y)* を表します。*polar=true* にすると、この 2 個の数は、点の動径 *r* と偏角 *phi*（*radians* オプションに従って度単位またはラジアン単位で）を表します。*type=circle・circular・elliptical・ellipse・move・line・curve・rect* のときは、この点が新たにカレント点となります。

type 表 7.5 に従った、点の種別。

表 7.5 PDF_add_path_point( ) の点の種別

種別	説明
addpath	パスに、svgpath オプションで指定されたパス定義を完全なサブパスとして追加。(x, y) を中心として用います。
circle	パスに、完全なサブパスとして円を追加。(x, y) を中心、radius をサイズとして用います。[1]
circular	カレント点から (x, y) への円弧を追加。あらかじめ、3 番目の円弧点として制御点を定義しておく必要があります。新規点をカレント点と同一にすると、カレント点と制御点を結ぶ線分を直径とする円が作成されます。[2]
control	ベジエ曲線の、または円弧の制御点。[2]
curve	カレント点から新規点へ、あらかじめ定義しておいた制御点群を用いてベジエ曲線を追加。少なくとも 1 個の制御点を与える必要があります。制御点を 1 個だけ与えると、これは曲線の 2 番目の制御点として用いられ、1 番目の制御点は、2 番目の制御点に対する、直前のベジエ曲線の終点における鏡像として算出されます。[2]
ellipse	パスに、完全なサブパスとして楕円を追加。(x, y) を中心、radius オプションの値群をサイズとして用います。[1] この楕円は、xrotate オプションを用いて回転することもできます。
elliptical	カレント点から (x, y) への楕円弧を追加。楕円のサイズと向きは、radius・xrotate・largearc・clockwise オプションによって定義されます。radius として値を 1 個だけ与えた場合には、円弧が作成されます。この場合には、適切な円弧点が自動的に作成されます。radius オプションで 2 個の値を与えた場合には、ベジエ曲線の集合が作成されます。[2]
line	カレント点から (x, y) への線分を追加。[2]
move	新規サブパスを開始。サブパスは連続的に付番されます（1, 2, …）。最初のサブパスは原点を始点とします。
pathref	path オプションで指定されたパスへの参照を、完全なサブパスとして追加。(x, y) を原点として用います。パスは参照されますので（複製はされませんので）、path に対する以後の偏向は、パスを描く際に反映されます。
rect	パスに、完全なサブパスとして長方形を追加。(x, y) を中心、width・height をサイズとして用います。[1] 長方形の隅を、round オプションを用いて丸めることもできます。あるいは、隅を、radius オプションを用いて楕円弧で丸めることもできます。

1. パスの後に自動的に、type=move と、同じ座標とグラフィック書式オプション群を持つ新たな点が作成されます。
2. これらの種別では、グラフィック書式オプション群とパス操作オプション群は許容されません。

optlist パス構築オプションを指定したオプションリスト：

- 点 1 個に対する、表 7.6 に従ったパス計算・命名オプション：
  name・polar・radians・relative
- 表 7.6 に従ったパス操作オプション：
  close・fill・round・stroke
- 表 7.6 に従った、パス定義を追加するためのオプション：
  path・svgpath
- 表 7.6 に従った、パス要素を構築するためのオプション：
  clockwise・height・largearc・radius・rectify・width・xrotate
- 表 7.1 に従ったグラフィック書式オプション（*type=addpath・circle・ellipse・move・rect・pathref* のいずれかの場合のみ）：
  dasharray・dashphase・fillcolor・fillrule・flatness・gstate・linecap・linejoin・linewidth・miterlimit・strokecolor

戻り値 パスハンドル。*PDF_delete_path( )* で削除するまで使えます。

詳細 パスオブジェクトは、ベクトル図形のためのコンテナとして機能します。パスオブジェクトには、パスとサブパスを 1 個ずつ加えていくことができます。ここで新規パス要素は、個々のパスノードを指定することによって、あるいはパスハンドルか SVG パス記述を通じて指定されたパス定義を追加することによって、作成することができます。作成されたパスは、その後、*PDF_draw_path( )* などのメソッドで用いることができます。

パスオブジェクトは、任意の数のパスを保持することができます。各パスはさらに、1 個ないし複数のサブパスを含むことができます。サブパスは、*PDF_draw_path( )* の *subpaths* オプションで、描くために選択することができます。すべてのパスは、指定したオプションに従って個別に、閉じられ、塗られ、描線され、丸められます。

種別 *addpath・circle・ellipse・move・rect・pathref* のいずれかを用いた操作は、新規サブパスを開始させます。グラフィック書式オプション群とパス操作オプション群（*stroke・fill* など）は、*type=addpath・circle・ellipse・move・rect・pathref* のいずれかのためにのみ変更できます。この場合には、パスオブジェクト内の新規パスが自動的に開始されます。種別 *circle・ellipse・elliptical・rect* の形状は、デフォルトでは反時計回りの向きに作成されますが、これをオプション *clockwise* で変更することもできます。

スコープ 任意

C++ Java C# ***void draw_path(int path, double x, double y, String optlist)***

Perl PHP ***draw_path(int path, float x, float y, string optlist)***

C ***void PDF_draw_path(PDF *p, int path, double x, double y, const char *optlist)***

パスオブジェクトを描きます。

path *PDF_add_path_point( )* を、またはパスハンドルを返すその他のメソッド（例：*PDF_info_image( )* で *boundingbox* キーワードを指定）を呼び出して返された有効なパスハンドル。

表 7.6 PDF_add_path_point( ) のオプション

オプション	説明
clockwise	（論理値。type=circle・ellipse・elliptical・rect でのみ可）true にすると、その形状は時計回りの向きに作成されます。そうでない場合は反時計回り。デフォルト：false
close	（論理値。type=move でのみ可）true にすると、サブパスは直線で閉じられます。デフォルト：脚注を参照[1]
fill	（論理値。type=move でのみ可）true にすると、サブパスは閉じられて塗られます。デフォルト：脚注を参照[1]
height	（float。type=rect でのみ可。その場合には必須）長方形の高さ
largearc	（論理値。type=elliptical でのみ可）true にすると、大きい楕円弧のうちの 1 つが作成されます。そうでない場合は、小さい楕円弧のうちの 1 つが作成されます。デフォルト：false
name	（文字列）点の名前。デフォルト：p<i>（例：p1）、ここで i は、与えた点の連続番号です。
path	（パスハンドル。type=pathref でのみ可）指定されたパスが、カレントパスに、参照によって追加されます。追加されるパスの座標は、カレント点を原点として参照します。グラフィック書式オプション群と name オプションは無視されます。
polar	（論理値）true にすると、引数 (x, y) は極座標で動径 r と偏角 phi を表し、そうでないならデカルト座標で値 x・y を表します。デフォルト：false
radians	（論理値）true にすると、極座標における偏角はラジアンで表され、そうでないなら度単位で表されます。デフォルト：false
radius	（1 個か 2 個の float。type=circle・ellipse・elliptical では必須。type=rect でも可）1 つ目の値は、円の半径または楕円の x 半径を指定します。2 つ目の float 値は、存在する場合には、楕円の y 半径を指定します。1 つ目の値が、2 つ目の値に対するデフォルトとして用いられます。 type=rect の場合には、これらの値は、長方形の角の楕円弧の x・y 半径を指定します。この楕円弧はただちに作成されます。デフォルト：0
rectify	（論理値。type=ellipse・elliptical でのみ可）true にすると、小さすぎる半径は、楕円が構築できるよう変更されます。そうでない場合は、例外が発生します。デフォルト：false
relative	（論理値）true にすると、(x, y) はカレント点に対して相対的となり、そうでないならカレント原点に対して相対的となります。デフォルト：脚注を参照[1]

表 7.6 PDF_add_path_point( ) のオプション

オプション	説明
round	（float。type=move・rect でのみ可）サブパス内の隣りあう線の頂点が、それらの接合点で、指定した半径を持ち、それらの線分を接線とする円弧によって丸められます。半径を負にすると、角が円形にへこむように弧が切り取られます。close=true とすると、終了点から開始点への線を指定していないならば、最初の線と閉じる線も丸められます。round=0 とすると丸めは行われません。円弧は、パスが描かれる時点で作成されます。デフォルト：脚注を参照 [1]
stroke	（論理値。type=move でのみ可）true にすると、サブパスは描線されます。デフォルト：脚注を参照 [1]
svgpath	（文字列。type=addpath でのみ可）に従った SVG 文法のパス記述を内容とする文字列（path SVG エレメントの d 属性に与えられるものと同じ。www.w3.org/TR/SVG11/paths.html#PathData を参照）。指定されたパスはカレントパスに追加されます。この SVG パスの座標は、カレント点を原点として参照します。グラフィック書式オプション群をこの SVG パスに対して指定することもできます。そのパスが SVG ファイルから下向き座標系によって生じている場合には、それを平行移動・反転させる必要があります（たとえ PDFlib が topdown モードで動作していても）。詳しくは PDFlib チュートリアルを参照。
width	（float。type=rect でのみ可。その場合には必須）長方形の幅
xrotate	（float。type=ellipse・elliptical でのみ可）楕円に対する回転角をカレント単位で表したもの（オプション radians を参照）、すなわち、楕円の x 軸の、カレント座標系の x 軸に対する角度を、度単位で表したもの。楕円弧の開始点と終了点は固定されたままです。このオプションは、radius として値 1 個だけが与えられた場合には無視されます。デフォルト：0

1. デフォルトは、PDF_draw_path( ) でも、PDF_info_path( ) でも、PDF_fit_textline( ) の textpath オプションでも、PDF_fit_textflow( ) の wrap オプションでも、PDF_add_table_cell( ) の fitpath オプションでも指定されます。

x・y 参照点の座標を、ユーザー座標で指定します。この参照点はさまざまなオプションで用いられ、かつパスの原点のカレントユーザー座標系における位置を指定します。これにより、パスオブジェクトが平行移動します。

usematchbox オプションを与えている場合には ***x・y*** は無視されます。

boxsize オプションを指定すると、***(x, y)*** は、パスオブジェクトがはめ込まれるはめ込み枠（表 6.1 参照）の左下隅となります。

optlist パス描画オプションを指定したオプションリスト：

- 表 6.1 に従ったはめ込みオプション：
  align・attachmentpoint・boxsize・fitmethod・orientate・position・scale
- 表 7.7 に従ったパス操作・サブパス選択オプション：
  clip・close・fill・round・stroke・subpaths
- 表 7.7 に従った枠オプション：
  bboxexpand・boundingbox・usematchbox
- 表 7.1 に従った、***fill・stroke*** オプションのためのグラフィック書式オプション：
  dasharray・dashphase・fillcolor・flatness・gstate・linecap・linejoin・linewidth・miterlimit・strokecolor
- 表 7.1 に従った ***clip*** オプションのための、表 7.1 に従ったグラフィック書式オプション：
  cliprule・fillrule
- 表 14.4 に従った、短縮構造エレメントタグ付けのためのオプション（**ページ**スコープでのみ可）：***tag***

詳細　パス（群）は、参照点 *(x, y)* に配置され、その後、指定したオプション群に従って、描線されたり、塗られたり、クリッピングパスとして用いられたりします。このメソッドは、*clip* オプションを用いていない限り、カレントグラフィックステートに変更を加えることはありません。書式・操作オプション群はデフォルト設定をオーバーライドしますが、*PDF_add_path_point()* の中のサブパスに対して書式オプションを指定していた場合は、それをオーバーライドすることは一切ありません。

PDF/UA　すべてのパスオブジェクトは、*tag* オプションを用いて、あるいは事前の *PDF_begin_item()* への呼び出しを用いて、*Artifact* か *Figure* としてタグ付けされる必要があります。

スコープ　ページ・パターン・テンプレート・グリフ

表 7.7　パスオブジェクト内のすべてのサブパスを制御するための PDF_draw_path( ) のパス操作オプション

オプション	説明
bboxexpand	（float のリスト。boundingbox オプションを指定している場合には無視されます） 自動的に算出された外接枠（パスオブジェクトを囲む最小の長方形）の拡張を示す 1 個か 2 個の float。デフォルト：{0 0}
bounding-box	（長方形）パスオブジェクトをはめ込み枠内へはめ込むための外接枠として用いられる、パスオブジェクトの長方形を座標系で表したもの。デフォルト：パスオブジェクトを囲む最小の長方形、ただし bboxexpand オプションに従って拡張されている可能性もあります
clip	（論理値）true にすると、パスは閉じられ、クリッピングパスとして用いられます。デフォルト：false
close	（論理値）true にすると、各サブパスが直線で閉じられます。デフォルト：パスが構築された際に指定された値、あるいは値が何も指定されなかったなら false
fill	（論理値。clip をオーバーライドします）true にすると、各パスが塗られます。デフォルト：パスが構築された際に指定された値、あるいは値が何も指定されなかったなら false
round	（float）各サブパスについて、隣りあう線の頂点が、それらの接合点で、指定した半径を持ち、それらの線分を接線とする円弧によって丸められます。半径を負にすると、角が円形にへこむように弧が切り取られます。close=true とすると、終了点から開始点への線を指定していないならば、最初の線と閉じる線も丸められます。round=0 とすると丸めは行われません。デフォルト：パスが構築された際に指定された値、あるいは値が何も指定されなかったなら 0
stroke	（論理値。clip をオーバーライドします）true にすると、パスは描線されます。デフォルト：false
subpaths	（整数のリスト、またはキーワード 1 個）描きたいサブパスの番号のリスト。最初のサブパスは番号 1 です。キーワード all ですべてのサブパスが指定されます。デフォルト：all
usematchbox	（文字列のリスト。ページスコープでのみ可）指名した範囲枠のすべての長方形の中にパスを描きます。このオプションリストの中の最初の要素は、範囲枠を指定する名前文字列です。2 番目の要素は、望む長方形の番号（1 から始まる）を指定する整数にするか、あるいはその範囲枠のすべての長方形を指定したければキーワード all にします。2 番目の要素が指定されていない場合には、デフォルトとして all になります。 この範囲枠の、指定した長方形（群）の中に、与えたパスが描かれます。x/y 引数のかわりにその長方形の左下隅が参照点として用いられ、その長方形の幅と高さが boxsize として用いられます。引数 x/y とオプション boxsize は、usematchbox を与えた場合には無視されます。 その範囲枠か、指定した長方形がカレントページに存在しない場合には、このメソッドは何も描かず、エラーも出さずに返ります。

C++ Java C# ***double info_path(int path, String keyword, String optlist)***

Perl PHP ***float info_path(int path, string keyword, string optlist)***

C ***double PDF_info_path(PDF *p, int path, const char *keyword, const char *optlist)***

パスオブジェクトを描いた結果を、実際にそれを描くことなく取得します。

path *PDF_add_path_point( )* を、またはパスハンドルを返すその他のメソッド（例：*PDF_info_image( )* で *boundingbox* キーワードを指定）を呼び出して返された有効なパスハンドル。

keyword 求める情報を指定したキーワード：

- 表 6.3 に従った、オブジェクトはめ込みの結果をクエリーするためのキーワード：*boundingbox*・*fitscalex*・*fitscaley*・*height*・*objectheight*・*objectwidth*・*width*・*x1*・*y1*・*x2*・*y2*・*x3*・*y3*・*x4*・*y4*
- 表 7.8 に従ったさらなるキーワード：*bboxwidth*・*bboxheight*・*numpoints*・*pathlength*・*px*・*py*

表 7.8 PDF_info_path( ) のキーワード

キーワード	説明
bboxwidth*・*bboxheight	パスに対する外接枠の幅と高さ
numpoints	与えられた点の数。subpaths オプションは無視されます。
px*・*py	name オプションで指定したパス点の x・y 座標（ユーザー座標で）。subpaths オプションは無視されます。
pathlength	パスの道のりの長さ。
type	オプション name で指定した点の種別の数値識別子。種別「楕円孤」・「円」・「楕円」のパスコンポーネントはすでにベジエ曲線群へ変換済、種別「長方形」は直線群へ変換済です： ***0*** move ***1*** line ***2*** control ***3*** curve ***4*** circular

optlist パス描画オプションを指定したオプションリスト：

- 表 7.7 に従った *PDF_draw_path( )* のすべてのオプション
- および、表 6.1 に従ったはめ込みオプション：*refpoint*
- および、表 7.9 に従ったオプション：*name*

戻り値 キーワードで求められた何らかのパス特性の値。

詳細 このメソッドは、*PDF_draw_path( )* と同じ計算を実行しますが、ページ上に目に見える出力を一切生成しません。

スコープ 任意

表 7.9 PDF_info_path( ) のオプション

オプション	説明
name	キーワード px か py か type に対するパス点の名前。PDF_add_path_point( ) で明示的な名前を指定してあっても、デフォルト名（p1 等）も用いることができます。

C++ ***void delete_path(int path)***

Perl PHP ***delete_path(int path)***

C ***void PDF_delete_path(PDF *p, int path)***

パスオブジェクトを削除します。

path *PDF_add_path_point( )* を、またはパスハンドルを返すその他のメソッド（例：*PDF_info_image( )* で *boundingbox* キーワードを指定）を呼び出して返された有効なパスハンドル。

詳細 パスオブジェクトと、それに関連したすべての内部データ構造を削除します。*PDF_end_document( )* でパスオブジェクトが自動的に削除されるわけではありません。

スコープ 任意

8 色メソッド

この章の API メソッド：

- *PDF_setcolor( )*
- *PDF_load_iccprofile( )*
- *PDF_makespotcolor( )*
- *PDF_create_devicen( )*
- *PDF_shading( )*
- *PDF_shading_pattern( )*
- *PDF_shfill( )*
- *PDF_begin_pattern_ext( )*
- *PDF_end_pattern( )*

8.1 色を設定

塗り色・描線色を設定するには、*PDF_setcolor( )* を用いるか、あるいはオプション *fillcolor*・*strokecolor* などを用います。オプションを用いる方法のほうが汎用的ですので推奨します。13 ページ「1.1.4 章 色データ型」にカラーオプションに関する説明があります。

C++ Java C# ***void setcolor(String fstype, String colorspace, double c1, double c2, double c3, double c4)***

Perl PHP ***setcolor(string fstype, string colorspace, float c1, float c2, float c3, float c4)***

C ***void PDF_setcolor(PDF *p,***
const char *fstype, const char *colorspace, double c1, double c2, double c3, double c4)

グラフィック・テキストステートのためのカレント色空間と色を設定します。

fstype *fill*・*stroke*・*fillstroke* のいずれかで、塗り・描線・両方のいずれに色を設定したいのかを指定します。

colorspace 指定するカラー値群に対して用いたい色空間か、または RGB カラー値を名前か 16 進値で指定します。

- 1 番目の形式：*gray*・*rgb*・*cmyk*・*spot*・*devicen*・*pattern*・*iccbasedgray*・*iccbasedrgb*・*iccbasedcmyk*・*lab* のいずれか 1 つで、色空間を指定します。
- 2 番目の形式：RGB カラー名（例：*pink*）か、またはハッシュキャラクターの後に 6 桁の 16 進数（例：*#FFC0CB*）。この場合、引数 *c1*・*c2*・*c3*・*c4* は無視されます。

c1*・*c2*・*c3*・*c4 選んでいる色空間における色要素。これらの値の解釈は、*colorspace* 引数によって異なります（色空間と値の完全な説明については PDFlib チュートリアルを参照してください）：

- *gray*：*c1* でグレー値を指定します。
- *rgb*：*c1*・*c2*・*c3* で赤・緑・青の値を指定します。
- *cmyk*：*c1*・*c2*・*c3*・*c4* でシアン・マゼンタ・イエロー・黒の値を指定します。
- *iccbasedgray*：*c1* でグレー値を指定します。
- *iccbasedrgb*：*c1*・*c2*・*c3* で赤・緑・青の値を指定します。
- *iccbasedcmyk*：*c1*・*c2*・*c3*・*c4* でシアン・マゼンタ・イエロー・黒の値を指定します。

- *spot* : *c1* で、*PDF_makespotcolor( )* によって返されたスポットカラーハンドルを指定し、*c2* で、0 から 1 までの濃度値を指定します。
- *devicen* : *c1* で、*PDF_create_devicen( )* によって返された DeviceN 色空間ハンドルを指定します。*c2*・*c3*・*c4* で、0 ～ 1 の濃度値を 3 個まで指定します。N>3 の DeviceN カラーをこのメソッドで指定することはできません。
- *lab* : *c1*・*c2*・*c3* で、CIE L*a*b* 色空間内のカラー値を指定します。*c1* で *L** （輝度）値を範囲 0 ～ 100 で指定し、*c2*・*c3* で *a**・*b** （色度）値を範囲 -128 ～ 127 で指定します。
- *pattern* : *c1* で、*PDF_begin_pattern_ext( )* によって返されたタイリングパターンハンドルを、あるいは、*PDF_shading_pattern( )* によって返されたシェーディングパターンハンドルを指定します。タイリングパターンが *PDF_begin_pattern_ext( )* でオプション *painttype=uncolored* を用いて作成されている場合、そのパターンが塗りか描線に使用される際には、カレントの塗り色または描線色が適用されます。この場合には、カレント色空間がまた別のパターン色空間であってはいけません。

詳細 *gray*・*rgb*・*cmyk* 色空間におけるすべてのカラー値と、*spot* 色空間における濃度値は、0 以上 1 以下の数値にする必要があります。使わない引数は、0 に設定するべきです。色空間とカラー値に関してさらに詳しい情報が PDFlib チュートリアルにあります。

gray・*rgb*・*cmyk* 色空間における描線・塗り色の値は、ページの始まりごとに、デフォルト値である黒に設定されます。スポット・パターンカラーの場合はデフォルトはありません。

iccbasedgray/rgb/cmyk 色空間を使う場合には、*iccprofilegray/rgb/cmyk* オプションのうちのいずれか 1 つを使う前に、適切な ICC プロファイルを設定しておく必要があります。

このメソッドは、*fillcolor* および / または *strokecolor* オプションを伴う *PDF_set_text_option( )*・*PDF_set_graphics_option( )* と等価です。*PDF_setcolor( )* はこれらのオプションの値をオーバーライドします。

iccbased 色空間の色と、N>3 の DeviceN カラーは、このメソッドでは指定できず、カラーオプションでのみ指定できます。

PDF/A *colorspace=gray* は出力インテント（任意の種類）か *defaultgray* オプションを必要とします。*colorspace=rgb* は RGB 出力インテントか *defaultrgb* オプションを必要とします。*colorspace=cmyk* は CMYK 出力インテントか *defaultcmyk* オプションを必要とします。

PDF/UA 情報を色かコントラストだけで伝達するべきではありません。

PDF/X-3 : *iccbasedgray/rgb/cmyk*・*lab* カラーは、出力インテントの中に ICC プロファイルを必要とします（この場合、標準名では不十分です）。

PDF/X-3/4/5p : *colorspace=gray* は、グレースケールまたは CMYK デバイスインテントか *defaultgray* オプションを必要とします。

colorspace=rgb は RGB 出力インテントか *defaultrgb* オプションを必要とします。

colorspace=cmyk は CMYK 出力インテントか *defaultcmyk* オプションを必要とします。

PDF/X-5n : *colorspace=gray* は、出力インテントがインキ *Black* を含んでいるか、*defaultgray* オプションが設定されている場合にのみ使用できます。

colorspace=rgb は、*defaultrgb* オプションが設定されている場合にのみ使用できます。

colorspace=cmyk は、出力インテントがインキ *Cyan*・*Magenta*・*Yellow*・*Black* をすべて含んでいるか、*defaultcmyk* オプションが設定されている場合にのみ使用できます。

スコープ **ページ・パターン**（*painttype=colored* の場合のみ）・**テンプレート・グリフ**（グリフの *colorized* オプションが *true* の場合のみ）・**文書**。パターンカラーは、それ自身の定義の中

では使えません。**文書**スコープ内での色の設定は、*PDF_makespotcolor()* でスポットカラーを定義する際に有用です。

8.2 ICC プロファイル

C++ Java C# ***int load_iccprofile(String profilename, String optlist)***

Perl PHP ***int load_iccprofile(string profilename, string optlist)***

C ***int PDF_load_iccprofile(PDF *p, const char *profilename, int len, const char *optlist)***

ICC プロファイルを検索して、以後の使用に備えます。

profilename （名前文字列）*ICCProfile* リソースの名前か、またはディスクベースのファイルか仮想ファイルの名前。

len （C 言語バインディングのみ）*profilename* の長さ（バイト単位）。*len=0* にすると null 終了文字列を与える必要があります。

optlist プロファイル処理の諸特性を記述したオプションリスト：

- 一般オプション：*errorpolicy*（表 1.5 参照）
- 表 8.1 に従ったプロファイル処理オプション：*description*・*embedprofile*・*outputintenttype*・*urls*・*usage*

表 8.1 PDF_load_iccprofile( ) のオプション

オプション	説明
description	（文字列。usage=outputintent の場合にのみ可）出力インテントとともに用いられる ICC プロファイルに関する人間向けの説明。
embedprofile	（論理型。usage=outputintent・usage=pageoutputintent の場合にのみ意味を持ちます）出力インテント ICC プロファイルの埋め込みか紐付けを制御します： PDF/X-5n：true にすると、ICC プロファイルを文書に、埋め込みファイルストリームとして紐付けされます。デフォルト：true PDF 2.0：true にすると、ICC プロファイルが埋め込まれます。そうでない場合、外部参照が作成されます。デフォルト：true PDF/X-4p：このオプションは強制的に false になります。すなわち、プロファイルは外部参照されます。 PDF/X-3/4 などその他すべての場合：このオプションは強制的に true になります。すなわち、プロファイルは埋め込まれます。sRGB プロファイルはつねに埋め込まれます。
outputintent-type	（名前文字列。PDF 2.0。PDF/A・PDF/X では不可。usage=outputintent・pageoutputintent の場合にのみ可）出力インテントのカスタム用途種別。この種別は先頭が GTS_・Plib_ ではいけません。デフォルト：Plib_OutputIntent。
urls	（文字列 1 個ないし複数のリスト。PDF 2.0・PDF/X-4p でのみ可。PDF/X-4p では必須）ICC プロファイルが PDF へ埋め込まれるのではなく、かわりに 1 個ないし複数の外部のプロファイルへの参照が作成されます。このリストは、参照されている出力インテント ICC プロファイルを入手できる場所を示した URL 群を内容とします。送り手と受け手が、適切な URL 項目をアレンジする必要があります。この文字列は自由に選ぶことができますが、有効な URL 文法を内容とする必要があります。

表 8.1 PDF_load_iccprofile( ) のオプション

オプション	説明
usage	（キーワード）ICC プロファイルの想定用途。使えるキーワード（デフォルト：iccbased）： ***iccbased*** ICC プロファイルは、テキストかグラフィックのための ICC ベース色空間として使われるか、画像に適用されるか、デフォルト色空間として使われるか、透過グループのためのブレンディング色空間として使われることが可能です。 ***outputintent*** （PDF/A・PDF/A・PDF 2.0 でのみ可）ICC プロファイルは、文書全体の出力インテントを暗黙的に指定します。PDF 2.0 ではこのプロファイルはページ独自の出力インテントとしても使用される場合があります。 ***pageoutputintent*** （PDF 2.0）ICC プロファイルは、1 つないし複数のページに対する出力インテントを指定します。返される ICC プロファイルハンドルを PDF_begin_page_ext( ) に与えることができます。

戻り値 *usage=iccbased* の場合には、返された ICC プロファイルハンドルを、以後の *PDF_load_image( )* への呼び出しで使うか、またはプロファイル関連のオプションを設定するために使うことができます。

usage=pageoutputintent の場合には、返された出力インテントハンドルを、*PDF_begin_page_ext( )* の *outputintents* オプションで使うことができます。

errorpolicy=return の場合、戻り値 -1（PHP では 0）はエラーを示しますので、そうでないかを呼び出し側で検査する必要があります。返されたハンドルは、複数の PDF 文書にわたって再利用することはできません。メソッドの呼び出しが失敗したときは、その失敗の原因は、*PDF_get_errmsg( )* を使って取得することができます。

詳細 名前付きプロファイルは、プロファイル検索戦略に従って検索されます。用途によっては、ICC プロファイルは、PDFlib チュートリアルに挙げている条件を満たしている必要があります。*sRGB* プロファイル（*srgb* としても利用可能です）は、つねに内部的に得られますので、構成する必要はありません。

PDF/A 文書レベルの出力インテントは、このメソッドを使うか、または、*PDF_process_pdi( )* を使って取り込み文書の出力インテントをコピーして、設定することができます。デバイス独立な色だけを文書で使っているときは、出力インテントは必要ありません。

PDF/X 文書レベルの出力インテントは、このメソッドを使うか、または、*PDF_process_pdi( )* を使って取り込み文書の出力インテントをコピーして、設定する必要があります。

PDF/X-4：文書レベルの出力インテント ICC プロファイルが、指定される必要があり、埋め込まれます。

PDF/X-4/5：CMYK 出力インテントプロファイル（すなわち *usage=outputintent* を用いて読み込まれたもの）を、同一文書内で ICC ベース色空間（すなわち *usage=iccbased* を用いて読み込まれたもの）のために使うことはできません。この必要条件は、PDF/X 規格によって義務付けられているものであり、CMYK プロファイルにのみ適用され、グレースケール・RGB プロファイルには適用されません。出力インテント内と同じ CMYK ICC プロファイルを ICC ベースカラーとしても用いたい（画像にタグ付けするため等）という必要がある場合には、単にその ICC プロファイルを省略すれば足ります。なぜなら PDF/X では、その出力インテントプロファイルがいずれにせよ用いられることを暗黙に前提しているからです。

PDF/X-4p：プロファイルは埋め込まれませんが、外部プロファイルへの参照が作成されます。このプロファイルは、PDF を生成する際に得られる必要があり、かつ、PDF 消費者がその文書を閲覧または印刷する際にも得られる必要があります。

PDF/X-5n:n 色 ICC プロファイル（*xCLR* プロファイルともいう）を、*usage=outputintent* に対して与える必要があります。外部参照された出力インテント ICC プロファイルは、*embedprofile* オプションに従って、文書内に紐付ける形で含めることも可能です。

スコープ *usage=outputintent* の場合、許される唯一のスコープは**文書**。文書レベルの出力インテントは *PDF_begin_document()* の直後に設定する必要があります。

usage=pageoutputintent の場合、以下のスコープが許されます：**文書・ページ・パターン・テンプレート・グリフ**。

usage=iccbased の場合、次のスコープが許されます：**文書・ページ・パターン・テンプレート・グリフ**。

8.3 スポットカラー

C++ Java C# ***int makespotcolor(String spotname)***

Perl PHP ***int makespotcolor(string spotname)***

C ***int PDF_makespotcolor(PDF *p, const char *spotname, int reserved)***

組み込みスポットカラーの名前を検索するか、または、カレント塗り色から名前付きスポットカラーを作ります。

spotname 組み込みスポットカラーの名前か、または、定義したいスポットカラーにつけたい任意の名前。この名前は Unicode キャラクター 63 個までに制限されています。

特殊なスポットカラー名 ***All*** を使うと、色をすべての色分版に適用することができますので、トンボを描く際に便利です。スポットカラー名 ***None*** を指定すると、どの色分版にも、目に見える出力がまったく生成されなくなります。インキ名 ***Cyan*・*Magenta*・*Yellow*・*Black*** はつねに CMYK プロセスカラーを参照します。

reserved （C 言語バインディングのみ）予約済。0 にする必要があります。

戻り値 スポットカラーハンドル。以後の *PDF_setcolor( )* への呼び出しで、また、*PDF_set_graphics_option( )* などのメソッドの ***fillcolor*・*strokecolor*** オプションに対して使えます。スポットカラーハンドルは、すべてのページにわたって再利用できますが、文書を越えた再利用はできません。

詳細 ***spotname*** がもしも、PANTONE・HKS カラー群の内蔵カラーテーブルにおいて既知であり、かつ、グローバルな ***spotcolorlookup*** オプションが ***true***（これがデフォルトです）の場合には、指定されたスポットカラー名と、対応する内部 ***Lab*** 代替カラー値群が使われます。そうでない場合には、カレント塗り色のカラー値を使って新規スポットカラーの視覚表現が定義されます。

spotname がもしも、以前に *PDF_makespotcolor( )* を呼び出した時にすでに使っていたものであれば、戻り値は以前の呼び出しと同じになり、カレント塗り色が代替色として使われることもありません。

このメソッドは通常、**色**型のオプション群を与えることによって、使わずに済みます。この方法であれば、スポットカラー名または定義とスポットカラーの利用とを、スポットカラーハンドルを受け渡す必要なしに、ただ 1 つのオプションリストの中で実現できます（表 1.2 に例がありますので参照してください）。

このメソッドの動作に影響するグローバルオプションを表 8.2 に挙げます（25 ページ「2.1 グローバルオプション」参照）。

スコープ **ページ・パターン・テンプレート・グリフ・文書**。カスタムスポットカラーが定義される場合、カレント塗り色はスポットカラー・DeviceN カラー・パターンであってはいけません。

表 8.2 PDF_set_option( ) のスポットカラー関連グローバルオプション

オプション	説明
spotcolorlookup	（論理値）false にすると、PDFlib は、スポットカラー名の内蔵データベースを使いません。その場合は、既知のスポットカラーにカスタムの定義を与えることができます。カスタム定義は、他のアプリケーションで使われる定義との整合をとる手段として必要になることがあります。この機能は、使用には注意が必要であり、推奨しません。デフォルト：true

8.4 DeviceN カラー

C++ Java C# ***int create_devicen(String optlist)***

Perl PHP ***int create_devicen(string optlist)***

C ***int PDF_create_devicen(PDF *p, const char *optlist)***

任意の数の色要素を持つ DeviceN 色空間を作成します。

optlist カラー作成オプションを指定したオプションリスト

- 一般オプション：***errorpolicy***（表 1.5 参照）
- 表 8.3 に従った DeviceN オプション：***alternate*・*names*・*process*・*subtype*・*transform***

戻り値 DeviceN 色空間ハンドル。これを *PDF_set_graphics_option()* などのメソッドの ***fillcolor*・*strokecolor*** オプションで使用できます。この DeviceN 色空間ハンドルは、それを囲う**文書**スコープの終了まで使用できます。

デフォルトではこのメソッドはエラー時には例外を発生させます。しかしこの動作は、***errorpolicy*** オプションを用いて、エラー戻り値 -1（PHP では 0）へ変更することも可能です。

PDF/A PDF/A-1/2/3：***alternate*** オプションで与えられる色空間は以下の制約に従っている必要があります：

alternate=devicegray は、グレースケールか RGB か CMYK の出力インテントまたは ***defaultgray*** オプションを必要とします。

alternate=devicergb は RGB 出力インテントか ***defaultrgb*** オプションを必要とします。

alternate=devicecmyk は CMYK 出力インテントか ***defaultcmyk*** オプションを必要とします。

PDF/A-2/3：*PDF_create_devicen()* の前に、その DeviceN 色空間で使用されるすべてのカスタムスポットカラーについて *PDF_makespotcolor()* を呼び出す必要があります。

PDF/X ***alternate*** オプションで与えられる色空間は以下の制約に従っている必要があります：

PDF/X-3：***alternate=iccbased*・*lab*** は、グレースケールか RGB か CMYK の出力インテントを必要とします。

PDF/X-3/4/5n：***alternate=devicegray*** は、グレースケールか CMYK の出力インテントまたは ***defaultgray*** オプションを必要とします。

alternate=devicergb は RGB 出力インテントか ***defaultrgb*** オプションを必要とします。

alternate=devicecmyk は CMYK 出力インテントか ***defaultcmyk*** オプションを必要とします。

PDF/X-4：***process*** オプションの ***colorspace*** サブオプションは PDF/X 出力インテントに合致している必要があります。

PDF/X-4・PDF/X-5n：*PDF_create_devicen()* の前に、その DeviceN 色空間で使用されるすべてのカスタムスポットカラーについて *PDF_makespotcolor()* を呼び出す必要があります。PDF/X-5n では、出力インテントのインキリストの中に見つかるスポットカラーについてはこの必要条件の対象外です。

PDF/X-5n：オプション ***subtype=nchannel*・*process*** は不可。

スコープ **オブジェクト**以外任意

表 8.3 PDF_create_devicen( ) のオプション

オプション	説明
alternate	（キーワードかオプションリスト。PDF/A・PDF/X では制約が課せられます。必須）その DeviceN 色空間に対する代替色空間。以下のキーワードを与えることができます：devicegray・devicergb・devicecmyk・lab。あるいは、以下のサブオプションを持ったオプションリストを与えることもできます： ***iccbased*** （キーワードか ICC プロファイルハンドル）ハンドルかキーワード *srgb* によって指定された ICC プロファイル。この ICC プロファイルは、usage=iccbased を用いて読み込まれている必要があります。
names	（名前文字列のリスト。必須）32 個までのインキ名（PDF 1.4 では 8 個までのインキ名）を内容とするリスト。すべての名前は互いに異なっている必要があります。ただし None は複数回現れることも可能です。インキ名 All は不可。インキ名 Cyan・Magenta・Yellow・Black はつねに CMYK プロセスカラーを参照します。 subtype=nchannel の場合には、インキ名 None は不可であり、その DeviceN 色空間の中で使用されるすべてのスポットカラーの名前が PDFlib に知られている必要があります。すなわち、それらは内部スポットカラーデータベースに含まれているか、あるいは PDF_makespotcolor( ) がすでに呼ばれている必要があります。
process	（オプションリスト。PDF 1.6 以上。subtype=nchannel かつ names オプションが 1 個ないし複数のプロセスカラーを含んでいる場合には必須。PDF/X-5n では不可）： ***colorspace***（キーワード。必須）プロセス色空間：devicegray か devicergb か devicecmyk。PDF/X-4：この色空間は PDF/X 出力インテントに合致している必要があります。 ***components*** （文字列のリスト。必須）colorspace サブオプションで指定されたプロセス色空間のすべての要素の名前。この要素名のリストは、要素名 None または内蔵かカスタムのスポットカラー名を含むことはできません。names オプションが 1 個ないし複数のプロセスカラーを含んでいる場合、それらはここで与える名前と一致している必要がありますが、ただしこの名前は通常の名前と異なっていてもかまいません（たとえば、colorspace=devicecmyk に対して Cyan のかわりに Process Cyan を用いることも可能）
subtype	（キーワード。PDF 1.6）その色空間に対する望ましい取り扱い（デフォルト：devicen）： ***devicen*** その色空間はプレーンな DeviceN 色空間として扱われます。 ***nchannel*** （PDF/X-5n では不可）その色空間は NChannel 色空間として扱われます。この場合、names オプションには特定の制約が課せられます。
transform	（文字列。必須）その DeviceN 色空間の変換メソッドの PostScript コード。この変換メソッドは、N 個の範囲 0 ～ 1 の濃度値を、その代替色空間の中の値群へ変換するものである必要があります。その出力値の数は、その代替色空間の要素数に対応する必要があります。PostScript コードは中括弧キャラクター { } で囲う必要があり、またオプションリスト文字列も空白を内容に含む場合には中括弧キャラクターで囲う必要があることから、このオプション値は通常、二重の中括弧キャラクターで囲われます。

8.5 シェーディングとシェーディングパターン

C++ Java C# ***int shading(String type, double xo, double yo, double x1, double y1, double c1, double c2, double c3, double c4, String optlist)***

Perl PHP ***int shading(string type, float xo, float yo, float x1, float y1, float c1, float c2, float c3, float c4, string optlist)***

C ***int PDF_shading(PDF *p, const char *type, double xo, double yo, double x1, double y1, double c1, double c2, double c3, double c4, const char *optlist)***

2 個以上の色の間のカラーシェーディング（カラーグラデーション）を定義します。

type そのシェーディングの種類。線形シェーディングなら *axial*、円状シェーディングなら *radial* にする必要があります。

xo・yo 始点（*type=axial* の場合）、または開始円の中心（*type=radial* の場合）。これらの値はユーザー座標系で解釈されます。

x1・y1 終点（*type=axial* の場合）、または終了円の中心（*type=radial* の場合）。これらの値はユーザー座標系で解釈されます。

c1・c2・c3・c4 そのシェーディングの終了色のカラー値群。そのシェーディングの開始色の色空間で解釈されます。もしカレント塗り色空間がスポットカラーならば *c1* は無視され、*c2* は濃度値を内容とします。この引数 *c1・c2・c3・c4* は、*endcolor* または *stopcolors* オプションが与えられている場合には無視されます。また、*N>4* の DeviceN 色空間におけるシェーディングに対しては使用できません。

optlist そのシェーディングの諸側面を表 8.4 に従って記述したオプションリスト。以下のオプションを使用できます：
antialias・boundingbox・end・endcolor・extendo・extend1・N・ro・r1・startcolor・stopcolors・type

戻り値 シェーディングハンドル。以後、それを囲う**文書**スコープの中で、*PDF_shading_pattern()*・*PDF_shfill()* を呼び出す際に使用できます。

詳細 このメソッドは、*startcolor*・*endcolor* オプションで与えられた色の間のシェーディングを作成します。オプション *stopcolors* が与えられている場合には、その指定された色リストの中のすべてのエントリーの間にシェーディングが作成されます。すべての色は同一の色空間のものである必要があり、かつ、パターン色空間を使用していてはいけません。異なるスポットカラー群を使用する場合には、以下のすべての条件を満たす必要があります：

- すべてのストップカラーが、PDFlib に内部的に知られているスポットカラーであるか、Lab 代替値を持つカスタムスポットカラーであるか、直接 Lab カラーであること：直接 Lab カラーが与えられた場合、PDFlib はそのカラーから疑似スポットカラーを作成します。
- ストップカラーのリストの中の、異なるスポットカラーの数が、19 個（PDF 1.4 では 8 個）を超えないこと。
- すべてのスポットカラー名が *All* でないこと。

ストップカラーとして与えられたスポットカラーが上記の条件を 1 つでも破っていれば、例外が発生します。

startcolor オプションのかわりにカレント塗り色を使用することも可能です。*endcolor* オプションのかわりに値 *c1・c2・c3・c4* を使用することも可能です。*startcolor* と *endcolor*

オプションの組み合わせのかわりにオプション *stopcolors* を使用することも可能です。この場合には少なくとも 2 個の色を指定する必要があります。

スコープ　オブジェクト以外任意

表 8.4 PDF_shading( ) のオプション、および shading グラフィック書式オプションのサブオプション

オプション	説明
antialias	（論理値）シェーディングにおいてアンチエイリアシングを有効にするかどうかを指定します。デフォルト：false
boundingbox	（長方形）シェーディングの外接枠をユーザー座標で定義した長方形。この外接枠は、シェーディングが描画される際に一時的なクリッピングパスとして適用されます（カレントクリッピングパスが効力を持っているときはそれに加えて）。このオプションは、PDF_clip( ) を適用せずにシェーディングを切り抜くのに有用でしょう。PDF_shading( ) に対するデフォルト：クリッピングなし shading グラフィック書式オプションに対するデフォルト：範囲枠か表セル長方形
end	（float 2 個かパーセント値 2 個のリスト。PDF_shading( ) では不可）終点の（type=axial の場合）、または終了円上の点の（type=radial の場合）座標を、長方形の幅・高さに対するパーセント値として、またはユーザー座標で指定したもの。デフォルト：{100% 100%}
endcolor	（色。stopcolors を与えている場合には無視されます。shading グラフィック書式オプションに対してはオプション endcolor・stopcolors のうち 1 つが必須）そのシェーディングの終了色。PDF_shading( ) に対するデフォルト：stopcolors オプションが指定されている場合にはその末尾で与えられている色、そうでない場合には引数 c1・c2・c3・c4 で与えられている値群
extendo	（論理値）シェーディングを始点または開始円より先へ拡げるかどうかを指定します。デフォルト：false
extend1	（論理値）シェーディングを終点または終了円より先へ拡げるかどうかを指定します。デフォルト：false
N	（正の float）色遷移メソッドに対する累乗指数。デフォルト：1
r0	（float。type=radial の場合のみ可）開始円の半径をユーザー座標で。デフォルト：0
r1	（float。type=radial の場合のみ可）終了円の半径をユーザー座標で。PDF_shading( ) でのデフォルト：0。shading グラフィック書式オプションでのデフォルト：start と end の間の距離
start	（float 2 個かパーセント値 2 個のリスト。PDF_shading( ) では不可）始点の（type=axial）、または開始円の中心の（type=radial）座標を、長方形の幅・高さに対するパーセント値として、またはユーザー座標で指定したもの。デフォルト：{0% 0%}
startcolor	（色。stopcolors が与えられている場合には無視されます）そのシェーディングの開始色。デフォルト：stopcolors オプションが指定されている場合にはその先頭で与えられている色、そうでない場合には範囲枠か表セルに対するオプション fillcolor で与えられている色

stopcolors	（ペアのリスト。shading グラフィック書式オプションではオプション endcolor・stopcolors のうち 1 つが必須）そのシェーディングのための 2 個以上の色のリスト。各ペアは、中間シェーディング色の位置を閉区間 0 ～ 1 の float 値かパーセント値であらわしたものと、対応するカラー値。カラー値はすべて、同一の色空間を使用している必要があります。ただし「詳細」の項ですでに記したように Lab 代替色を持つスポットカラーを除きます。位置は昇順にソートされている必要がありますが、隣接する位置値どうしが同一でもかまいません。位置 0 に対する値がない場合は先頭のリストエントリーが使用されます。位置 1 に対する値がない場合は末尾のリストエントリーが使用されます。このオプションが与えられている場合には、オプション startcolor・endcolor と引数 c1 ～ c4 とカレント塗り色は無視されます。例：
	`stopcolors={0 red 0.4 magenta 0.75 green 1 black}`
	`stopcolors={0% {cmyk 1 0 0 0} 33% {cmyk 0 0.4 0.3 0}} 100% {cmyk 0 0 0.2 0.8}}`
	`stopcolors={0% {spotname {PANTONE 123 U} 1} 100% {spotname {PANTONE 289 U} 1} }`

表 8.4 PDF_shading( ) のオプション、および shading グラフィック書式オプションのサブオプション

オプション	説明
type	（キーワード）シェーディングの種類：線形シェーディングなら axial、円状シェーディングなら radial。デフォルト：引数 type がある場合にはその値（PDF_shading( ) の場合のみ）、そうでないなら axial

C++ Java C# ***int shading_pattern(int shading, String optlist)***

Perl PHP ***int shading_pattern(int shading, string optlist)***

C ***int PDF_shading_pattern(PDF *p, int shading, const char *optlist)***

シェーディングパターンを、シェーディングオブジェクトを使って定義します。

shading *PDF_shading( )* によって返されたシェーディングハンドル。

optlist そのシェーディングパターンの詳細を記述したオプションリスト：

- 表 7.1 に従ったグラフィック書式オプション：*gstate*
- 表 8.5 に従った変換オプション：*transform*

戻り値 パターンハンドル。以後の *PDF_setcolor( )* への呼び出しで、また、オプション *fillcolor*・*strokecolor* のために、**文書**スコープが終わるまで使えます。

詳細 オブジェクトにシェーディングを適用するには以下の手順が必要です：

- *PDF_shading( )* を使ってシェーディングハンドルを取得する必要があります。
- このシェーディングにもとづいて *PDF_shading_pattern( )* を使ってシェーディングパターンを定義する必要があります。
- このパターンハンドルを *PDF_setcolor( )* に、またはオプション *fillcolor*・*strokecolor* に与えれば、カレントカラーをシェーディングパターンに設定することができます。

スコープ **オブジェクト**以外任意

表 8.5 PDF_shading_pattern( ) の追加オプション

オプション	説明
transform	（オプションリスト）シェーディングパターン座標系を、このシェーディングパターンが使用されるターゲットページまたはテンプレートまたはグリフ記述のデフォルト座標系へマップする変換を、定義するリスト。シェーディングパターン行列をターゲットページまたはテンプレートまたはグリフ記述のそれと連結することによって形成される座標系において、このシェーディングパターンが解釈されます。 このリストは、表 9.12 に従ったキーワード 1 個と float リスト 1 個とのペア群を内容とします。

C++ Java C# ***void shfill(int shading)***

Perl PHP ***shfill(int shading)***

C ***void PDF_shfill(PDF *p, int shading)***

領域をシェーディングで塗ります。

shading *PDF_shading( )* によって返されたシェーディングハンドル。

詳細 このメソッドを使えば、*PDF_shading_pattern( )*・*PDF_setcolor( )* またはオプション *fillcolor*・*strokecolor* を使用せずに、シェーディングを使うことができます。ただしこれは、塗るべ

きオブジェクトの形がシェーディング自体の形と同じという、単純な形に対してのみ動作します。カレント切り抜き領域がシェーディングで塗られるので（シェーディングの *extendo・extend1* オプションに従って）、一般にこのメソッドは、*PDF_clip()* と組み合わせて使います。

スコープ **ページ・パターン**（*painttype=colored* の場合のみ）**・テンプレート・グリフ**（グリフの *colorized* オプションが *true* の場合のみ）

8.6 タイリングパターン

C++ Java C# ***int begin_pattern_ext(double width, double height, string optlist)***

Perl PHP ***int begin_pattern_ext(float width, float height, string optlist)***

C ***int PDF_begin_pattern_ext(PDF *p, double width, double height, const char *optlist)***

タイリングパターンの定義を、オプション群を用いて開始します。

width・height そのパターンの外接枠の寸法をパターン座標系で表したもの。

optlist パターンの詳細を記述したオプションリスト：

- 表 8.6 に従ったパターン独自オプション：
  painttype・tilingtype・xstep・ystep
- 表 9.9 に従ったテンプレートオプション：***boundingbox・topdown***
- 表9.11に従った共通XObjectオプション（ただしXObjectは関与しません）：***defaultcmyk・defaultgray・defaultrgb・transform***

戻り値 パターンハンドル。このハンドルは、これを囲う**文書**スコープの中において、以後の *PDF_setcolor()* への呼び出しで、および、オプション ***fillcolor・strokecolor*** に対して、使うことができます。

詳細 このメソッドはタイリングパターンの定義を開始します。この関数は、すべてのテキスト・グラフィック・カラーステートパラメーター群を、そのデフォルト値へリセットします。その ***transform*** オプションは、パターン座標系から、そのパターンが使用されるページかテンプレートかグリフ定義の座標系へのマッピングを定義します。

スコープ **オブジェクト**以外任意。このメソッドは、**パターン**スコープを開始させます。マッチする *PDF_end_pattern()* 呼び出しと必ずペアにする必要があります。

表 8.6 PDF_begin_pattern_ext( ) のオプション：いくつかのオプションは PDF_begin_template_ext( ) でも使えます。

オプション	説明
painttype	（キーワード）このパターンが、自身の色指定を内容として持っているか、それともこのパターンが塗りか描線のために使用される際にカレントの塗りまたは描線色を用いて着色されるステンシルとして使われるかを示します（デフォルト：colored）： ***colored*** このパターンは、PDF_setcolor( ) への 1 回ないし複数回の呼び出しを用いて、またはオプション fillcolor/strokecolor を用いて、着色される。そのパターン記述は、画像か PDF ページかグラフィックを配置することができます。 ***uncolored*** このパターンは、色指定を一切内容として持っていない。このパターンが塗りか描線のために使用される際には、カレントの塗りまたは描線色が適用されます。画像マスクを使うこともできますが、画像、配置された PDF ページ、グラフィックは一切使えません。このパターンを使用する前には、PDF_setcolor( ) かオプション fillcolor/strokecolor を呼び出して、パターンに基づかない色空間を持ったカレントカラーを設定する必要があります。

表 8.6 PDF_begin_pattern_ext( ) のオプション：いくつかのオプションは PDF_begin_template_ext( ) でも使えます。

オプション	説明
tilingtype	（キーワード）パターンタイル群のスペーシングへの調整を制御します（デフォルト：constantspacing）： ***constantspacing*** パターンセル群を、一貫性をもって、すなわち 1 つのデバイスピクセルの倍数によってスペーシング。PDF 消費ソフトウェアは、xstep・ystep・変換行列に微細な調整を行うことによってパターンセルをわずかに変形させなければならない可能性があります。 ***nodistortion*** パターンセルを歪めない。しかし、このパターンが描かれる際に、パターンセル群の間のスペーシングが、最大 1 デバイスピクセルだけ、縦横ともに変動する可能性があります。これは平均として、求められるスペーシングを実現しますが、個別のパターンセルについてはこの限りではありません。 ***fastertiling*** パターンセル群を、constanttiling と同様に一貫性を持ってスペーシングするが、ただし、より効率的な実装を実現するためにさらなる変形を許容する。
xstep	（0 以外の float。0 に近い値は避けるべきです）パターンセル群の間の横スペーシングをパターン座標で表したもの。デフォルト：width
ystep	（0 以外の float。0 に近い値は避けるべきです）パターンセル群の間の縦スペーシングをパターン座標で表したもの。デフォルト：height

C++ Java C# ***void end_pattern( )***

Perl PHP ***end_pattern( )***

C ***void PDF_end_pattern(PDF *p)***

タイリングパターンの定義を完了します。

スコープ **パターン**。このメソッドは**パターン**スコープを終了させます。対応する*PDF_begin_pattern_ext( )* と必ずペアにして呼び出す必要があります。

9 画像・SVG・テンプレートメソッド

この章の API メソッド：

- *PDF_load_image()*
- *PDF_close_image()*
- *PDF_fit_image()*
- *PDF_info_image()*
- *PDF_load_graphics()*
- *PDF_fit_graphics()*
- *PDF_info_graphics()*
- *PDF_begin_template_ext()*
- *PDF_end_template_ext()*

9.1 画像

C++ Java C# ***int load_image(String imagetype, String filename, String optlist)***

Perl PHP ***int load_image(string imagetype, string filename, string optlist)***

C ***int PDF_load_image(PDF *p,***
const char *imagetype, const char *filename, int len, const char *optlist)

ディスクベースか仮想の画像ファイルを、さまざまなオプションに従って開きます。

imagetype 文字列 *auto* にすると、PDFlib に画像ファイル形式を自動検出するよう指示します（raw 画像に対しては、必ず正しい画像種別を用いて読み込む必要がありますので、不可能です）。画像形式を明示的に文字列 *bmp*・*jbig2*・*jpeg*・*jpeg2000*（PDF 1.5 以上）・*png*・*raw*・*tiff* のいずれかで指定すると、若干速度が向上します。各画像形式の詳細は PDFlib チュートリアルで説明しています。

filename （名前文字列。グローバル *filenamehandling* オプションに従って解釈されます。表 2.1 参照）開きたい画像ファイルの名前。これは、ローカルファイルか仮想ファイルの名前にする必要があります。PDFlib は、URL からは画像データを取り寄せません。

指定したファイル名のファイルが見つからず、かつ *imagetype=auto* にしているときは、PDFlib は自動的に適切なファイル名拡張子を決定しようとします。PDFlib は、与えられた *filename* に以下の一覧の拡張子をすべてつけてみて（小文字と大文字の両方で）、検索パスで指定しているディレクトリの中にその名前のファイルを見つけようとします：

`.bmp`・`.jbig2`・`.jb2`・`.jpg`・`.jpeg`・`.jpx`・`.jp2`・`.jpf`・`.j2k`・`.png`・`.raw`・`.tif`・`.tiff`

len （C 言語バインディングのみ）*filename* の長さ（バイト単位）。*len=0* にすると null 終了文字列を与える必要があります。

optlist 画像関連の特性を表 9.1 に従って指定したオプションリスト。以下のオプションが使えます：

- 一般オプション：*errorpolicy*（表 1.5 参照）

- 色関連オプション：*chromakey*・*colorize*・*decode*・*honoriccprofile*・*iccprofile*・*invert*・*renderingintent*
- クリッピング・マスク・等価オプション：*alphachannelname*・*clippingpathname*・*honorclippingpath*・*ignoremask*・*mask*・*masked*
- 特殊な PDF 機能群：*interpolate*・*templateoptions*
- 共通 XObject オプションを、生成された画像 XObject に適用できます（表 9.11 参照）：*associatedfiles*・*georeference*・*layer*・*metadata*・*pdfvt*
- PDF 出力を書かずに画像を分析するためのオプション：*infomode*
- 画像データを処理するためのオプション：*ignoreorientation*・*page*・*passthrough*
- 表 9.2 に従った、JBIG2・raw 画像のためのオプション：*bpc*・*components*・*copyglobals*・*height*・*imagehandle*・*inline*・*width*
- C の場合と、Perl・PHP・Ruby で *stringformat=legacy* の場合のためのエンコーディングオプション：*hypertextencoding*（表 2.1 参照）

戻り値 画像ハンドル（または *templateoptions* を与えた場合にはテンプレートハンドル）。以後の画像関連の呼び出しで使えます。*errorpolicy=return* の場合、戻り値 -1（PHP では 0）はエラーを示すので、そうでないかを呼び出し側で検査する必要があります。返された画像ハンドルは、複数の PDF 文書にわたって再利用することはできません。メソッドの呼び出しが失敗したときは、その失敗の原因を *PDF_get_errmsg()* で取得することができます。

詳細 このメソッドは、*imagetype* 引数で決定される、対応形式のいずれかのラスタグラフィックファイルを開いて分析し、その画像データを出力文書へコピーします。このメソッドは、出力上にいかなる視覚効果をも与えません。取り込んだ画像を、生成する出力文書のどこかに実際に配置するには、*PDF_fit_image()* を使う必要があります。同じ画像を、1 つの生成文書内で複数回開くことは推奨しません。なぜなら、画像データ本体が出力文書へいくつもコピーされてしまうからです。アプリケーションがこの状況を回避できない場合には、*PDF_begin_document()* の *optimize* オプションを用いて冗長な画像データを除去することもできます。

PDFlib は、与えられた *filename* の画像ファイルを開き、内容を処理して、呼び出しから戻る前にファイルを閉じます。画像は 1 つの文書内に何回でも貼ることができますが（*PDF_fit_image()* を用いて）、画像ファイル自体は、この呼び出しの後も開かれたままになっているのではありません。

PDF/A いくつかのオプションは制約されます。

グレースケール画像は、出力インテント（任意の種類）か *defaultgray* オプションを必要とします。

CMYK 画像は、ICC プロファイルか CMYK 出力インテントか *defaultcmyk* オプションを必要とします。

PDF/A-2/3：JPEG 2000 画像は、一定の条件を満たす必要があります。詳しくは PDFlib チュートリアルを参照してください。

PDF/VT もし、*PDF_begin_document()* で *usestransparency=false* オプションを指定したのに、取り込んだ画像が透過を含んでいる場合には、この呼び出しは失敗する場合があります。

PDF/X いくつかのオプションは制約されます。
PDF/X-3：JBIG2 画像は許されません。

PDF/X-3/4/5p/5pg：グレースケール画像は、グレースケールか CMYK の出力インテント、または *defaultgray* オプションを必要とします。

CMYK 画像は、ICC プロファイルか CMYK 出力インテントか *defaultcmyk* オプションを必要とします。

JPEG 2000 画像は、一定の条件を満たす必要があります。詳しくは PDFlib チュートリアルを参照してください。

PDF/X-5n：グレースケール画像は、出力インテントがインキ *Black* を含んでいるか、あるいは *defaultgray* オプションが設定されている場合にのみ使用できます。

CMYK カラーは、出力インテントがインキ *Cyan*・*Magenta*・*Yellow*・*Black* をすべて含んでいるか、あるいは *defaultcmyk* オプションが設定されている場合にのみ使用できます。

スコープ **オブジェクト**以外任意。*inline=true* を与える場合には、このメソッドはページパターン・テンプレート・グリフスコープ内でのみ呼び出せます。対応する *PDF_close_image()* とペアにして呼び出すべきです。

表 9.1 PDF_load_image( ) のオプション

オプション	説明
alphachannel-name	（名前文字列。TIFF 画像では不可。ignoremask=true にしている場合には無視されます）指定した名前のアルファチャンネルを画像ファイルから読み取り、それをソフトマスクとして画像に適用します。その名前のチャンネルが画像ファイル内に存在している必要があります。デフォルト：画像内の最初のアルファチャンネル
chromakey	（画像色空間の要素数を n としたとき、範囲 0 ～ $2^{\text{要素あたりビット数}}-1$ の n 個または 2×n 個の整数のリスト。強制的に ignoremask=true になります。mask を指定している場合には使用不可。imagetype=jpeg・jpeg2000 では不可。Lab 画像では無視されます）単色または色範囲に対して有効化させるクロマキーマスキング（カラーキーマスキングともいいます）。リスト内の各ペアは、色要素範囲の境界値を含む下限と上限を内容とします。ピクセル群のうち、そのすべての色要素（decode 値群と invert オプションによる色反転を適用する前の段階における）がこの指定した範囲に収まるものが透過の扱いになり、つまり、描画されずに背景が見えます。 n 個のペアでなく n 個の値を指定した場合には、各要素範囲は単色値のみを内容とすることになり、すなわち、各リスト値の記述は色要素の下限と上限を兼ねることになります。 このオプションは PNG 画像の中で見つかったクロマキー値をオーバーライドします。
clipping-pathname	（文字列。imagetype=tiff・jpeg でのみ可。honorclippingpath=false にしている場合には無視されます）指定した名前のパスを画像ファイルから読み込んで、それをクリッピングパスとして使います。その名前のパスが画像ファイル内に存在している必要があります。特別な名前 Work Path を使うと、Photoshop で作成された一時パスを指定することができます。デフォルト：画像ファイル内でクリッピングパスとして与えられているパスの名前
colorize	（スポットまたは DeviceN カラーハンドル、あるいはスポットカラーを定義する色オプション。このオプションは、iccprofile オプションを与えている場合には無視されます）スポットまたは DeviceN カラーを用いて画像に着色します。スポットカラーを用いて着色できる画像は、白黒かグレースケールの画像のみです。スポットカラーを用いて画像に着色するとい、カラー値の極性解釈が反転します。すなわち、カラー値 0 は最大量のスポットカラーになります。この効果を補償するには invert オプションを用います。 DeviceN カラーを用いで着色できる画像は、raw 画像のみです。画像要素の数が、その DeviceN 色空間の中の色要素の数 N に合致している必要があります。DeviceN 色空間はつねに減法です。すなわち、カラー値 0 は白になります。

表 9.1 PDF_load_image( ) のオプション

オプション	説明
decode	（画像色空間の要素数を n としたとき、2×n 個の float かパーセント値のリスト。JPEG 2000 画像では、または mask を指定している場合には、使用不可）画像サンプル値を、その画像の色空間の中の値へマップする数値の、n 個のペアのリスト。各ペアは、最小・最大要素値 0 と (2^ 要素あたりビット数)-1 がマップされるターゲットカラー値を記述します。中間値は線形に内挿されます。 パレット画像に対しては、デコードペアを 1 個だけ指定する必要があり、ベース色空間に対する複数のペアを指定してはいけません（たとえば RGB パレットを持った画像に対し、ペアを 1 個）。このデコード値は、パレット項目ではなくパレット番号に対して適用されますので、このオプションをパレット画像に対して適用するべきではありません。
honor-clippingpath	（論理値。imagetype=tiff・jpeg でのみ可。インライン画像に対しては強制的に false になります）画像ファイルから、もしあればクリッピングパスを読み込んで、それを画像に適用します。デフォルト：true
honor-iccprofile	（論理値。imagetype=jpeg・jpeg2000・png・tiff でのみ可。colorize オプションを指定している場合には、または埋め込みプロファイルが PDF/X-4/5 の CMYK 出力インテントと同一である場合には、強制的に false になります）画像内で ICC プロファイルが得られる場合には、それが埋め込みによって直接的にであれ、間接的に（例：Exif マーカー内の参照を通じて）であれ、それに従い、そしてそれをその画像に適用します。デフォルト：true
iccprofile	（ICC ハンドルかキーワード）画像に適用される ICC プロファイルのハンドル。以下のキーワードが使えます： ***none*** ICC プロファイルを適用しない。 ***srgb*** sRGB プロファイルを適用する。 デフォルト：プロファイルが画像内に存在し、かつ honoriccprofile=true にしているときは、その埋め込まれたプロファイル（またはその画像ファイル内の等価な Exif 情報）。埋め込まれている、またはユーザーが指定した ICC プロファイルが得られない場合には、BMP・JPG・PNG・TIFF 形式の RGB 画像に対して、sRGB プロファイルが適用されます。
ignoremask	（論理値。PDF/X-1/3・PDF/A-1 では、内部アルファチャンネルを持つ画像では true に設定する必要があります）画像内の透過情報とアルファチャンネルを無視します。デフォルト：false
ignore-orientation	（論理値。imagetype=tiff・jpeg でのみ可）画像内の向き情報をすべて無視します。これは、画像データ内の誤った向き情報を補正したいときに有用です。デフォルト：false
infomode	（論理値）true にすると、画像は読み込まれますが、そのピクセルデータは一切出力へ書き出されません。画像特性群を PDF_info_image( ) を用いてクエリーすることはできますが、画像を PDF_fit_image( ) やその他のメソッドを用いてページ上に配置することはできません。このオプションは、画像を、PDF 出力に副作用を一切ほさずにチェックするために有用でしょう。false にすると、そのピクセルデータは PDF 出力へただちに書きだされます。デフォルト：false
interpolate	（論理値。PDF/A では false にする必要があります）画像の内挿を有効にして、画面や紙の上での見ばえを向上させます。これは、Type 3 フォント内のグリフ記述のためのビットマップ画像の場合に有用です。デフォルト：false
invert	（論理値。imagetype=jpeg2000 では mask=true にしていない場合には不可）画像を反転させます（明るい色と暗い色を入れ換え）。デフォルト：false
mask	（論理値。ビットマップ画像、すなわちピクセルあたり 1 ビットに対してのみ可。強制的に ignoremask=true になります）その画像をステンシルマスクとして使用するために用意します。もしこの画像自体がページ上に配置された場合には、黒ピクセルはカレント塗り色で描画され、一方白ピクセルは無視され、すなわち背景が変化しないままとなります。もしこの画像が、masked オプションを用いて別のベース画像をマスクするために使用された場合には、そのベース画像の内容はマスクが黒の所で見え、マスクが白の所では無視されます。

表 9.1 PDF_load_image( ) のオプション

オプション	説明
masked	（画像ハンドル。PDF/A-1・PDF/X-1/3：1 ビットマスクとともにのみ可。もしもその画像がアルファチャンネルを内容に含んでおり、かつ ignoremask=false の場合には無視されます）カレント画像に対するアルファチャンネルとして適用されるグレースケール画像へのハンドル。PDF/A-1・PDF/X-1/3 では、ビットマップマスクが mask オプションを用いて読み込まれている必要があります。マスク画像を decode オプションを付けて読み込む場合、そのデコード配列は {0 1} か {1 0} である必要があります。
page	（整数。imagetype=gif・jbig2・tiff でのみ可。それ以外の形式で使う場合には 1 にする必要があります）複数ページ画像ファイルから、指定番号の画像を取り出します。最初の画像の番号は 1 です。要求したページが画像ファイル内に見つからないときは、その呼び出しは失敗します。デフォルト：1
passthrough	（論理値。imagetype=tiff・jpeg でのみ可）画像データの処理を制御します。 TIFF 画像（デフォルト：true）：true にすると、圧縮された TIFF 画像データは、可能ならば PDF 出力へ直接パススルーされます。このオプションを false に設定すると、TIFF 画像が損傷したデータや不完全なデータを含むときに役立つ場合があります。 JPEG 画像（デフォルト：false）：false にすると、PDFlib は、PDF との互換性のために JPEG 画像データをトランスコードします（ピクセルデータを解凍せずに）。true にすると、JPEG 画像データは PDF 出力へ直接コピーされます。マルチスキャンおよびある種の CMYK JPEG 画像では、このオプションは無視されます。このオプションを true に設定すると、処理は若干速くなることがありますが、ある種のまれな種類の JPEG 画像が Acrobat 上で正しく表示されなくなります。
rendering-intent	（キーワード）カラーレンダリングインテント（デフォルト：Auto）： Auto・AbsoluteColorimetric・RelativeColorimetric・Saturation・Perceptual
template-options	（オプションリスト）与えられたオプションリスト（空にすることもできます）に従ってテンプレート（PDF フォーム XObject）を生成します。これは、輝度ソフトマスクや、画像 1 個だけを内容とするフォームフィールドアイコンのためのテンプレートを生成するために役立つでしょう。生成されたテンプレートへのハンドルが返されます。このオプションリストは、表 9.11 に従った共通 XObject オプション群を内容とします。

表 9.2 PDF_load_image( ) で imagetype=jbig2 か jpeg か raw の場合の追加オプション

オプション	説明
bpc	（整数。imagetype=raw でのみ可。その場合は必須）色要素あたりのビット数。1・2・4・8・16 のいずれかにする必要があります（PDF 1.4 では値 16 は許容されません）
components	（整数。imagetype=raw でのみ可。その場合は、colorize オプションを与えていない限りは必須）画像要素（チャンネル）の数。1・3・4 のいずれかにする必要があります。colorize オプションが与えられている場合には、要素の数は色空間から決定されますので、このオプションは無視されます。
copyglobals	（キーワード。imagetype=jbig2 でのみ可）JBIG2 ストリーム内のどのグローバルセグメントが PDF へ複製されるかを指定します。JBIG2 ストリーム内にグローバルセグメントが全然ないときは、このオプションは全く効力を持ちません（デフォルト：current）： ***all*** JBIG2 ストリーム内のすべてのページに対するグローバルセグメントを PDF へ複製します。これは、同じ JBIG2 ストリームから複数のページが取り込まれる場合には、用いる必要があります。同じ JBIG2 ストリームからさらなるページ群が後で取り込まれる場合には、imagehandle オプションを用いる必要があります。 ***current*** JBIG2 ストリーム内のカレントページ（すなわち、page オプションで指定したページ）に対して必要なグローバルセグメントのみを PDF へ複製します。これは、同じ JBIG ストリーム内からさらなるページが取り込まれない場合には、用いる必要があります。

表 9.2 PDF_load_image( ) で imagetype=jbig2 か jpeg か raw の場合の追加オプション

オプション	説明
height	（整数。imagetype=raw でのみ可。その場合は必須）画像の高さを、ピクセル単位で指定します。
imagehandle	（画像ハンドル。imagetype=jbig2 でのみ可）同じ JBIG2 ストリームから作成された別の画像に付けられた既存のグローバルセグメントへの参照を追加します。この画像は、以前に copyglobals=all オプションで読み込まれている必要があります。カレント JBIG2 ストリーム以外のファイルから作成されている画像を参照することはエラーです。指定した画像ハンドルは、閉じられていてはいけません。デフォルト：画像ハンドルなし、すなわち、新規 PDF オブジェクトが、カレントページのみに対するすべての必要なグローバルセグメントを持って生成されます
inline	（論理値。imagetype=jpeg・raw でのみ可。templateoptions を与えている場合には使用不可）true にすると、画像は直接、ページ・パターン・テンプレート・グリフのいずれかの記述の内容ストリーム内へ書き込まれます。このオプションは暗黙的に、PDF_fit_image( ) と PDF_close_image( ) を呼び出します（PDFlib チュートリアル参照）。このオプションは、Type 3 フォントのビットマップグリフに対してのみ推奨されるものであり、それ以外の場面では用いるべきではありません。このオプションを与えた場合には *PDF_close_image( )* を呼び出してはいけません。デフォルト：false
width	（整数。imagetype=raw のみ。その場合は必須）画像の幅を、ピクセル単位で指定します

C++ Java C# ***void close_image(int image)***

Perl PHP ***close_image(int image)***

C ***void PDF_close_image(PDF *p, int image)***

画像かテンプレートを閉じます。

image *PDF_load_image( )* か *PDF_begin_template_ext( )* で取得した有効な画像かテンプレートハンドル。

詳細 このメソッドは、PDFlib 内部の関連する画像構造に対してのみ効力を持ちます。画像をファイルから開いていたとしても、画像ファイル本体は、その *PDF_load_image( )* への呼び出しが完了した時点ですでに閉じられているため、この呼び出しには影響されません。画像ハンドルは、このメソッドで閉じた後はもう使えません。

スコープ **オブジェクト**以外任意。対応する *PDF_load_image( )*（*inline* オプションが用いられている場合を除き）か *PDF_begin_template_ext( )* と必ずペアにして呼び出す必要があります。

C++ Java C# ***void fit_image(int image, double x, double y, String optlist)***

Perl PHP ***fit_image(int image, float x, float y, string optlist)***

C ***void PDF_fit_image(PDF *p, int image, double x, double y, const char *optlist)***

画像またはテンプレートをページ上に、さまざまなオプションに従って配置します。

image *PDF_load_image( )* または *PDF_begin_template_ext( )* で取得した有効な画像またはテンプレートハンドル。この画像は、*infomode=true* を用いて読み込まれていてはいけません。テンプレートハンドルは、その定義が *PDF_end_template_ext( )* を用いて完了されている場合にのみ使用できます。

x・y 画像またはテンプレートを、さまざまなオプションに従って貼りたい参照点の座標を、ユーザー座標系で指定します。

optlist 画像はめ込み・処理オプションを指定したプションリスト。以下のオプションが使えます：

- 表 6.1 に従ったはめ込みオプション：
  boxsize・blind・dpi・fitmethod・matchbox・orientate・position・rotate・scale・showborder
- 表 9.3 に従った、画像処理のためのオプション：
  adjustpage・gstate・ignoreclippingpath・ignoreorientation
- 表 14.4 に従った、短縮構造エレメントタグ付けのためのオプション（**ページ**スコープでのみ可）：*tag*

詳細 画像またはテンプレート（以下、まとめてオブジェクトといいます）は、参照点 *(x, y)* に合わせて配置されます。デフォルトでは、オブジェクトの左下隅が参照点に配置されます。しかしこの動作は、*orientate・boxsize・position・fitmethod* オプションで変えることもできます。デフォルトでは、画像はその解像度の値に従って拡縮されます。この動作は、*dpi・scale・fitmethod* オプションで変えることもできます。

PDF/UA 画像は、*Artifact* か *Figure* としてタグ付けされる必要があります。

スコープ **ページ・パターン**（パターンの *painttype* が *colored* であるか、または画像がマスクである場合のみ）**・テンプレート・グリフ**（グリフの *colorized* オプションが *true* であるか、または画像がマスクの場合のみ）。このメソッドは、画像ハンドルを *PDF_close_image()* で閉じない限り、任意のページ上で任意の回数呼び出すことができます。.

表 9.3 PDF_fit_image( )・PDF_fit_graphics( )・PDF_fit_pdi_page( )・PDF_fill_imageblock( )・PDF_fill_graphicsblock( )・PDF_fill_pdfblock( ) を用いた画像・グラフィック・PDF ページ・テンプレート処理のためのオプション

オプション	説明
adjustpage	（論理値。ページスコープでのみ効力を持ちます。PDF_begin_page_ext( ) で topdown オプションを与えている場合には不可。PDF_fill_*block( ) では不可）カレントページの寸法を、オブジェクトに合わせて調整し、ページの右上隅がオブジェクトの右上隅プラス (x, y) と一致するようにします。ここで x・y はメソッドの引数です。MediaBox が調整され、それ以外の枠項目はすべてデフォルトにリセットされます。position オプションを値 0 にして、以下のような使い方ができます： x ≧ 0 かつ y ≧ 0 オブジェクトは余白で囲われます。この余白は、横方向の厚さが y、縦方向の厚さが x になります。 x < 0 かつ y < 0 画像から横と縦の帯が切り取られます。 デフォルト：false
gstate	（PDF_create_gstate( ) に対するオプションのオプションリスト、またはグラフィックステートハンドル）グラフィックステートオプション群またはハンドル。このグラフィックステートは、このメソッドで作成されるすべてのグラフィック要素に対して効力を持ちます。デフォルト：グラフィックステートなし、すなわち、カレントステートが用いられます
ignore-clippingpath	（論理値。TIFF・JPEG 画像でのみ可）画像ファイルの中にクリッピングパスがあっても無視されます。デフォルト：false、すなわちクリッピングパスは適用されます
ignore-orientation	（論理値。TIFF・JPEG 画像でのみ可）画像内の向き情報をすべて無視します。これは、誤った向き情報を補正したいときに有用です。デフォルト：PDF_load_image( ) の ignoreorientation オプションの値

C++ Java C# **double info_image(int image, String keyword, String optlist)**

Perl PHP **float info_image(int image, string keyword, string optlist)**

C **double PDF_info_image(PDF *p, int image, const char *keyword, const char *optlist)**

画像かテンプレートを組版する際の、寸法などの特性群を取得します。

image 関数 *PDF_load_image()* か *PDF_begin_template_ext()* で取得した有効な画像かテンプレートハンドル。

keyword 求める情報を指定したキーワード：

- 表 6.3 に従った、オブジェクトはめ込みの結果をクエリーするためのキーワード：
  boundingbox・*fitscalex*・*fitscaley*・*height*・*objectheight*・*objectwidth*・*width*・*x1*・*y1*・*x2*・*y2*・*x3*・*y3*・*x4*・*y4*
- 表 9.4 に従ったさらなるキーワード：
  clippingpath・*checkcolorspace*・*filename*・*iccprofile*・*imageheight*・*imagemask*・*imagetype*・*imagewidth*・*infomode*・*mirroringx*・*mirroringy*・*orientation*・*resx*・*resy*・*strips*・*transparent*・*xid*

optlist 以下のオプションを使えます：

- *PDF_fit_image()* に対するオプション群。求められたキーワードの値を決定するのに関係のないオプションは無視されます。
- 基礎をなす画像とテンプレートの間を切り替えるためのオプション：
  useembeddedimage

戻り値 キーワードで要求した何らかの画像特性の値。要求した特性が画像ファイル内で得られないときは、このメソッドは 0 を返します。オブジェクトハンドルを要求したときは（*clippingpath* 等）、このメソッドはそのオブジェクトのハンドルを返すか、あるいはそのオブジェクトが得られないときは -1（PHP では 0）を返します。要求されたキーワードがテキストを生み出す場合には、文字列番号が返され、対応する文字列を、*PDF_get_string()* を用いて取得する必要があります。

詳細 このメソッドは、与えたオプション群に従って画像を配置するために必要なすべての計算を実行しますが、ページ上には実際の出力を一切生成しません。画像の参照点は *{o o}* と見なされます。

スコープ **オブジェクト**以外任意

表 9.4 PDF_info_image( ) のキーワード

キーワード	説明
clippingpath	画像のクリッピングパスのパスハンドルか、あるいはクリッピングパスが存在しないときは -1（PHP では 0）
checkcolorspace	その画像かテンプレートが、カレントページ上に、PDF/A か PDF/X の色関係の違反のリスクを冒さずに安全に配置できるなら 1、そうでないなら 0。このチェックは、そのページのデフォルト色空間を考慮に入れます。このデフォルト色空間は、その画像を読み込む際にはチェックされません。
filename	画像ファイル（あてはまる場合は searchpath ディレクトリを含む）の名前に対する文字列番号、あるいはテンプレートの場合には -1
iccprofile	画像内に埋め込まれている ICC プロファイルのハンドルか、あるいはプロファイルが全然存在しないときは -1（PHP では 0）

表 9.4 PDF_info_image( ) のキーワード

キーワード	説明
imageheight	画像：高さをピクセル単位で表したもの テンプレート：ユーザーが与えた高さ
imagemask	画像に関連付けられたマスクの画像ハンドルか、あるいはマスクが付けられていないなら −1（PHP では 0）
imagetype	画像の種別（形式）に対する文字列番号： ラスタ画像の場合には bmp・jbig2・jpeg・jpeg2000・png・raw・tiff。与えられたハンドルに関連付けられたオブジェクトが PDF_begin_template_ext( ) を用いて作成されている場合には、文字列 template が返されます。
imagewidth	画像：幅をピクセル単位で表したもの テンプレート：ユーザーが与えた幅
infomode	その画像が infomode オプションを用いて読み込まれている場合には 1、そうでないなら 0
mirroringx・mirroringy	指定したオプション群に従った画像の横反転・縦反転（1 か −1 かで表されます）
orientation	画像の向きの値。orientation タグを含む画像についてはこのタグの値が返され、それ以外の場合にはすべて −1 が返されます。PDFlib は、1 以外の向きの値を自動的に補償します。
resx・resy	画像の横解像度・縦解像度。正の値は、画像の解像度をインチあたりピクセル数（dpi）で表します。値 0 は、解像度が未知であることを意味します。負の値は、非正方ピクセルの縦横比を決定するために縦横相伴って用いられることがありますが、絶対的な意味は一切持ちません。
strips	画像ページの数（ある種の複数ページ TIFF 画像の場合のみ 1 以外になることがあります）
transparent	画像が透過（＞ 1 ビットのアルファチャンネル）を含んでいるなら 1、そうでないなら 0。 アルファチャンネルが元画像ファイルから読み込まれた場合、または画像が mask オプションを用いて読み込まれている場合には、透過が存在すると見なされます。
xid	（PDF/VT のみ）画像かテンプレートの GTS_XID エントリーに対する文字列番号、あるいは GTS_XID 値が割り当てられていない場合には −1。この文字列は、DPI のための CIP4/Summary/Content/Referenced メタデータプロパティで使用できます。

表 9.5 PDF_info_image( ) のオプション

オプション	説明
useembedded-image	（論理値。templateoptions を与えている場合にのみ意味を持ちます）true にすると、テンプレート内に埋め込まれている画像の情報がクエリーされ、そうでないならテンプレートの情報がクエリーされます。useembeddedimage=true の場合には、いくつかのキーワードは、元画像に対してのみ意味を持ち、生成されたテンプレートでは意味を持ちません。とりわけ、これらの値を、その画像のために作成されたテンプレートをはめ込むために使用するべきではありません。デフォルト：false

9.2 SVG グラフィック

C++ Java C# ***int load_graphics(String type, String filename, String optlist)***

Perl PHP ***int load_graphics(string type, string filename, string optlist)***

C ***int PDF_load_graphics(PDF *p, const char *type, const char *filename, int len, const char *optlist)***

さまざまなオプションに従って、ディスクベースか仮想ベクトルグラフィックファイルを開きます。

type ベクトルグラフィックファイルの種別。キーワード ***auto*** は、自動的にファイル種別を決定します。これは、SVG グラフィックを指定する ***svg*** と等価です。

filename （名前文字列。グローバル ***filenamehandling*** オプションに従って解釈されます。表 2.1 参照）開きたいグラフィックファイルの名前。これは、ディスクベースか仮想ファイルの名前である必要があります。PDFlib は、URL からグラフィックを引っ張ってきません。

指定されたファイル名を持つファイルが見つからないときは、PDFlib は、適切なファイル名接尾辞を自動的に決定しようと試みます。これは、以下のリストからすべての接頭辞を（小文字と大文字の両方で）、指定された ***filename*** に付加して、その名前を持つファイルを、検索パスで指定されたディレクトリ群の中で見つけようと試みます：

`.svg`・`.svgz`

len （C 言語バインディングのみ）***filename*** の長さ（バイト単位で）。***len = 0*** の場合は、ヌル終端文字列を与える必要があります。

optlist グラフィック関連特性群を指定したオプションリスト。以下のオプションを使えます：

- 一般オプション：***errorpolicy***（表 1.5 参照）・***hypertextencoding***（表 2.1 参照）
- 表 9.6 に従ったフォント関連オプション：
  defaultfontfamily・***defaultfontoptions***・***fallbackfontfamily***・***fallbackfontoptions***
- 表 9.6 に従ったサイズオプション：
  bboxexpand・***fallbackheight***・***fallbackwidth***・***forcedheight***・***forcedwidth***
- 表 9.6 に従った画像関連オプション：***defaultimageoptions***・***fallbackimage***
- 表 9.6 に従ったその他の SVG 処理オプション：***downloadlifetime***・***errorconditions***・***externalrefs***・***lang***
- 表 9.6 に従った色制御オプション：***devicencolors***・***forcesrgb***・***honoriccprofile***・***iccprofilecmyk***・***iccprofilegray***・***iccprofilergb***
- 表 9.6 に従ったテンプレート関連オプション：***inline***・***templateoptions***
- C の場合と、Perl・PHP・Ruby で ***stringformat=legacy*** の場合のためのエンコーディングオプション：***hypertextencoding***（表 2.1 参照）

戻り値 以後のグラフィック関連呼び出しで使用できるグラフィックハンドル。***errorpolicy= return*** の場合には、戻り値 -1（PHP では 0）がエラーを知らせますので、呼び出し側はこれをチェックする必要があります。このメソッドが**オブジェクト**スコープ内で呼ばれた場合には、このグラフィックハンドルは、複数の PDF 文書にわたって再利用することが可能です。このメソッド呼び出しが失敗した場合には、その失敗の原因を、***PDF_get_errmsg()*** を用いて要求することもできます。

このメソッドが**オブジェクト**スコープ内で呼ばれた場合には、このグラフィックハンドルは、複数の PDF 文書にわたって再利用することが可能です。そうでない場合には、このグラフィックハンドルは、必要に応じて、カレント文書の終了とともに自動的に閉じられます。

詳細 このメソッドは、*type* 引数によって決定された通りの、対応フォーマット群のうちの 1 つのベクトルグラフィックファイルを、開いて分析します。そのグラフィックデータは、このグラフィックが *PDF_close_graphics( )* を用いて閉じられるか、あるいは PDFlib オブジェクトの継続期間の終了まで格納されます。このメソッドは、PDF 出力上に何の視覚的影響も与えません。取り込まれたグラフィックを、生成される文書内のどこかに実際に配置するためには、*PDF_fit_graphics( )* を使う必要があります。同一生成文書上で同一グラフィックを複数回開くことは、そのグラフィックデータがその出力文書へ複数回複製されることになりますので、推奨しません。

PDFlib は、与えられた *filename* を持つグラフィックファイルを開き、その内容を処理して、デフォルトでは、この呼び出しから返る前にこのファイルを閉じます。ただし、*templateoptions* オプションが与えられている場合には、そのファイル内容は、返されたグラフィックハンドルを用いて *PDF_fit_graphics( )* を初めて呼ぶ時までは必要です。

フォント埋め込み（とりわけ PDF/A・PDF/X・PDF/UA の場合に意味を持ちます）：そのグラフィックの中で使われているすべてのフォント（または然るべきデフォルトフォント群）に対するフォントファイルを構成する必要があります。

PDF/A すべてのフォントを埋め込む必要があります。色が *device-gray* か *device-rgb* か *device-cmyk* として指定されている場合には、これらに PDF/A の色空間の必要条件が課せられます（*PDF_set_color( )* 参照）。
PDF/A-1：透過を持つグラフィックは許容されません。
PDF/A-2a/3a：グラフィックが、PUA キャラクターを持つテキストを含む場合には、*ActualText* サブオプションを伴う *tag* オプションを与える必要があります。

PDF/UA すべてのフォントを埋め込む必要があります（上述）。グラフィックが、PUA キャラクターを持つテキストを含む場合には、*ActualText* サブオプションを伴う *tag* オプションを与える必要があります。

PDF/VT *PDF_begin_document( )* で *usestransparency=false* オプションが指定されたにもかかわらず、取り込まれたグラフィックが透過を含んでいる場合、この呼び出しは失敗するおそれがあります。

PDF/X すべてのフォントを埋め込む必要があります。色が *device-gray* か *device-rgb* か *device-cmyk* として指定されている場合には、これらに PDF/X の色空間の必要条件が課せられます（*PDF_set_color( )* 参照）。
PDF/X-3：透過を持つグラフィックは許容されません。

スコープ 任意

表 9.6 PDF_load_graphics( ) のオプション

オプション	説明
bboxexpand	（2 個の float かキーワードのリスト。絶対寸法を持たない SVG グラフィックでのみ有効）算出される外接枠は、線幅と過大なグリフが考慮されていないので、小さすぎる場合があります。このオプションを用いると、算出された外接枠を、x・y 方向に両側へ数値分（デフォルト座標で）拡張できます。あるいは以下のキーワードも使えます（デフォルト：{offset offset}）： ***offset*** SVG 座標系の原点からの外接枠の水平または垂直距離を使用。

表 9.6 PDF_load_graphics( ) のオプション

オプション	説明
defaultfont-family	（名前文字列）そのグラフィックファイルの中のいずれかのテキストに対するフォントが指定されていないか得られないときに用いられるフォントファミリーの名前。デフォルト：Arial Unicode MS が得られるならそれ、得られないなら Helvetica
defaultfont-options	（オプションリスト）表 4.1 に従ったフォント読み込みオプション群。グラフィックファイル内のテキストに対してフォントが必要な際に、このフォントがそれまでにまだ読み込まれていなかった場合には、ここで指定されたオプション群が PDF_load_font( ) に与えられます。デフォルト：空リスト （このメソッドがオブジェクトスコープ内で呼ばれた場合には、すべての場合においてオプション keepfont が追加されます）
defaultimage-options	（オプションリスト）表 9.1 に従った画像読み込みオプション群。埋め込まれた、または外部の画像が処理される際には、ここで指定されたオプション群が PDF_load_image( ) に与えられます。デフォルト：{ }
devicencolors	（DeviceN カラーハンドルのリスト）読み込まれる SVG グラフィックで使用するための DeviceN 色空間群へのハンドル。ここで与えられた DeviceN 色空間群は、この SVG グラフィックの中の device-nchannel 色群に対して、インキの数に従って適用されます。
download-lifetime	（キーワード）自動的に生成される、ダウンロードしたネットワークリソース群を内容とする PVF ファイル群の寿命（デフォルト：graphics）： ***graphics*** PVF ファイル群は PDF_close_graphics( ) で解放されますが、ただし例外として、フォントリソース群は PDF_end_document( ) で解放されます。 ***document*** PVF ファイル群は PDF_end_document( ) で解放されます。グラフィックファイルをオブジェクトスコープで読み込む場合には、このキーワードはキーワード graphics へマップされます。 ***object*** PVF ファイル群は、それを囲うオブジェクトスコープの終了において解放されます。
error-conditions	（オプションリスト）エラーを引き起こす条件のリスト（デフォルト：空）： ***attributes*** （文字列のリスト）指定された SVG 属性群のうちの 1 つが存在し、しかし PDFlib がそれに対応していない場合に、エラーが発生します（非対応属性の一覧については PDFlib チュートリアルを参照）。デフォルトでは、非対応属性は無視されます。 ***elements*** （文字列のリスト）指定された SVG エレメント群のうちの 1 つが存在し、しかし PDFlib がそれに対応していない場合に、エラーが発生します（非対応エレメントの一覧については PDFlib チュートリアルを参照）。デフォルトでは、非対応エレメントは無視されます。 ***references*** （キーワードのリスト）以下の種別の参照のうちの 1 つが解決または実行できない場合にエラーが発生します（デフォルトでは、image 以外のすべての種別が黙って無視され、警告が出力されます）： ***image*** 画像またはグラフィックファイルへの参照。デフォルト動作についてはオプション fallbackimage を参照 ***internal*** SVG エレメントへの内部参照 ***external*** 画像かグラフィック以外のファイルへの参照 ***fontfamily*** フォントファミリーへの参照 ***font*** フォントファミリー・ウェイト・スタイルを通じて指定されたフルフォント名への参照
externalrefs	（論理値。network オプションリストを通じてネットワーキングが有効化されている場合にのみ意味を持ちます）false にすると、外部リソース群への URL 参照群は解決されず、すなわち、ネットワーク活動は開始されません。このオプションは、先頭にプロトコル識別子が付いているリソース参照に対してのみ効力を持ちます。SVG 内で外部参照が見つかったが externalrefs=false である場合には、動作はオプション errorconditions によって制御されます。デフォルト：true

表 9.6 PDF_load_graphics( ) のオプション

オプション	説明
fallback-fontfamily	（名前文字列）各フォントのための予備フォントを作成するために用いられるフォントファミリーの名前。そのグラフィックファイル内で予備フォント群がすでに指定されている場合にはそれに付加されます。デフォルト：空
fallback-fontoptions	（オプションリスト）fallbackfontfamily オプションを通じて作成された予備フォント群に適用されるオプション群。表 4.2 に従った、以下のオプションを使えます：fontsize・forcechars・textrise。デフォルト：空
fallbackheight・*fallback-width*	（float。絶対寸法を持たない SVG グラフィックでのみ有効。forcedheight/forcedwidth が与えられている場合には無視されます）はめ込み処理のための SVG グラフィックの高さ／幅（デフォルト座標で）。どちらかの値がゼロの場合には PDFlib が外接枠を算出します。デフォルト：SVG グラフィック内に値があるならその値、ないなら 0
fallback-image	（オプションリスト）グラフィックか画像要素のはめ込み枠のために確保されているスペースを、その要素が得られないときにどのように視覚化するかを指定します。このオプションを何もサブオプションなしで与えると、グレーの半透明の市松模様が描かれます（デフォルト：代替表示なし）： ***fillcolor*** （sRGB カラーかキーワード）その領域を市松模様で塗るために用いられる色（gridsize > 0 の場合）。この場合、半数の正方形はこの指定した色で描かれ、残り半数の正方形は透明になります。あるいは gridsize=0 の場合には、その領域を無地で塗るために用いられる色。キーワード none とすると、その領域は塗られません。デフォルト：LightGrey ***gridsize*** （float かパーセント値）市松模様の中の正方形の幅を、デフォルト座標で、またはそのはめ込み枠の幅に対するパーセント値として表したもの。パーセント値は、その領域内に整数個の正方形が収まるように丸められます。gridsize=0 とすると、その領域は、市松模様でなく無地で塗られます。デフォルト：10 ***image*** （画像またはテンプレートハンドル）そのはめ込み枠の中へ fitmethod=entire を用いて配置される画像かテンプレート。デフォルト：画像またはテンプレートなし ***opacity*** （範囲 0 ～ 1 の float、またはパーセント値）市松模様の正方形かその領域内部の不透明度。デフォルト：0.5 ***strokecolor*** （sRGB カラーかキーワード）その境界領域とその領域内部の十字を描線するために用いられる色。キーワード none とすると、境界と十字は描線されません。デフォルト：Red
forcedheight・*forcedwidth*	（float ≧ 1）その SVG グラフィックの高さ／幅が無視され、そのかわりに、指定された値（デフォルト座標で）が適用されます。デフォルト：そのグラフィックの height/width 属性
forcesrgb	（論理値）true にすると、非 sRGB 色指定がすべて無視され、かわりに sRGB フォールバックカラーが使用されます。このオプションは参照画像には影響しません。デフォルト：false
honor-iccprofile	（論理値）true にすると、icc-color 指定の中にある、あるいは sRGB カラーの中に暗黙的にある、すべての ICC プロファイルに従います。そうでない場合には、明示的な ICC カラーも、暗黙的な sRGB カラーも、すべてが device-gray/device-rgb/device-cmyk カラーとして解釈されます。PDF/A・PDF/X-3/4/5 では、honoriccprofile=false オプションは、defaultrgb オプションか、RGB 出力インテントか、iccprofilergb オプションを必要とします。デフォルト：true
iccprofilecmyk・*iccprofilegray*・*iccprofilergb*	（ICC ハンドルかキーワード srgb。icc-color(gray/rgb/cmyk) に対する forcesrgb をオーバーライドします。オブジェクトスコープでは使用不可）icc-color 指定の中の明示的なプロファイルのかわりに、または暗黙的な sRGB プロファイルのかわりに使用させたい ICC プロファイル。デフォルト：icc-color 指定の中の ICC プロファイル、およびデフォルト色指定の場合には sRGB

表 9.6 PDF_load_graphics( ) のオプション

オプション	説明
inline	（論理値。templateoptions を与えている場合には無視されます）true にすると、出力を内容ストリーム内にインラインに生成します。そうでない場合にはテンプレート（フォーム XObject）が生成されます。このインライン方式には、透過の処理とファイルサイズに関して難点があります（詳しくは PDFlib チュートリアルを参照）。一方で、このオプションは、グラフィック内のアクティブリンクを変換するべき場合には必要です（PDF_fit_graphics( ) のオプション convertlinks）
lang	（文字列）そのグラフィックファイルに対する自然言語。これはたとえば、SVG の switch エレメントで使用できます。この言語指定の形式は、PDF_begin_document( ) の lang オプションと同じです（表 3.3 参照）。デフォルト：LANG 環境変数で見つかった言語識別子。
template-options	（オプションリスト）この与えられたオプションリストに従ってテンプレート（フォーム XObject）を生成します。 この与えられたオプションリスト（空でもよい）は、PDF_begin_template_ext( ) に対して用いられます。以下の共通 XObject オプション群を使えます（表 9.11 参照）： associatedfiles・defaultgray・defaultrgb・defaultcmyk・iconname・layer・metadata・pdfvt・transparencygroup デフォルト：templateoptions={ }、すなわち、デフォルトオプション群を用いたテンプレートが生成されます。

C++ Java C# ***void close_graphics(int graphics)***

Perl PHP ***close_graphics(int graphics)***

C ***void PDF_close_graphics(PDF *p, int graphics)***

ベクトルグラフィックを閉じます。

graphics *PDF_load_graphics( )* を用いて取得された有効なグラフィックハンドル。

詳細 そのグラフィックに関連している内部構造を削除します。そのグラフィックがファイルから開かれていた場合には、そのグラフィックファイル自体はこの呼び出しによって影響を受けません。なぜならそれはすでに、それに対応する *PDF_load_graphics( )* 呼び出しの終了で閉じられているからです。グラフィックハンドルは、閉じられた後は、もう使うことができません。

スコープ 任意。対応する *PDF_load_graphics( )* への呼び出しでが指定され、かつそのグラフィックが少なくとも 1 回配置された場合には、**オブジェクト**スコープは許容されません。組になる *PDF_load_graphics( )* への呼び出しと必ずペアにする必要があります。

C++ Java C# ***void fit_graphics(int graphics, double x, double y, String optlist)***

Perl PHP ***fit_graphics(int graphics, float x, float y, string optlist)***

C ***void PDF_fit_graphics(PDF *p, int graphics, double x, double y, const char *optlist)***

ベクトルグラフィックを内容ストリーム上に、さまざまなオプションに従って配置します。

graphics *PDF_load_graphics( )* を用いて取得された有効なグラフィックハンドル。

x・y グラフィックが配置される参照点の座標をユーザー座標系で表したもの。

optlist グラフィックはめ込み・処理オプションを指定したオプションリスト。以下のオプションが使えます：

- 表 6.1 に従ったはめ込みオプション：
  boxsize・*fitmethod*・*matchbox*・*orientate*・*position*・*refpoint*・*rotate*・*scale*・*showborder*
- 表 9.3 に従った、グラフィック処理のためのオプション：
  adjustpage・*gstate*
- 表 9.7 に従った、グラフィック内のインタラクティブリンク群を処理するためのオプション：*convertlinks*
- 表 14.4 に従った、短縮構造エレメントタグ付けのためのオプション（**ページ**スコープでのみ可）：*tag*

詳細 グラフィックは、参照点 *(x, y)* を基準として配置されます。デフォルトでは、そのオブジェクトの左下隅が参照点に配置されます。ただし、*orientate*・*boxsize*・*position*・*fitmethod* オプションでこの動作を変更することもできます。デフォルトでは、グラフィックはその内部で指定されたサイズに従って拡縮されます。この動作は、*scale*・*fitmethod* オプションで変更することもできます。

PDF_fit_graphics( ) を呼び出す前に、*PDF_fit_graphics( )* が成功するかどうかをチェックするために（そして失敗して例外が発生することを避けるために）、*PDF_info_graphics( )* で *fittingpossible* キーワードを用いることを推奨します。

PDF/UA グラフィックかラスタ画像を含むグラフィックは、*Figure* か *Artifact* としてタグ付けする必要があります。

スコープ **ページ・パターン**（そのパターンの *painttype* が 1 の場合のみ）・**テンプレート・グリフ**（グリフの *colorized* オプションが *true* の場合のみ）。このメソッドは、そのグラフィックハンドルが *PDF_close_graphics( )* を用いて閉じられない限り、任意の回数、任意のページ上で呼び出すことができます。

表 9.7 PDF_fit_graphics( ) のさらなるオプション

オプション	説明
convertlinks	（論理値。inline オプションを用いて読み込んだグラフィックに対してのみ意味を持ちます）true にすると、グラフィックファイル内のインタラクティブリンクが PDF 内のインタラクティブ Link 注釈へ変換されます。この設定にかかわらず、リンクは以下の状況では作成されません（デフォルト：true）： ▸ このメソッドをページ以外のスコープで呼び出している。 ▸ PDF_load_graphics( ) で templateoptions オプションを与えている。 ▸ タグ付き PDF モード：カレントでアクティブな構造アイテムがページ装飾である。PDF 2.0 の標準ページ装飾群（direct=false を用いて生成されているページ装飾群など）はこの制約の影響を受けません。 ▸ PDF/X：そのリンク注釈が BleedBox（あるいは BleedBox が存在しない場合には TrimBox/ArtBox）内に置かれている。

C++ Java C# ***double info_graphics(int graphics, String keyword, String optlist)***

Perl PHP ***float info_graphics(int graphics, string keyword, string optlist)***

C ***double PDF_info_graphics(PDF *p, int graphics, const char *keyword, const char *optlist)***

ベクトルグラフィックを整形し、メトリックその他特性群をクエリーします。

graphics *PDF_load_graphics( )* を用いて取得された有効なグラフィックまたはテンプレートハンドル。

keyword 求める情報を指定するキーワード：

- 表 6.3 に従った、オブジェクトはめ込みの結果をクエリーするためのキーワード：
  boundingbox・fitscalex・fitscaley・height・objectheight・objectwidth・width・x1・y1・x2・y2・x3・y3・x4・y4
- 表 9.8 に従ったさらなるキーワード：
  description・filename・fittingpossible・graphicswidth・ graphicsheight・istemplate・metadata・title・type・xid

optlist PDF_fit_graphics( ) に対するオプション群を指定したオプションリスト。求められたキーワードの値を決めるのに関係のないオプション群は無視されます。

戻り値 *keyword* によって求められた通りの何らかのグラフィック特性の値。視覚的な特性がページの外で求められたときは、このメソッドは -1（PHP では 0）を返します。オブジェクトハンドルが求められたときは、このメソッドはそのオブジェクトへのハンドルを返しますが、もしそのオブジェクトが得られない場合には -1（PHP では 0）を返します。求められたキーワードがテキストを生み出す場合には、文字列番号が返され、それに対応する文字列を、PDF_get_string( ) を用いて取得する必要があります。

詳細 このメソッドは、与えられたオプション群に従ってそのグラフィックを配置するために必要なすべての計算を行いますが、しかしページ上に実際に出力を作成しません。グラフィック参照点は *{0 0}* であると見なされます。

スコープ 任意

表 9.8 PDF_info_graphics( ) のキーワード

キーワード	説明
description	一番外側の svg エレメントが存在するならばそれの、あるいは一番外側の g エレメントが存在するならばそれの desc エレメントの内容に対する文字列番号、それ以外の場合には -1。この文字列はマークアップを含む場合があります。
filename	そのグラフィックファイルの名前に対する文字列番号（あてはまる場合には検索パスディレクトリを含みます）
fittingpossible	そのグラフィックがカレントのコンテキストで PDF_fit_graphics( ) を用いて配置できるかどうかをチェックします。そのグラフィックが配置できるなら値 1 が返されます。以下のいずれかの理由ではめ込みが失敗する（すなわち PDF_fit_graphics( ) が例外を発生させるであろう）なら値 0 が返されます： ▸ そのグラフィックファイル内の内部的問題。 ▸ カレントの規格の必要条件群との衝突（透過が許されていない、フォントを埋め込む必要があるのにフォントファイルが得られないなど） 0 が返された場合には、その問題の性質を、PDF_get_errmsg( ) を用いてクエリーすることもできます。この結果はカレントのコンテキストに対してのみ有効ですので、このチェックはグラフィックの配置を試みる直前に行うべきです。
graphicswidth・graphicsheight	そのグラフィックの、そのグラフィックファイル内の情報に従った幅と高さを、デフォルト座標系で表したもの。そのグラフィックファイル内で値が得られない場合には 0 が返されます。
istemplate	templateoptions オプションが与えられている場合には 1、そうでないなら 0
metadata	最も外の svg エレメントの metadata エレメントの内容に対する文字列番号、あるいはそのエレメントが存在しない場合には -1。この文字列はマークアップを含む場合があります。
title	最も外の svg エレメントの title エレメントの内容に対する文字列番号、あるいはそのエレメントが存在しない場合には -1。この文字列はマークアップを含む場合があります。

表 9.8 PDF_info_graphics( ) のキーワード

キーワード	説明
type	そのグラフィックの種別（形式）に対する文字列インデックス：つねに svg
xid	（PDF/VT のみ）グラフィックのために作成されたテンプレートの GTS_XID エントリーに対する文字列番号、あるいはテンプレートが作成されなかったか GTS_XID 値がそのテンプレートに割り当てられていない場合には -1。この文字列は、DPM のための CIP4/Summary/Content/Referenced メタデータプロパティ内で使用することができます。

9.3 テンプレート（フォーム XObject）

注「テンプレート」という言葉をここでは PDF フォーム XObject と同義に用います。この節で解説するテンプレートメソッドは、PDFlib ブロックによる可変データ処理とは無関係です。PDFlib Block Plugin で作成したブロックへの流し込みを行うには *PDF_fill_*block()* を使います（235 ページ「11 章 ブロック流し込みメソッド（PPS）」参照）。

C++ Java C# ***int begin_template_ext(double width, double height, String optlist)***

Perl PHP ***int begin_template_ext(float width, float height, string optlist)***

C ***int PDF_begin_template_ext(PDF *p, double width, double height, const char *optlist)***

テンプレート定義を開始します。

width・height テンプレートの外接枠の寸法を、ポイント単位で指定します。この *width・height* 引数は 0 にすることもできます。この場合、それらは *PDF_end_template_ext()* 内で、あるいは *boundingbox* オプションを用いて与える必要があります。結局どちらの値も 0 以外とする必要があります。ただし *watermark* オプションが指定されている場合を除きます。

width または *height* オプションを与えている場合には、対応する *width* または *height* API 引数は無視されます。

optlist テンプレート関連の特性群を指定したオプションリスト。

- 表 9.11 に従った共通 XObject オプション：*associatedfiles・defaultcmyk・defaultgray・defaultrgb・iconname・layer・metadata・pdfvt・transform・transparencygroup*
- 表 9.9 に従ったテンプレートオプション：*boundingbox・height・topdown・watermark・width*

戻り値 以後の *PDF_fit_image()*・*PDF_info_image()* への呼び出しで、およびその他メソッド群のさまざまなオプションで使えるテンプレートハンドル、あるいはエラーの場合には -1（PHP では 0）。

詳細 このメソッドは、テキスト・グラフィック・色ステータスパラメーター群をすべてそれらのデフォルト値にリセットして、*topdown* オプションに従って座標系を定義します。

テンプレートサイズ：最も単純な場合では、幅と高さは *PDF_begin_template_ext()* で与えられます。しかし、もしそれらが未知の場合には、それらを 0 として指定することもできます。この場合にはそれらを、それに対応する *PDF_end_template_ext()* への呼び出しで与える必要があります。

編集できる透かし 廃止された *watermark* オプションが与えられている場合には、このテンプレートは、Acrobat で編集または削除が可能な透かしを定義します。この場合には、以下の内容置換方式のうちの 1 つか、**文書**または**オブジェクト**スコープ内で許されるメソッドのみを使うことができます：

- テキスト行の外接枠が、テンプレートの幅と高さを定義します：*width・height* 引数とオプション *boundingbox・transform* は無視されます。透かしのテキスト行の外接枠は、その回転角と寸法、とりわけオプション *boxheight* から算出されます。透かしに使われるテキスト行では、*boxheight* のデフォルト値が *{capheight 0}* ではなく *{ascender*

descender} になります（表 6.4（144 ページ）参照）。*PDF_fit_textline()* の引数 *x*・*y* は無視されます。

- *PDF_fit_textflow()*：与えられたはめ込み枠が、テンプレートの幅と高さを定義します。はめ込み枠の位置は影響を及ぼしません。オプション *firstlinedist* のデフォルト値は *ascender* です。*lastlinedist* のデフォルト値は *descender* です。テキストフローオプション *fillcolor*・*font*（または *fontname*）・*fontsize*・*underline* については、最初に現れた指定が、Acrobat の透かしメニューにおける対応する設定を決定します。
- *PDF_fit_image()*
- *PDF_fit_pdi_page()*

この透かしテンプレートは、サブオプション *startpage/endpage/pagesubset* で指定されたすべてのページに自動的に配置されます。タグ付き PDF モードではそれはページ装飾としてマークされます。透かしテンプレートを作成して使用する際には、以下の規則に従う必要があります：

- 透かしテンプレートは、それが使用される最初のページより前に作成される必要があります。
- *watermark* オプションは、文書ごとに 1 回しか用いることができません。
- 透かしテンプレートのハンドルを、*PDF_fit_image()* への呼び出しの中で、または画像ハンドルを持つオプションリストの中で（*PDF_add_table_cell()* の中など）、一切用いてはいけません。なぜなら透かしは、オプション *startpage*・*endpage* を通じて指定されたページ範囲に自動的に配置されるからです。

透かしの内容に対しては、以下の色空間のみを使用できます：*gray*、*iccbasedgray*、グレースケールまたは RGB プロファイルを持つ *iccbased*、*rgb*、*iccbasedrgb*

スコープ　**オブジェクト**以外任意。このメソッドは**テンプレート**スコープを開始させます。対応する *PDF_end_template()* と必ずペアにして呼び出す必要があります。

表 9.9 PDF_begin_template_ext( ) のオプション

オプション	説明
boundingbox	（長方形）パターンセルまたはテンプレートの外接枠の左・下・右・上辺の座標。この外接枠は、このパターンセルまたはテンプレートを切り抜くためにも使えますし、このパターンセルまたはテンプレートのまわりに余白を作るためにも使えます。デフォルト：{0 0 width height}
height	（float。boundingbox を与えている場合には無視されます）テンプレートの外接枠の寸法をポイント単位で。width・height API 引数とオプションを PDF_begin_template_ext( ) において 0 にすることはできますが、しかしその場合にはそれらを PDF_end_template_ext( ) か boundingbox オプションで与える必要があります。最終的にどちらの値も 0 以外でなければなりませんが、ただし watermark オプションを与えていればその限りではありません。width・height オプションは、同じ名前の API 引数をオーバーライドします。 以下のシンボリック ISO ページ寸法名の後に .width か .height を付けて（a4.width・a4.height など）キーワードとして使用できます：a0・a1・a2・a3・a4・a5・a6・b5・letter・legal・ledger・11x17
topdown	（論理値）true にすると、ページの開始における座標系の原点はパターンセルまたはテンプレートの左上隅に配置され、y 座標は下向きに増加します。そうでない場合にはデフォルト座標系が用いられます。topdown=true の場合には、テンプレートの高さを PDF_end_template_ext( ) へ先送りすることはできず、PDF_begin_template_ext( ) に対する引数 height かオプション height として与える必要があります。デフォルト：false
watermark	（オプションリスト。文書ごとに 1 回のみ使用可能。廃止。PDF 2.0・PDF/A・PDF/UA・PDF/X・PDF/VT では許されません）表 9.10 のオプションにしたがって、編集可能な透かしを作成します。

表 9.9 PDF_begin_template_ext( ) のオプション

オプション	説明
width	height を参照。

表 9.10 PDF_begin_template_ext( ) の廃止された watermark オプションのサブオプション

オプション	説明
endpage	（整数かキーワード）その透かしを入れたい最後のページの番号、またはキーワード last で末尾ページ。デフォルト：last
fixedprint	（論理値）異なるページ寸法へ印字する際にその透かしの位置と寸法を一定に保ちます。デフォルト：false
horizalign	（キーワード）その透かしの外接枠の、ページ[1] 上における相対横位置を指定する、キーワード left/center/right のうちのいずれか 1 つ。デフォルト：center
horizshift	（float）相対横位置からのずらしをポイント単位で。デフォルト：0
location	（キーワード）その透かしを他のページ内容の後ろに印字するか手前に印字するかを指定します（デフォルト：ontop）： ***behind*** 透かしをページ内容の後ろに印字します。location=behind を与えた場合には、その同一文書内でのその後の PDF_begin_page_ext( ) への呼び出しはすべて、width・height 引数に対して 0 以外の値を与える必要があります。 ***ontop*** 透かしをページ内容の手前に印字します。
onprint	（論理値。PDF 1.5。PDF/A-2/3・PDF/X-4/5・PDF/UA-1 では値 false は不可）true にすると、その透かしはページを印刷する際に印字されます。デフォルト：true
onscreen	（論理値。PDF 1.5。PDF/A-2/3・PDF/X-4/5・PDF/UA-1 では値 false は不可）true にすると、その透かしはページを画面に表示する際に印字されます。デフォルト：true
opacity	（float かパーセント値。PDF/A-1・PDF/X-3 では値 1 にする必要があります）その透かしをページ上で塗るための不透明度値。デフォルト：1
pagessubset	（キーワード）startpage と endpage で指定した範囲の中のページ群の部分集合を選択します（デフォルト：all）： ***all*** その範囲の中のすべてのページを選択します。 ***even*** その範囲の中の偶数番号のページすべてを選択します。 ***odd*** その範囲の中の奇数番号のページすべてを選択します。
scale	（float かパーセント値かキーワード）対象ページに対するその透かしの外接枠の拡縮倍率。拡縮はそのテキスト行の外接枠のアスペクト比を温存します（デフォルト：1）。使えるキーワード： ***none*** 拡縮なし
startpage	（整数）。透かしを入れたい最初のページの番号（先頭ページの番号は 1 です）。次のページ、すなわち、次に PDF_begin_page_ext( ) を呼び出した時に作成されたページの番号が、startpage で指定されたものよりも大きい番号である場合には、この値は次のページへ増加させられます。デフォルト：次のページ
vertalign	（キーワード）その透かしの外接枠の、ページ[1] 上における相対縦位置を指定する、キーワード bottom/center/top のうちのいずれか 1 つ。デフォルト：center
vertshift	（float）相対縦位置からのずらしをポイント単位で。デフォルト：0

1. ページ寸法は、その CropBox があればそれによって、なければその MediaBox によって定義されます。

C++ Java C# ***void end_template_ext(double width, double height)***

Perl PHP ***end_template_ext(float width, float height)***

C ***void PDF_end_template_ext(PDF *p, double width, double height)***

テンプレート定義を完了します。

width・height テンプレートの外接枠の寸法をポイント単位で指定します。*width* か *height* を 0 にすると、*PDF_begin_template_ext( )* で与えた値が用いられます。そうでないときは、*PDF_begin_template_ext( )* の *boundingbox* オプションと *width・height* 引数で与えた値はオーバーライドされます。ただし、対応する *PDF_begin_template_ext( )* への呼び出しで *watermark* オプションが与えられている場合には、*PDF_end_template_ext( )* へ与えられた値は無視されます。

スコープ **テンプレート**。このメソッドは**テンプレート**スコープを終了させます。対応する *PDF_begin_template_ext( )* と必ずペアにして呼び出す必要があります。

9.4 共通 XObject オプション

この節で挙げるオプションは、以下の、フォーム XObject か画像 XObject を作成するメソッドで利用できます：

- *PDF_load_image()* に *templateoptions* オプションを付けた場合
- *PDF_open_pdi_page()*
- *PDF_load_graphics()*（*inline* オプションを与えている場合を除く）
- *PDF_begin_template_ext()*

表 9.11 に従って以下の XObject オプションを使用できます（すべてのオプションを上掲のメソッドで利用できるわけではありません）：
associatedfiles・*defaultcmyk*・*defaultgray*・*defaultrgb*・*georeference*・*iconname*・*layer*・*metadata*・*pdfvt*・*transform*・*transparencygroup*

PDF/A いくつかのオプションが制約されています。

PDF/VT *pdfvt* オプションは PDF/VT に関連します。特定の条件が満たされている場合には、生成される XObject はカプセル化されていると標識されます（詳しくは PDFlib チュートリアルを参照してください）。

PDF/X いくつかのオプションが制約されています。

表 9.11 PDF_load_image( )・PDF_open_pdi_page( )・PDF_load_graphics( )・PDF_begin_template_ext( ) の共通 XObject オプション

オプション	説明
associatedfiles	（アセットハンドルのリスト。PDF 2.0・PDF/A-3 でのみ可）その XObject と連携しているファイル群に対するアセットハンドル群。このファイル群は、PDF_load_asset( ) と type=attachment を用いて読み込まれている必要があります。
defaultgray ***defaultrgb*** ***defaultcmyk***	（ICC ハンドルかキーワード。PDF_load_image( ) では templateoptions を指定している場合にのみ可。PDF_open_pdi_page( ) では不可）与えられた ICC プロファイルハンドルに従って、そのテンプレートに対するデフォルトのグレーか RGB か CMYK の色空間を設定します。オプション defaultrgb ではキーワード srgb も使えます。ここで指定された色空間は、そのテンプレートの中の（ただしネストされたテンプレート群を除く）デバイス依存なグレーか RGB か CMYK カラー群をマップするために使用されます。
georeference	（オプションリスト。PDF 1.7ext3。PDF_load_image( ) で templateoption を指定していない場合にのみ可）地理空間測量に利用するために XObject に関連づけられる地球ベース座標系の記述。詳しくは 290 ページ「13.4 地理空間機能」を参照してください。
iconname	（ハイパーテキスト文字列。PDF_load_image( ) では templateoptions を指定している場合にのみ可）フォーム XObject に名前を付けて、JavaScript で参照できるようにします。たとえば、XObject をフォームフィールドのアイコンとして使いたいときに有用です。
layer	（レイヤーハンドル。PDF 1.5）XObject を属させたいレイヤー。ただし、この XObject を配置する前に PDF_begin_layer( ) で別のレイヤーが有効にされていない場合にかぎります。この XObject を配置する前に PDF_begin_layer( ) を呼び出してレイヤーを有効にさせると、この XObject の layer オプションはオーバーライドされます。この XObject の layer オプションがオーバーライドされないようにするには、この XObject を配置する前に PDF_end_layer( ) を呼び出してください。
metadata	（オプションリスト）XObject に対するメタデータ（307 ページ「15.2 XMP メタデータ」参照）。
pdfvt	（オプションリスト。PDF/VT のみ）表 9.13 に従った、XObject のための PDF/VT サブオプション群。

表 9.11 PDF_load_image( )・PDF_open_pdi_page( )・PDF_load_graphics( )・PDF_begin_template_ext( ) の共通 XObject オプション

オプション	説明
transform	（表 9.12 に従った変換リスト。ページに対して PDF_open_pdi_page( ) の cloneboxes オプションを与えている場合には無視されます。PDF_load_image( ) では不可）そのページまたはパターンまたはテンプレート座標系を、このページかパターンかテンプレートが使用される対象ページまたはテンプレートまたはグリフ記述のデフォルト座標系へマップする変換を定義したリスト。このページまたはパターンまたはテンプレート行列を、対象ページまたはテンプレートまたはグリフ記述のそれと結合して形成される座標系の中で、このページまたはパターンまたはテンプレートの中のすべてのグラフィックオブジェクトが解釈されます。 このリストは、表 9.12 に従ったキーワード 1 個と float リスト 1 個とのペア群を内容とします。ここで各ペアは 1 個の変換を定義します。この float リストの解釈と長さは変換によって異なります。変換群は、その指定された順に適用されます。ペア内の要素は、等号「=」を用いて区切ることができます。デフォルト：変換なし。 例：transform={rotate=45 translate={100 0}}
transparency group	（オプションリストかキーワード。PDF/A-1・PDF/X-1a/3 では不可。PDF/A-2/3・PDF/X-4/5 モードでは PDF_open_pdi_page( )・PDF_load_graphics( ) では強制的に auto になります）フォーム XObject に透過グループを紐付けます。以下のキーワードを使えます（デフォルト：auto）： ***auto*** PDF_open_pdi_page( )・PDF_load_graphics( )：文書オプション usestransparency=false を設定していない限り、配置される内容が出力ページに配置される前と同じ表現になるよう透過グループを生成。PDF_load_image( ) で templateoptions を指定、および PDF_begin_template_ext( )：none と同じ ***none*** そのフォーム XObject に対する透過グループを生成しません。 以下のサブオプションを用いて明示的に透過グループを生成することもできます： ***colorspace***（キーワードか ICC プロファイルハンドル。type=luminosity を指定した PDF_create_gstate( ) の softmask オプションに対してテンプレートが用いられる場合には none 以外の値を指定することが必須）ブレンドする色空間。使えるキーワードと制約については表 3.9（オプション transparencygroup）を参照してください（デフォルト：none）。 ***isolated***（論理値）透過グループが分離されているかどうかを指定します。分離グループの中のすべてのオブジェクトは、透過な初期の背景に対して合成され、そしてその結果が、既存のページ内容に対して合成されます。つまり、グループオブジェクトの合成が、背景から独立になります：そのグループは「分離されて」います。非分離グループ内のオブジェクト群は互いに、そして既存のページ内容に対して個別に合成されます。デフォルト：false ***knockout***（論理値）透過グループがノックアウトグループかどうかを指定します。ノックアウトグループ内のすべてのオブジェクトは、そのグループの中の先行要素とともにではなく、直接に、ページ背景（あるいはそのグループが分離されている場合には透過背景）に対して合成されます。つまり、グループ内のオブジェクトどうしの間には何の相互作用もありません：各オブジェクトは、そのグループの中の、自分より前の、重なるオブジェクトをノックアウトし、そして一番上のオブジェクトがページ背景に対して合成されます。デフォルト：false

表 9.12 PDF_begin_pattern_ext( )・PDF_begin_template_ext( )・PDF_shading_pattern( )・PDF_open_pdi_page( ) の transform オプションのキーワードと float リスト

キーワード	説明
align	方向ベクトル {dx dy} だけ回転。ベクトル {0 0} は回転なしを意味します。
matrix	6 個の値 {a b c d e f} を持つ非縮退変換行列を指定。キーワード current を指定すると、カレント変換行列を構成する 6 個の値が生成されます。カレントユーザー座標系に合致する変換を生成したいときにこれは有用でしょう。

表 9.12 PDF_begin_pattern_ext( ) ・ PDF_begin_template_ext( ) ・ PDF_shading_pattern( ) ・ PDF_open_pdi_page( ) の transform オプションのキーワードと float リスト

キーワード	説明
rotate	{phi} だけ回転。ここで角度 phi は、パターンまたはページまたはテンプレート座標系の x 軸正の向きから反時計回りの度単位で測ります。
scale	{sx sy} だけ拡縮。sy が与えられない場合、それは sx に等しいと見なされます。
skew	{alpha beta} だけ斜形化（シアー）。ここで alpha はパターン座標系の x 軸正の向きから反時計回りの度単位で測り、beta は y 軸正の向きから時計回りで測ります。どちらの角度も、90° の奇数倍にしてはいけません。
translate	{tx ty} だけ平行移動。平行移動はそのページを配置する際に補正されますので、この平行移動は取り込み PDF ページに対しては何の視覚的影響も与えません。

表 9.13 PDF_load_image( ) ・ PDF_open_pdi_page( ) ・ PDF_begin_template_ext( ) と、templateoptions オプションを持つ PDF_load_graphics( ) の、pdfvt オプションのサブオプション

オプション	説明
environment	（ハイパーテキスト文字列。scope=stream か scope=global の場合には必須）PDF/VT 環境コンテキスト。これは一種の識別子であり、PDF/VT プロセッサーはこれを用いて、関連 XObject 群を管理するための管理インターフェイスを提供することができます。たとえば、顧客名やジョブ名を用いて環境を識別できます。
scope	（キーワード）その XObject の PDF/VT スコープ（PDFlib のメソッドのスコープとは関係ありません）（デフォルト：unknown）： ***unknown*** その XObject のスコープは未知です。 ***singleuse*** その XObject はその PDF/VT ファイル内で 1 回だけ参照されます。 ***record*** （PDF_begin_document( ) に対して recordlevel オプションが指定されている場合のみ可）その XObject は、単一のレコードに属するページ群の中で複数回参照されますが、他のレコード内では参照されません。 ***file*** その XObject は、その PDF/VT ファイル内で複数回参照されます。recordlevel オプションが与えられている場合には、scope=file は、その XObject が複数のレコード内で使用される場合にのみ用いられるべきです（そうでないなら scope=record を用いるべきです）。 ***global*** （environment オプションを必要とします）その XObject か等価な XObject は、複数の PDF/VT ファイル内で参照されます。
xid	（文字列。PDF_begin_template_ext( ) でのみ可。なぜならこれ以外の種類の XObject に対しては識別子は自動的に作成されるからです）そのテンプレートに対して作成されるフォーム XObject に対する一意識別子。この識別子を、ISO 16612-2:2010 の 6.7.2 項で勧告されている形式で、すなわち uuid スキームと RFC 4122 に従った 128 ビット数値を持った URI として与えることを強く推奨します。この識別子は、PDF/VT に従って等価な PDF フォーム XObject を作成するテンプレート定義群（すなわち、同一の視覚的出力を作成するテンプレート群）に対しては、等しくするべきです。等価でないテンプレートは、異なる識別子を持つか、あるいは識別子を全く持たない必要があります。 scope=stream か scope=global を用いたテンプレートに対しては、複数の文書にわたるフォーム XObject のキャッシュ処理を可能にするため、この xid オプションを与えることを強く推奨します。 推奨される形式での xid の例：uuid:1228c416-48f2-e817-ad69-8206e41dca2d

10 PDF 取り込み（PDI）・pCOS 関数

この章の API メソッド：

- *PDF_open_pdi_document()*
- *PDF_close_pdi_document()*
- *PDF_open_pdi_page()*
- *PDF_close_pdi_page()*
- *PDF_fit_pdi_page()*
- *PDF_info_pdi_page()*
- *PDF_process_pdi()*
- *PDF_pcos_get_number()*
- *PDF_pcos_get_string()*
- *PDF_pcos_get_stream()*

注 この章で説明するメソッドはすべて、PDF 取り込み ライブラリー（PDI）を必要とします。PDI は、PDFlib+PDI と PDFlib Personalization Server（PPS）には含まれていますが、基本 PDFlib 製品には含まれていません。PDI の入手についての詳しい情報を得るには私達のウェブサイトにおいでください。

10.1 PDF 文書取り込み関数

C++ Java C# ***int open_pdi_document(String filename, String optlist)***

Perl PHP ***int open_pdi_document(string filename, string optlist)***

C ***int PDF_open_pdi_document(PDF *p, const char *filename, int len, const char *optlist)***

ディスクベースか仮想の PDF 文書を開き、以後の使用に備えます。

filename （名前文字列。グローバル *filenamehandling* オプションに従って解釈されます。表 2.1 参照）PDF ファイルの名前。

optlist PDF の開くオプションを指定したオプションリスト：

- 一般オプション：*errorpolicy*（表 1.5 参照）
- 表 10.1 に従った PDF 文書オプション：
  acceptdynamicxfa・*checkoutputintentprofile*・*infomode*・*inmemory*・*password*・*pcosengines*・*repair*・*requiredmode*・***shrug***・***useactions***・***usejavascript***・***userenditions***
- 表 10.1 に従ったタグ付き PDF 処理オプション：
  checktags・*usetags*
- 表 10.1 に従ったレイヤー処理オプション：
  parentlayer・*parenttitle*・*uselayers*

len （C 言語バインディングのみ）*filename* の長さ（バイト単位）。*len=0* にすると null 終了文字列を与える必要があります。

戻り値 PDI 文書ハンドル。文書の個々のページの処理や、文書のプロパティの取得に使えます。戻り値 -1（PHP では 0）は、PDF 文書を開くことができなかったことを示します。任意の数の PDF 文書を同時に開いておくことができます。メソッドの呼び出しが失敗したときは、その失敗の理由を *PDF_get_errmsg()* で取得することができます。

エラー動作は、*errorpolicy* オプションで変更することができます。

詳細　デフォルトでは、以下の条件のうち 1 つでも真のときは、その文書は拒否されます：

- 文書が破損しており、かつ修復できなかった（または *repair=none* が指定されていた）。
- 文書が暗号化されているのに、そのマスターパスワードが *password* オプションで与えられていない。*shrug* オプションを用いると、特定の条件下での保護された文書からのページ取り込みを可能にすることができます（PDFlib チュートリアル参照）。

この 2 つ目の場合には、オプション *requiredmode=minimum* か *requiredmode=restricted* を使って文書を開くこともできます。これは、*PDF_pcos_get_*()* メソッド群を使って、暗号化や文書情報フィールドなど、その PDF に関する情報を取得したいときに有用でしょう。

PDF の取り込みに関連した問題（PDF ファイル名の誤り、不適切な PDF データなど）の性質について、もっと詳しい情報を得るには、*PDF_get_errmsg()* を使って、より詳しいエラーメッセージを取得します。

スコープ　任意

表 10.1　PDF_open_pdi_document( ) のオプション

オプション	説明
accept-dynamicxfa	（論理値）true にすると、動的 XFA フォームを成功裡に開くことができます。唯一意味を持つ操作は pCOS パスへのクエリーです。ページは一切取り込めませんので PDF_open_pdi_page( ) への呼び出しは失敗します。デフォルト：false
checktags	（キーワード）取り込まれた構造エレメント群に対して構造エレメントネスト化規則群（PDFlib チュートリアル参照）が PDF_open_pdi_page( ) でチェックされるかどうかを指定します。使えるキーワード（デフォルト：none）： ***none*** タグネスト化規則は強制されません。この設定がデフォルトです。なぜなら、世の中に実在する文書の多くは構造エレメントネスト化規則群に違反しており、こうでなければ取り込めないからです。 ***relaxed*** strict と同様ですが、ただしいくつかの規則が強制されません（PDFlib チュートリアル参照）。 ***strict*** 取り込まれたタグがネスト化規則群に違反している場合には PDF_open_pdi_page( ) への呼び出しが失敗します。
checkoutput-intentprofile	（論理値。PDF/A・PDF/X でのみ意味を持ちます）true にすると、出力インテントの色要素の数が、その関連する ICC プロファイル内の要素の数に対してチェックされます。このオプションを false に設定すると、メモリー必要量が減りますが、入力文書群が一貫性のある出力インテントプロファイル群を内容として持っていることがわかっている場合にのみ使用するべきです。
infomode	（論理値）true にすると、pCOS インターフェイスで情報は取得できるけれども、PDF_open_pdi_page( ) を用いてページをカレント出力文書へ取り込むことはできないように文書が開きます。以下の文書を、infomode=true では開くことができます：パスワードがわからない暗号化された PDF（例外：Distiller の設定「オブジェクトレベルの圧縮：最高」を使って作成された PDF 1.6 以上の文書）。デフォルト：requiredmode=full にしているときは false、そうでなければ true
inmemory	（論理値）true にすると、PDI はファイル全体をメモリー内に読み込んで、そこからそれを処理します。これはシステムによっては非常な速度向上につながりますが（特に z/OS）、そのかわりメモリー使用が増えます。false にすると、文書の個々の部分が必要に応じてそのつどディスクから読み込まれます。デフォルト：false

表 10.1 PDF_open_pdi_document( ) のオプション

オプション	説明
parentlayer	（レイヤーハンドル。その入力文書がレイヤーを含んでいない場合か、uselayers=false の場合には無視されます）その文書から取り込まれたすべてのレイヤー定義を、この指定されたレイヤーの子として挿入します。もしこの指定されたレイヤーが、出力文書内のどこかでアクティブ化されていた場合には、それが親として用いられ、そうでなかった場合には、それはタイトル（区切り）としてのみ用いられます。デフォルト：親レイヤーなし
parenttitle	（ハイパーテキスト文字列。その入力文書がレイヤーを含んでいない場合か、uselayers=false の場合には無視されます）タイトルレイヤーを追加します。タイトルレイヤーは、ページの表示／非表示を直接制御せず、取り込まれたレイヤー定義群に対するヒエラルキー上の区切りとして働きます。デフォルト：親タイトルなし
password	（文字列）保護された PDF 文書を開いて取り込むために必要なマスターパスワード。暗号化された文書に対してパスワードを一切与えないと、その文書ハンドルは、その暗号化ステータスを取得するためだけに使えます。shrug オプションを用いると、特定の条件下での保護された文書からのページ取り込みを可能にすることができます（PDFlib チュートリアル参照）。
pcosengines	（オプションリスト）文書全体からのリソース収集について pCOS エンジン群を有効化か無効化します。あるエンジンを切るとは、それに対応する文書ごとの、およびページごとの疑似オブジェクト配列群が空になるということです。使えるキーワード（デフォルト：すべての pCOS エンジンが有効）： colorspace・extgstate・font・image・pattern・property・shading・template
repair	（キーワード）破損した PDF 入力文書の扱い方を指定します。文書を修復すると、通常の処理よりも時間がかかりますが、ある種の破損 PDF の処理が可能になるかもしれません。ただし文書によっては、修復不能なほど破損していることもあります。とりうるキーワード（デフォルト：auto）： ***auto*** PDF を開きつつあるときに問題を検出したときだけ文書を修復します。 ***force*** 文書に問題があるかないかにかかわらず、無条件に修復を試みます。 ***none*** 文書の修復を一切試みません。PDF に問題があるときは関数の呼び出しは失敗します。
requiredmode	（キーワード）文書を開くときに受け入れることのできる最低の pCOS モード（minimum/restricted/full）。要求しているモードよりも結果の pCOS モードのほうが低いときは、呼び出しは失敗します。呼び出しが成功したときは、結果の pCOS モードは最低でもこのオプションで指定しているものであることが保証されます。ただし、それより高いこともあります。たとえば requiredmode=minimum にしていても、暗号化されていない文書では結果は full モードになります。デフォルト：full
shrug	（論理値）true にすると、特定の条件下での保護された文書からのページ取り込みを可能にするシュラッグ機能がアクティブになります（PDFlib チュートリアル参照）。この shrug オプションを使用することによってあなたは、あなたが PDF 文書作成者の諸権利を遵守することを宣明します。デフォルト：false
useactions	（論理値。PDF/A では不可）true にすると、取り込んだ文書のすべての文書アクションが、生成される出力文書へ複製されます。複数の PDI 文書から同一のトリガーイベントに対するアクションを取り込んだ場合には、どのアクションが出力へ書かれるかは未定義です。PDF_begin/end_document( ) のオプション action・destination は、PDF 文書から取り込まれたアクションに優先します。デフォルト：false
usejavascript	（論理値）すべての文書 JavaScript（文書を開くとか閉じるとかのアクションに対する JavaScript のことではないので混同に注意）を取り込みます。そのような文書レベル JavaScript は、計算や検証のスクリプトなどフィールドまたは注釈アクションに対して必要になることがあります。ですので、ページ群をフォームフィールド群とともに取り込む場合（PDF_fit_pdi_page( ) のオプション usefields）には、JavaScript を取り込むことを推奨します。複数の文書から同じスクリプト名が取り込まれた場合には最初のスクリプトだけが使われます。デフォルト：false

表 10.1 PDF_open_pdi_document( ) のオプション

オプション	説明
userenditions	（論理値）true にすると、文書の名前ツリーからすべての表現を取り込みます。それら表現は JavaScript から名前でアクセスできます。デフォルト：false
uselayers	（論理値。入力がレイヤーを含んでいる場合にのみ意味を持ちます）true にすると、取り込みページ群において使用されているすべてのレイヤー定義が取り込まれます。このオプションは、レイヤー定義群に関してのみ影響を及ぼし、レイヤーの内容本体に関しては影響しません。なぜなら PDI はつねにページ上のすべてのレイヤーの内容を取り込むからです。uselayers=false を使用するためには、生成される文書はレイヤーを一切含んではいけません。すなわち、レイヤーを持っている PDF 文書についてはすべて uselayers=false を用いて開く必要がありますし、PDF_define_layer( ) を呼び出してもいけません。デフォルト：true
usetags	（論理値。タグ付き PDF 入出力に対してのみ意味を持ちます）true にすると、取り込まれた文書の構造ヒエラルキーが、その構造エレメントタグ群を後でそのページ群とともに取り込めるよう、読み込まれます。デフォルト：オブジェクトスコープでは false、それ以外では true

---

*C int PDF_open_pdi_callback(PDF *p, void *opaque, pdf_off_t filesize,*
*size_t (*readproc)(void *opaque, void *buffer, size_t size),*
*int (*seekproc)(void *opaque, pdf_off_t offset),*
*const char *optlist)*

PDF 文書を、カスタムのデータ元から開いて、以後の使用に備えます。

opaque 入力 PDF 文書に関連づけたい何らかのユーザデータへのポインター。このポインターは、コールバック関数の 1 番目の引数として渡され、どのようにでも使えます。PDI はこの不透明ポインターを、いかなる形でも使いません。

filesize PDF 文書全体のサイズを、バイト単位で指定します。

readproc *buffer* で指し示されるメモリーへ *size* バイトをコピーするコールバックメソッド。文書の終わりに達したときは、要求より少ないバイトをコピーすることがあります。この関数は、コピーしたバイト数を返す必要があります。

seekproc 文書内のカレントの読み取り位置を設定するコールバック関数。*offset* で、文書の先頭からの位置を指定します（0 が先頭バイトを意味します）。この関数は、成功したときは 0 を返し、そうでなければ -1 を返す必要があります。

optlist PDF を開くオプションを指定したオプションリスト。*PDF_open_pdi_document( )* のすべてのオプションを使えます。

戻り値 PDI 文書ハンドル。文書の個々のページの処理や、文書のプロパティの取得に使えます。戻り値 -1 は、PDF 文書を開くことができなかったことを示します。任意の数の PDF 文書を同時に開いておくことができます。戻り値は、カレントの文書スコープを終えるまで使えます。メソッドの呼び出しが失敗したときは、その失敗の理由を *PDF_get_errmsg( )* で取得することができます。

詳細 これは、PDF 文書をディスク上のファイルやメモリー内から与えるのではなく、何らかのデータ元から任意の大きさごとに PDF データを取り出したい応用のための、特殊なインターフェイスです。*pdf_off_t* 型は *pdflib.h* 内で条件的に定義されています。それは通常、2GB を超える大きなファイルのためのオフセット型として 64 ビット値を保持します。アプリケーションを大容量ファイルサポート（LFS）ありでビルドする必要があります。

スコープ 任意。**オブジェクト** スコープでは、PDI 文書ハンドルは PDF 文書から情報を取得するためにだけ使えます。

バインディング C 言語バインディングでのみ利用可能。

C++ Java C# ***void close_pdi_document(int doc)***

Perl PHP ***close_pdi_document(int doc)***

C ***void PDF_close_pdi_document(PDF *p, int doc)***

開いている PDI ページハンドルをすべて閉じてから、入力 PDF 文書を閉じます。

doc *PDF_open_pdi_document( )* で取得した有効な PDF 文書ハンドル。

詳細 このメソッドは、PDF 取り込み文書を閉じて、その文書に関連したすべてのリソースを解放します。文書のページを開いていたなら、すべて自動的に閉じられます。その文書ハンドルは、この呼び出しの後は使ってはいけません。さらに取り込みたいページがあるときは、その PDF 文書は閉じるべきではありません。PDF 取り込み文書は、任意の回数開いたり閉じたりできますが、そうすると PDF 出力ファイルが不必要に大きくなることがあります。

スコープ **任意**

10.2 PDF ページ取り込み関数

C++ Java C# ***int open_pdi_page(int doc, int pagenumber, String optlist)***

Perl PHP ***int open_pdi_page(int doc, int pagenumber, string optlist)***

C ***int PDF_open_pdi_page(PDF *p, int doc, int pagenumber, const char* optlist)***

ページを、以後の *PDF_fit_pdi_page()* での使用のために準備します。

doc *PDF_open_pdi_document()* で取得した有効な PDF 文書ハンドル。

pagenumber 開きたいページの番号。先頭ページの番号は 1 です。

optlist ページ関連オプションを指定したオプションリスト：

- 一般オプション：*errorpolicy*（表 1.5 参照）
- 表 10.2 に従った PDF ページオプション：
  boxexpand・*checktransgroupprofile*・*clippingarea*・*cloneboxes*・*forcebox*・*ignorepdfversioninitgraphicsstatepdiusebox*・*usetags*
- 共通 XObject オプション（表 9.11 参照）：
  associatedfiles・*iconname*・*layer*・*metadata*・*pdfvt*・*transform* ・*transparencygroup*
- C の場合と、Perl・PHP・Ruby で *stringformat=legacy* の場合のためのエンコーディングオプション：
  hypertextencoding（表 2.1 参照）

戻り値 PDI ページハンドル。*PDF_fit_pdi_page()* でページを配置するために使えます。戻り値 -1（PHP では 0）は、そのページが開けなかったことを示します。メソッドの呼び出しが失敗したときは、その失敗の原因を *PDF_get_errmsg()* で取得することができます。返されたハンドルは、カレントの文書スコープを終えるまで使えます。

エラー動作は、*errorpolicy* オプションで変更することができます。

詳細 このメソッドは、取り込んだページを構成するすべてのデータを出力文書へコピーしますが、出力上にいかなる視覚効果をも与えません。取り込んだページを、生成する出力文書のどこかに実際に配置するには、*PDF_fit_pdi_page()* を使う必要があります。このメソッド以下の場合には失敗します：

- 文書が、カレント PDF 文書と非互換な PDF バージョンを使用している。PDF 1.6 以下の PDF バージョンについては、同一バージョン以下のすべてのバージョンが互換です。PDF 1.7・PDF 1.7ext3・PDF 1.7ext8・PDF 2.0 は、PDI を用いたページ取り込みに関する限りはすべて互いに互換です。しかし、PDF/A モードでは入力 PDF バージョン番号は無視されます。なぜなら PDF/A では PDF バージョンヘッダーを無視する必要があるからです。
- 文書が、カレントの PDF/A または PDF/X または PDF/VT または PDF/UA 出力準拠レベルと互換でないか、または非互換な出力インテントを使用している。
- 文書が、矛盾する PDF/A または PDF/X 出力インテントを含んでいる場合には、ページは一切取り込まれません。

PDF 取り込みに関連した問題（不正な PDF データなど）に関する情報をより詳しく知りたいときは、*PDF_get_errmsg()* を呼び出すことができます。

任意の数のページを同時に開くことができます。同じページを複数回開くと、別々のハンドルが返され、そして各ハンドルを 1 回ずつ閉じる必要があります。

タグ付き PDF 2.0 のページ群を取り込もうとする場合、その取り込まれるページ上の構造エレメントが PDF 1.7 以外の名前空間を使用しているならば、*usetags=true* を用いて PDF 1.7 へそのページ群を取り込むことはできません。

PDF/A 取り込まれる文書は、カレント PDF/A 出力準拠レベル（PDFlib チュートリアルを参照）および出力インテントに互換である必要があります。

PDF/UA 取り込まれる文書群が PDF/UA に準拠している必要があります。タグなし文書も、非 PDF/UA 文書も、*PDF_open_pdi_document()* で *usetags=false* を用いて読み込んだタグ付き文書も、配置することは可能です。しかし、そのようなページは自動的にページ装飾として標識されます。ゆえにこの場合には、カレントでアクティブなエレメントが Artifact を子として許容している必要があります。

取り込まれる文書のロールマップが、*PDF_begin_document()* の *rolemap* オプションによって与えたマッピングと互換である必要があります（詳しくは PDFlib チュートリアルを参照）。すなわち、カスタム構造種別を、*rolemap* オプションと、取り込まれる文書のロールマップとで、異なる標準種別へマップしていてはいけません。

取り込まれるページの見出し構造は、生成される文書の構造種別と互換である必要があります。すなわち、*structuretype=weak* ならば、ページ上で *H1*・*H2* などだけが使われている必要があります（*H* は不可）し、*structuretype=strong* ならば、取り込まれるページ上で H だけが使われている必要があります（*H1*・*H2* は不可）。数字ありと数字なしの見出し要素が混在しているページは拒否されます。

PDF/VT 取り込まれる文書は、PDF/X-3/4/4p か PDF/VT-1 に準拠している必要があり、かつ、生成される文書と同じ出力インテントを使用している必要があります。取り込まれる文書の中の文書部分メタデータ（DPM）は取り込まれません。*PDF_begin_document()* で *usestransparency=false* オプションが指定されているにもかかわらず、取り込まれるページが透過を含んでいる場合には、この呼び出しは失敗します。

取り込みページから生成されるフォーム XObject は、特定の条件が満たされているときには、カプセル化されているとして標識されます（PDFlib チュートリアル参照）。

PDF/X 取り込まれる文書は、カレントの PDF/X 出力準拠レベル（詳しくは PDFlib チュートリアルを参照）に互換である必要があり、かつ、生成される文書と同じ出力インテントを使用している必要があります。

PDF/X-4/5：取り込まれるページが、生成される文書の出力インテントプロファイルと同一の CMYK ICC プロファイルを使用している場合には、そのページは拒絶されます。

スコープ **オブジェクト**以外任意

表 10.2 PDF_open_pdi_page( ) のオプション

オプション	説明
boxexpand	（float または float4 個のリスト）pdiusebox オプションで選んだページ枠を、4 辺すべてを同じ値だけ（値 1 個を与えた場合）、または左 / 右 / 下 / 上辺を個別に（値 4 個を与えた場合）拡張します。負の値でページサイズを小さくすることもできます。このオプションを利用すれば、取り込んだページのすべてのページ枠の外に位置する内容を配置したり、余白を加えたりすることができます。デフォルト：0

表 10.2 PDF_open_pdi_page( ) のオプション

オプション	説明
checktrans-groupprofile	（論理値。PDF/A・PDF/X でのみ意味を持ちます）true にすると、取り込まれたページが透過グループを含んでいる場合には、その色空間が、生成される出力文書との整合性と互換性についてチェックされます。整合しない入力文書と、色空間の衝突は、準拠しない PDF/X または PDF/A 出力を生成するおそれがありますので、これによってそれを防止します。このオプションを false に設定すると、メモリー必要量が低減されますが、この設定は、取り込まれるページが、準拠する透過グループ（もしあれば）を内容としていることがわかっている場合にのみ用いるべきです。デフォルト：true
clippingarea	（キーワード）取り込んだページのページ枠群のうちのどれを切り抜きに用いるかを指定します。指定した領域の外側のページ内容は、この取り込んだページを新しいページ上に配置した後、見えません。使えるキーワード（デフォルト：pdiusebox）： ***art*** ArtBox が存在するならそれを、ないなら CropBox を用います ***bleed*** BleedBox が存在するならそれを、ないなら CropBox を用います ***crop*** CropBox が存在するならそれを、ないなら MediaBox を用います ***media*** MediaBox（これは必ず存在します。かつ、他のすべての枠を内包している必要があります）を用います ***pdiusebox*** cloneboxes が指定されているなら MediaBox を用い、そうでないなら pdiusebox オプションで指定された枠を用います ***trim*** TrimBox が存在するならそれを、ないなら CropBox を用います
cloneboxes	（論理値。boxexpand・forcebox・pdiusebox のいずれかを与えている場合には不可。PDF_fit_pdi_page( ) の cloneboxes オプションと整合させる必要があります）にすると、ページは、PDF_fit_pdi_page( ) の cloneboxes オプションによる枠複製のために用意されます。デフォルト：false
forcebox	（長方形）ページ枠を強制的に、指定した値にします。このオプションは pdiusebox・boxexpand オプションをオーバーライドします。これを利用すれば、取り込んだページのすべてのページ枠の外に位置する内容を配置することができます。この値は、取り込んだページが Rotate キーを含んでいる場合には、注意深く選ぶ必要があります。boxexpand オプションのほうが、Rotate キーの有無にかかわらずうまく働きますので、望ましいでしょう。デフォルト：pdiusebox オプションで選ばれた枠
ignore-pdfversion	（論理値）true にすると、入力 PDF 文書の PDF バージョン番号は無視されます。すなわち、カレント PDF 出力文書よりも高い PDF バージョンを持った文書からのページを取り込めるようになります。より高い PDF バージョンを持ちながらもカレント PDF 出力レベルに完全準拠している PDF 文書に対してこれは有用でしょう。取り込むページに PDF 出力互換性の違反がないようにするのはユーザー側の役割です。デフォルト：一般に false、ただし PDF/A・PDF/X モードでは true
initgraphics-state	（論理値。PDF/VT モードでは強制的に true）true にすると、すべてのグラフィックステートパラメーターが、取り込みページに対するデフォルト値で明示的に初期化されます。これは、明示的にすべての値を設定せずデフォルトに依拠している取り込みページに対し、カレントグラフィックステートパラメーター群が適用されることを防ぎます。デフォルト：false
pdiusebox	（キーワード。cloneboxes を与えている場合には不可）取り込んだページの寸法を決定するためにどの枠寸法を用いるかを指定します。この枠寸法は、PDF_fit_pdi_page( ) での拡縮操作のために用いられます。この枠は、ページの見える内容も、clippingarea オプションで変更されない限り、決定します。デフォルト：crop。 ***art*** ArtBox があればそれを、なければ CropBox を用います ***bleed*** BleedBox があればそれを、なければ CropBox を用います ***crop*** CropBox があればそれを、なければ MediaBox を用います ***media*** MediaBox（これは必ず存在します）を用います ***trim*** TrimBox があればそれを、なければ CropBox を用います

表 10.2 PDF_open_pdi_page( ) のオプション

オプション	説明
usetags	（論理値。タグ付き PDF 入力および出力に対して、かつ、文書を usetags=true を用いて開いている場合にのみ意味を持ちます）true にすると、取り込まれるページの構造タグ群が、出力文書の構造ヒエラルキーへ複製されます。この場合、PDF_fit_pdi_page( ) はページスコープ内でのみ呼び出すことができます。デフォルト：true

C++ Java C# ***void close_pdi_page(int page)***

Perl PHP ***close_pdi_page(int page)***

C ***void PDF_close_pdi_page(PDF *p, int page)***

ページハンドルを閉じて、ページ関連のリソースをすべて解放します。

page *PDF_open_pdi_page( )* で取得した有効な PDF ページハンドル（ページ番号ではありません！）。

詳細 このメソッドは、*page* で指定するページハンドルに結びついたページを閉じて、関連するリソースをすべて解放します。この呼び出しの後に *page* を使ってはいけません。

スコープ **オブジェクト**以外任意

C++ Java C# ***void fit_pdi_page(int page, double x, double y, String optlist)***

Perl PHP ***fit_pdi_page(int page, float x, float y, string optlist)***

C ***void PDF_fit_pdi_page(PDF *p, int page, double x, double y, const char *optlist)***

取り込んだ PDF ページを、ページ上に、さまざまなオプションに従って配置します。

page *PDF_open_pdi_page( )* で取得した有効な PDF ページハンドル（ページ番号ではありません！）。ページハンドルは閉じてあってはいけません。

x・y ページをさまざまなオプションに従って置きたい参照点の座標を、ユーザー座標系で指定します。

optlist ページオプションを指定したオプションリスト：

- 表 6.1 に従ったはめ込みオプション：
  blind・boxsize・fitmethod・matchbox・orientate・position・rotate・scale・showborder
- 表 9.3 に従ったページ処理のためのオプション：*adjustpage・gstate*
- 表 10.3 に従った処理オプション：*cloneboxes・useactions・useannots・usefields*
- 表 14.4 に従った、短縮構造エレメントタグ付けのためのオプション（**ページ**スコープでのみ可）：*tag*

詳細 このメソッドは *PDF_fit_image( )* に似ていますが、取り込み PDF ページに対して働く点が違います。

タグ付きページ（すなわち、タグ付き PDF が作成されて、かつ、そのページが *usetags=true* を用いてタグ付き PDF から取り込まれた）を複数回配置することはできません。

タグ付き PDF モードでは、*PDF_fit_pdi_page( )* を呼び出す前に、*PDF_fit_pdi_page( )* が成功するかどうかをチェックする（そして失敗して例外が発生することを避ける）ために、*PDF_info_pdi_page( )* で *fittingpossible* キーワードを用いることを推奨します。

取り込んだ1つのページをフォームフィールドとともに配置できるのは1回だけです。取り込まれるフィールドか注釈が JavaScript 名称ツリーの中のスクリプトを使用している場合には *PDF_open_pdi_document( )* のオプション *usejavascript* を用いればこのスクリプトを取り込むことができます。

注釈種別 *Caret*・*FreeText*・*Line*・*Stamp* のいずれかを内容として持っているページを、*useannots*（デフォルトで有効です）か *usefields* を用いて取り込む場合に、その注釈かフィールドが体裁ストリームを持っていないならば、Noto フォント群へのアクセスを構成する必要があります。これらのフォントは PDFlib ディストリビューションに含まれています（PDFlib チュートリアルを参照）。

PDF/UA *useannots*・*usefields* オプションに制約が課せられます。

PDF/X オプション *useannots*・*usefields*：注釈が取り込まれるのは、生成されるページの BleedBox（あるいは BleedBox が存在していない場合には TrimBox/ArtBox）の完全に外側にそれが位置している場合だけです。

スコープ **ページ・パターン・テンプレート・グリフ**。ただし、タグ付き PDF 文書内のページが *usetags=true* で読み込まれている場合には、このメソッドは**ページ**スコープ内でのみ呼び出し可能です。

表 10.3 PDF_fit_pdi_page( ) の追加オプション

オプション	説明
clip-annotations	（論理値）true にすると、取り込まれるページの外側に位置している注釈とフォームフィールドは無視されます。false にすると、注釈とフォームフィールドがその位置にかかわらず取り込まれます。デフォルト：false
cloneboxes	（論理値。PDF_begin_page_ext( ) で topdown オプションを与えている場合には不可。PDF_open_pdi_page( ) の cloneboxes オプションと整合させる必要があります。ページスコープ内でのみ可）。 このオプションは PDF 入力ページの幾何情報を転写します。このオプションを true に設定すると、以下の結果になります（デフォルト：false）： ▸ 取り込んだページの中の Rotate・MediaBox・TrimBox・ArtBox・BleedBox・CropBox 項目がすべて出力ページへ複製されます。 ▸ ページ内容が、入力ページが転写されるように配置されます。ユーザーが、配置されるページの位置や寸法を変えることはできません。引数 x・y と以下オプションは無視されます：adjustpage・boxsize・fitmethod・orientate・position・rotate・scale。入力ページの幾何情報の転写は、PDF_fit_pdi_page( ) の呼び出しの際にデフォルト座標系が有効な場合にのみ可能です。 ▸ cloneboxes オプションによって作成されるページ枠が、PDF_begin_page_ext( ) の artbox・bleedbox・cropbox・trimbox・mediabox・rotate オプションと width・height 引数をオーバーライドします。
useactions	（論理値。ページスコープ内でのみ意味を持ちます。PDF/A では不可）true にすると、配置されるページのすべてのページアクションが出力ページへ複製されます。個々の取り込まれたアクションは、他の PDI ページからさきに取り込まれていたかもしれないアクション、または、PDF_begin/end_page_ext( ) の action オプションを用いて指定されていたアクションをオーバーライドします。デフォルト：false

表 10.3 PDF_fit_pdi_page( ) の追加オプション

オプション	説明
useannots	（オプションリスト。ページスコープ内でのみ可。PDF/UA では、ページを usetags=false を用いて開いている場合には許されません）ページとともに取り込まれる注釈種別群（デフォルト：カレントスコープがページ以外なら none、そうでないなら all）： ***3D・Caret・Circle・FileAttachment・FreeText・Highlight・Ink・Line・Link・Movie[1]・Polygon・PolyLine・Popup・Projection[1]・Redact[1]・RichMedia・Screen・Sound1・Square・Squiggly・Stamp・StrikeOut・Text・Underline*** （論理値）取り込みたい注釈種別（群） ***all*** （キーワード）true にすると、値 false にした種別以外のすべての注釈種別が取り込まれます。 ***none*** （キーワード）true にすると、値 true にした種別以外の注釈種別はどれも取り込まれません。オプション all は none をオーバーライドします。
usefields	（オプションリスト。ページスコープ内でのみ可。PDF/UA では、ページを usetags=false を用いて開いている場合には許されません）ページとともに取り込まれるフィールド種別群（デフォルト：none）： ***barcode・checkbox・combobox・listbox・pushbutton・radiobutton・signature・textfield*** （論理値）取り込みたいフィールド種別（群） ***all*** （キーワード）true にすると、値 false にした種別以外のすべてのフィールド種別が取り込まれます。 ***none*** （キーワード）true にすると、値 true にした種別以外のフィールド種別はどれも取り込まれません。オプション all は none をオーバーライドします。 フォームフィールドを複数の文書から取り込むことはできず、ただ一つの PDF 入力文書から取り込むことのみ可能です。フォームフィールドを生成するには、他の PDF 文書から取り込むか、あるいは PDF_create_field( ) を用いるかのどちらかの方法を使います。両方の方法を一緒に使うことはできません。

1. この注釈種別を、PDF_create_annotation( ) を用いて生成することはできません。

C++ Java C# ***double info_pdi_page(int page, String keyword, String optlist)***

Perl PHP ***float info_pdi_page(int page, string keyword, string optlist)***

C ***double PDF_info_pdi_page(PDF *p, int page, const char *keyword, const char *optlist)***

PDI ページに対する組版計算を実行し、結果のメトリックを取得します。

page PDF_open_pdi_page( ) で取得した有効なページハンドル。

keyword 求める情報を指定したキーワード：

- 表 6.3 に従った、オブジェクトはめ込みの結果をクエリーするためのキーワード：
  boundingbox・fitscalex・fitscaley・height・objectheight・objectwidth・width・x1・y1・x2・y2・x3・y3・x4・y4
- 表 10.4 に従ったページ関連キーワード：
  *mirroringx・mirroringy・**pageheight**, **pagewidth**・rotate・**xid***
- 表 10.4 に従ったタグ付き PDF 関連キーワード：
  fittingpossible・lang・topleveltag・topleveltagcount

optlist 拡縮と配置の詳細を指定したオプションリスト：

- 一般オプション：*errorpolicy*（表 1.5 参照）

- 表6.1に従ったはめ込みオプション群(PDFページを*PDF_open_pdi_page()*の*cloneboxes*オプションで開いていたときには、これらのオプションは無視されます)：*boxsize*・*fitmethod*・*matchbox*・*orientate*・*position*・*rotate*・*scale*
- 表 9.3・表 10.3 に従ったページ処理のためのオプション群は意味を持ちません。ただしそれらを与えることはできますが、*PDF_fit_pdi_page()* と *PDF_info_pdi_page()* に対して統一的なオプションリストを実現するために無視されます：*adjustpage*・*gstate*・*useannots*・*usefields*
- 表 14.4 に従った、短縮構造エレメントタグ付けのためのオプション：*tag*
- ページの最上位レベル構造エレメント群のうちの 1 つから何らかの情報を取得するためにそれを選択するためのオプション：*index*

戻り値 *keyword* によって求められた通りの何らかのページ特性の値。求められた特性がそのページについて得られないときには、このメソッドは 0 を返します。オブジェクトハンドルが求められている場合には（例：*clippingpath*）、このメソッドはそのオブジェクトへのハンドルを、あるいはそのオブジェクトが得られないときには -1（PHP では 0）を返します。求められたキーワードがテキストを生み出す場合には、文字列番号が返されますので、それに対応する文字列を、*PDF_get_string()* を用いて取得する必要があります。

詳細 このメソッドは、指定したオプション群に従って取り込みページを配置するために必要なすべての計算を実行しますが、ページ上には実際の出力を一切生成しません。ページを配置するための参照点は *{o o}* であると見なされます。*PDF_open_pdi_page()* の *cloneboxes* オプションを与えているときは、ページは元のページと同じ位置に（ページ枠に対して相対的に）配置されます。

PDF/UA *fittingpossible* に対するチェックが、非 PDF/UA モードよりも厳格です。

スコープ **オブジェクト**以外任意

表 10.4 PDF_info_pdi_page( ) のキーワード

キーワード	説明
fittingpossible	（タグ付き PDF 出力の場合にのみ意味を持ちます）カレントコンテキストの中で PDF_fit_pdi_page( ) を用いてそのページを配置できるかどうかをチェックします。そのページを配置できる場合には値 1 が返されます。以下のいずれかの理由によりはめ込みが失敗するであろう（すなわち PDF_fit_pdi_page( ) が例外を発生させるであろう）場合には値 0 が返されます： ▸ そのページの最上位レベルタグ群のいずれかが、構造エレメント群に対するネスト化規則群によって、カレントでアクティブなタグの下では許容されない。 ▸（そのページが usetags=false を用いて読み込まれているか、あるいはページ装飾として配置されている場合には該当しません）その空でないページが、タグ解除されているか、構造エレメントを含んでおらず、かつ、カレントでアクティブなタグの子として直接コンテンツが許容されない。 ▸（そのページが usetags=false を用いて読み込まれているか、あるいはページ装飾として配置されている場合には該当しません）そのページはすでに配置されている。 ▸ 弱い文書構造を持った PDF/UA-1：カレントの構造エレメントとその親群と、取り込まれる構造下位ヒエラルキーとの間に、見出しレベル番号のギャップがある。 PDF_fit_pdi_page( ) の tag オプションを与えることもでき、これは考慮に入れられます。この tag オプションの tagname サブオプションのみが評価されます。 この結果は、カレントのコンテキストに対してのみ有効なものですので、このキーワードは、ページを配置しようと試みる直前に使用するべきです。
lang	取り込まれるページの最上位レベル構造エレメント群のいずれかの Lang 属性に対する文字列番号、あるいは Lang 属性が全く決定できない場合には –1。最上位レベルエレメントが複数ある場合には、index オプションを用いて 1 個を選ぶことができます。
mirroringx*・*mirroringy	与えたオプション群に従ったページの横反転・縦反転（1 か –1 かで表されます）
pageheight, pagewidth	元のページの高さと幅をポイント単位で表したもの
rotate	cloneboxes=true のとき：取り込んだページの回転角を度単位で表したもの、すなわち、そのページの Rotate キの値。とりうる値は 0・90・180・270。 cloneboxes=false のとき：つねに 0
topleveltag	取り込まれるページが、usetags=true を用いて開かれており、かつ、構造エレメントに紐付けられたマーク付きコンテンツを含んでいる場合には、このページの最上位構造エレメントのうちの 1 つの名前に対する文字列番号、そうでないなら –1（例：Artifact を表すページに対して）。最上位エレメントが複数ある場合には、index オプションを用いて 1 つを選ぶことができます。そのタグが、取り込まれる文書のロールマップ内でロールマップされているカスタムエレメントである場合には、そのカスタムエレメント名ではなく、それに対応する標準エレメント名が報告されます。
topleveltagcount	取り込まれるページの構造ヒエラルキーの最上位レベルにある構造エレメントの数。 lang・topleveltag キーワードを用いて、これらのエレメントに関する情報を、index オプションを用いて 1 つ選んで、取得することができます。そのページが usetags=false を用いて取り込まれているか、あるいはタグ解除されているために、または構造エレメントに対応するマーク付きコンテンツを全く含んでいないために、構造エレメントが全く取り込まれない場合には、0 が返されます。
xid	（PDF/VT のみ）そのページの GTS_XID エントリーに対する文字列番号、あるいは GTS_XID 値が何も割り当てられていない場合には –1。この GTS_XID 文字列は、DPM に対する CIP4/Summary/Content/Referenced メタデータプロパティ内で使用することができます。

Table 10.5 PDF_info_pdi_page( ) のオプション

オプション	説明
index	（整数。lang・topleveltag キーワードに対してのみ意味を持ちます） そのページの最上位構造エレメント群のうち、属性が取得されるものを 1 つ選びます。この値は範囲 0 ～（toplevelcount-1）内でなければなりません。デフォルト：0

10.3 その他の PDI 処理

C++ Java C# ***int process_pdi(int doc, int pagenumber, String optlist)***

Perl PHP ***int process_pdi(int doc, int pagenumber, string optlist)***

C ***int PDF_process_pdi(PDF *p, int doc, int pagenumber, const char* optlist)***

取り込んだ PDF 文書の、ある種の要素を処理します。

doc *PDF_open_pdi_document()* で取得した有効な PDF 文書ハンドル。

pagenumber （正の整数。*action=copypageoutputintent* の場合にのみ可。そうでない場合には -1 にするべきです）出力インテントが位置している入力文書の中のページの 1 から始まる番号。

optlist PDI 処理オプションを指定したオプションリスト：

- 一般オプション：*errorpolicy*（表 1.5 参照）
- 表 10.6 に従った PDI 処理オプション：*action*・*block*

戻り値 メソッドが成功したときは値 1、メソッドが失敗したときはエラーコード -1（PHP では 0）。*errorpolicy=exception* にすると、このメソッドはエラー時に例外を発生させます。

action=copypageoutputintent の場合には、有効なページレベル出力インテントがそのページに対して見つかったならば出力インテントハンドルが返され、そうでないならエラーコード 1（PHP では 0）が返されます。*PDF_process_pdi()* を呼び出す前に pCOS を用いてページレベル出力インテントの存在をチェックすることを推奨します。

action=copyallblocks に対して入力ページ上でブロックが全く見つからなかったときには、このメソッドは 1 を返します。

PDF/A 出力インテントは、このメソッドで *copyoutputintent* オプションを指定するか、または *PDF_load_iccprofile()* を使って設定することができます。ただし、デバイス独立な色しか文書で使っていないときは、出力インテントは必要ありません。

PDF/X 出力インテントは、このメソッドで *copyoutputintent* オプションを指定するか、または *PDF_load_iccprofile()* を使って設定する必要があります。

スコープ *action=copyoutputintent*・*copypageoutputintent* の場合は**文書**、
action=copyallblocks・*action=copyblock* の場合は**ページ**

表 10.6 PDF_process_pdi( ) のオプション

オプション	説明
action	（キーワード。必須）PDF 処理の種類を指定します： ***copyallblocks*** （PPS でのみ利用可能）入力文書のページからすべての PDFlib ブロックをカレント出力ページへ block オプションに従って複製します。 ***copyblock***（PPS でのみ利用可能）入力文書のページから PDFlib ブロック 1 個をカレント出力ページへ block オプションに従って複製します。 ***copyoutputintent*** （PDF/A・PDF/X でのみ意味を持ち、それ以外の場合には無視されます）取り込んだ文書の PDF/X か PDF/A の出力インテント ICC プロファイルを、出力文書へ複製します。出力インテントを複製しようとする 2 回目およびそれ以降の試みは無視されます。文書に複数の出力インテントが入っているときは、最初のものが使われます。文書レベル出力インテントの存在を、pCOS パス /Root/OutputIntents[] を用いてクエリーすることも可能です。 入力文書と出力文書が PDF/X-4p か PDF/X-5n に準拠しているときは、外部出力インテント ICC プロファイルへの参照が複製されます。参照されているプロファイルが埋め込まれている場合には、その紐付けされているプロファイルも複製されます。このオプション action=copyoutputintent は、入力が PDF/X-4p/5n に準拠しているのに出力がしていないときには許されません。 ページ出力インテントが存在していない場合には、ハンドル -1（PHP では 0）が返されるか例外が投げられます（errorpolicy に依存）。 PDI 入力文書と、生成される出力は、以下の条件に従う必要があります（そうでないとアクションは拒絶されます）： 入力が PDF/X-4p/5pg に準拠している場合には、出力は PDF/X-4p/5pg か PDF 2.0 に準拠している必要があります。 入力が PDF/A に準拠している場合には、出力は PDF/A か PDF 2.0 に準拠している必要があります。 入力が PDF/X に準拠している場合には、出力は PDF/X か PDF 2.0 に準拠している必要があります。 ***copypageoutputintent*** （PDF/A か PDF/X でのみ許されます）pagenumber で指定されたページの、PDF/A か PDF/X のページレベル出力インテントへのハンドルを返し、その出力インテントを出力文書へ複製します。この返されたハンドルは、PDF_begin_page_ext( ) の outputintents オプションで使うことができます。ページレベル出力インテントの存在については、pCOS パス pages[]/OutputIntents[] を用いてクエリーできます。
block	（オプションリスト。action=copyallblocks・action=copyblock では必須）ブロック複製処理の詳細を指定します。以下のサブオプションを使えます： ***blockname*** （名前文字列。action=copyblock でのみ、かつその場合には必須）そのブロックの名前 ***outputblockname*** （名前文字列。action=copyblock でのみ）そのブロックが出力ページ上で格納される名前。デフォルト：blockname オプションの値 ***pagenumber*** （整数。必須）入力文書内の、そのブロックが位置しているページの、1 から始まる番号。

10.4 pCOS メソッド

pCOS メソッドはすべて、PDF 文書内の目指すオブジェクトを指し示すパスによって動作します。pCOS パスについて詳しくは pCOS パスレファレンスで説明しています。

注 評価モードでは、pCOS は最大 1 MB または 10 ページの入力文書を受け付けます。ただし、次の要素はそれより大きい文書からも評価モードで取得できます：ページ数・ページ寸法・ブロック詳細・すべての汎用擬似オブジェクト。

C++ Java C# ***double pcos_get_number(int doc, string path)***

Perl PHP ***double pcos_get_number(long doc, string path)***

C ***double PDF_pcos_get_number(PDF *p, int doc, const char *path, ...)***

数値または**論理値**型の pCOS パスの値を取得します。

doc *PDF_open_pdi_document( )* で取得した有効な文書ハンドル。

path 数値または論理値オブジェクトへの完全な pCOS パス。

追加引数 （C 言語バインディングのみ）*key* 引数にプレースホルダを入れているときは（*%s* なら文字列、*%d* なら整数。*%%* を使うと 1 個のパーセント記号）、それに対応して可変の数の追加引数を与えることができます。これらの引数を使えば、可変の数値や文字列の値を入れて複雑なパスを明示的に構築する手間が省けます。プレースホルダの数と型を、与える追加引数と確かに一致させるのは、クライアント側の役割です。

戻り値 pCOS パスで同定されたオブジェクトの数値。論理値の場合は、*true* なら 1、そうでなければ 0 が返されます。

スコープ 任意

C++ Java C# ***string pcos_get_string(int doc, string path)***

Perl PHP ***string pcos_get_string(long doc, string path)***

C ***const char *PDF_pcos_get_string(PDF *p, int doc, const char *path, ...)***

名前または**数値**または**文字列**または**論理値**型の pCOS パスの値を取得します。

doc *PDF_open_pdi_document( )* で取得した有効な文書ハンドル。

path 名前または文字列または論理値オブジェクトへの完全な pCOS パス。

追加引数 （C 言語バインディングのみ）*key* 引数にプレースホルダを入れているときは（*%s* なら文字列、*%d* なら整数。*%%* を使うと 1 個のパーセント記号）、それに対応して可変の数の追加引数を与えることができます。これらの引数を使えば、可変の数値や文字列の値を入れて複雑なパスを明示的に構築する手間が省けます。プレースホルダの数と型を、与える追加引数と確かに一致させるのは、クライアント側の役割です。

戻り値 pCOS パスで同定されたオブジェクトの値の文字列。論理値の場合は、文字列 *true* か *false* が返されます。

詳細 このメソッドは、pCOS が完全モードで動作していないとき、オブジェクトが**文字列**型だと例外を発生させます。ただしオブジェクトによっては、特定の条件下において限定モードでも取得できるものがあります。詳しくは pCOS パスレファレンスを参照してください。

このメソッドは、PDF 文書から取得する文字列がテキスト文字列であることを前提にしています。バイナリーデータの入った文字列オブジェクトは、データを一切変更しない *PDF_pcos_get_stream( )* で取得する必要があります。

スコープ 任意

バインディング C 言語バインディング：文字列は BOM なしの UTF-8 形式（IBM System i・IBM Z では EBCDIC-UTF-8）で返されます。返される文字列は、最大 10 項目の円環バッファ内に格納されます。10 個を超える文字列が変換されるときは、バッファは再利用されますので、クライアント側では、もし 10 個を超える文字列に並列にアクセスしたいときには、文字列を複製する必要があります。たとえば、1 つの *printf( )* 文に対する引数群としてこのメソッドへの呼び出しを 10 個までなら使うことができます。なぜなら、同時に使われる文字列が 10 個以下ならば、返される文字列群は独立であることが保証されているからです。返された文字列が有効なのは、次に何らかの API メソッドを呼び出す時までだけです：他の API メソッド呼び出しに引数として渡すこともしてはいけません。

Java・.NET・Python：結果は Unicode 文字列として与えられます。テキストが全く得られない場合には null オブジェクトが返されます。

Perl・PHP 言語バインディング：結果は UTF-8 文字列として与えられます。テキストが全く得られない場合には null オブジェクトが返されます。

C++ ***const unsigned char *pcos_get_stream(int doc, int *len, string optlist, string path)***

Java C# ***final byte[ ] pcos_get_stream(int doc, String optlist, String path)***

Perl PHP ***string pcos_get_stream(long doc, string optlist, string path)***

C ***const unsigned char *PDF_pcos_get_stream(PDF *p, int doc, int *len, const char *optlist, const char *path, ...)***

---

stream または *fstream* または**文字列**型の pCOS パスの内容を取得します。

doc *PDF_open_pdi_document( )* で取得した有効な文書ハンドル。

len （C・C++・RPG 言語バインディングのみ）返されるストリームデータの長さをバイト単位で格納させたい変数へのポインター。

optlist 表 10.7 に従ったストリーム取得オプションを指定したオプションリスト。

path ストリームまたは文字列オブジェクトへの完全な pCOS パス。

追加引数 （C 言語バインディングのみ）*key* 引数にプレースホルダを入れているときは（*%s* なら文字列、*%d* なら整数。*%%* を使うと 1 個のパーセント記号）、それに対応して可変の数の追加引数を与えることができます。これらの引数を使えば、可変の数値や文字列の値を入れて複雑なパスを明示的に構築する手間が省けます。プレースホルダの数と型を、与える追加引数と確かに一致させるのは、クライアント側の役割です。

戻り値 ストリームまたは文字列に入っていたデータを復号したもの。ストリームか文字列が空のとき、または、暗号化されていない文書の中の暗号化されている添付の内容がクエリーされてその添付パスワードが与えられていないときは、返されるデータは空（C++ では NULL）になります。

オブジェクトが *stream* 型のときは、*keepfilter=true* でなければ、すべてのフィルターがストリームの内容から除去されます（すなわち、生データ本体が返されます）。オブジェ

クトが *fstream* または**文字列**型のときは、データは PDF ファイル内で見つかったそのままで返されますが、ただし例外として ASCII85・ASCIIHex フィルターは除去されます。

詳細 この関数は、pCOS が完全モードで動作していないときは、例外を発生させます。ただし例外として、オブジェクト */Root/Metadata* は、*nocopy=false* か *plainmetadata=true* にしていれば、限定 pCOS モードでも取得できます。また、*path* が *stream* か *fstream* か**文字列**型のオブジェクトを指し示していないときも、例外が発生します。

この関数は、その名に合わず、**文字列**型のオブジェクトを取得するためにも使えます。*PDF_pcos_get_string( )* では、オブジェクトをテキスト文字列として扱いますが、それと違ってこの関数は、返されるデータにいかなる変更も加えません。バイナリー文字列データは、PDF 内で使われることはまれで、自動的に正しく検出することはできません。ですので、文字列オブジェクトをバイナリーデータとテキストのどちらとして取得するか、適切なメソッドを選ぶのはユーザー側の役割です。

スコープ 任意

バインディング C 言語バインディング：*convert=unicode* を与えると、文字列は BOM なし UTF-8 形式（IBM System i・IBM Z では EBCDIC-UTF8）で返されます。

C・C++ 言語バインディング：返されたデータバッファは、次にこのメソッドを呼び出すまで使えます。
Python：結果は 8 ビット文字列（Python 3：*bytes*）として返されます。

表 10.7 PDF_pcos_get_stream( ) のオプション

オプション	説明
convert	（キーワード。非サポートのフィルターで圧縮されているストリームでは無視されます）文字列またはストリーム内容が変換されるかどうかを制御します（デフォルト：none）： ***none*** 内容をバイナリーデータとして扱い、何の変換も行いません。 ***unicode*** 内容をテキストデータとして（すなわち、PDF_pcos_get_string( ) の場合と全く同様に）扱い、これを Unicode に正規化します。Unicode 非対応言語バインディングでは、データが BOM なしの UTF-8 へ変換されます。 このオプションは、PDF 内のまれにしか用いられないデータ型「テキストストリーム」（たとえば、これは JavaScript のために用いられる場合があります。しかし JavaScript の多数は、文字列オブジェクト内に格納され、ストリームオブジェクトには格納されません）のために必要です。
keepfilter	（論理値。画像データストリームに対してのみ推奨。非サポートのフィルターで圧縮されているストリームでは無視されます）true にすると、ストリームデータは、画像の filterinfo 擬似オブジェクト内で指定されているフィルターで圧縮されます。デフォルト：すべての非サポートフィルターでは true、それ以外の場合には false

11 ブロック流し込みメソッド（PPS）

この章の API メソッド：

- *PDF_fill_textblock()*
- *PDF_fill_imageblock()*
- *PDF_fill_pdfblock()*
- *PDF_fill_graphicsblock()*

PDFlib Personalization Server（PPS）は、**テキスト・イメージ・*PDF* 型**の可変ブロックを処理するためのメソッド群を提供します。これらの PDFlib ブロックは、取り込む PDF ページに入れておく必要がありますが、生成する出力には残りません。取り込んだページは、ブロック流し込みメソッド群を使う前に、出力ページに *PDF_fit_pdi_page()* で配置しておく必要があります。ブロックメソッド群は、ページ上でのブロックの位置を算出する際、取り込んだページが *PDF_fit_pdi_page()* で配置された時に効いていた拡縮オプションを考慮します。

注 この章で解説するブロック処理メソッド群は、PDFlib Personalization Server（PPS）を必要とします。PDF テンプレート内に PDFlib ブロックを対話的に作成するには、Adobe Acrobat 用 PDFlib Block Plugin が必要です。あるいは、ブロックを PPS 自体で作成することもできます。

11.1 すべてのブロック種別のためのオプション

この節では、*PDF_fill_textblock()*・*PDF_image_block()*・*PDF_fill_pdfblock()*・*PDF_graphics_block()* のオプションを記します。各ブロック種別に特有のオプションは次節以降に挙げます。ほとんどすべてのブロックプロパティは同名のオプションでオーバーライドできますが、ただし以下のプロパティだけはオプションでオーバーライドできません：

`Name`・`Description`・`Subtype`・`Type`

11.1.1 長方形オプション

表 11.1 に、*PDF_fill_textblock()*・*PDF_image_block()*・*PDF_fill_pdfblock()*・*PDF_graphics_block()* の長方形オプションを挙げます。

表 11.1 PDF_fill_*block( ) メソッドの長方形オプション

オプション	説明
background-color	（色）ブロックの塗り色。この色は、ブロックへの流し込みより前に適用されます。これは、既存のページ内容を隠すのに有用でしょう。デフォルト：none
bordercolor	（色）ブロックの境界色。この色は、ブロックへの流し込みより前に適用されます。デフォルト：none
linewidth	（float。0 より大きくする必要があります）ブロック長方形を描くのに用いる線の描線幅。`bordercolor` を設定しているときにのみ用いられます。デフォルト：1
Rect	（長方形）ブロックの座標をブロック PDF の座標系で指定します。ブロックの位置と寸法は `refpoint`・`boxsize` はめ込みオプションで指定できます（表 6.1 参照）。どちらのオプションも絶対値と相対値を受け付けます。

表 11.1 PDF_fill_*block( ) メソッドの長方形オプション

オプション	説明
Status	（キーワード）ブロックがどのように処理されるかを記述します（デフォルト：active）： ***active*** ブロックはそのプロパティ群に従って処理されます。 ***ignore*** ブロックは無視されます。 ***ignoredefault*** active と同様ですが、ただし、defaulttext/image/pdf プロパティは無視されます。すなわち、内容が与えられていなければブロックは空のままになります。これは、ブロックがたとえ Block Plugin でのプレビューのためのデフォルト内容を持っていたとしても、そのブロックのデフォルト内容がサーバー側でブロックへの流し込みに使われないようにするために有用でしょう。また、これを利用すれば、デフォルト内容を表示させずにブロックをプレビューしたい場合に、それをブロックのプロパティから削除しなくても済みます。 ***static*** 可変内容が一切配置されません。ブロックのデフォルトテキスト・イメージ・PDF 内容がもしあればそれが用いられます。

11.1.2 タグ付けオプション

表 11.2 で、*PDF_fill_textblock( )*・*PDF_image_block( )*・*PDF_fill_pdfblock( )*・*PDF_graphics_block( )* のタグ付けオプションを解説します。特定のブロック種別（すなわちテキスト・イメージ・PDF ブロック）に特有のオプションは次節以降に挙げます。

表 11.2 PDF_fill_*block( ) メソッドのタグ付けオプション

オプション	説明
tag	（オプションリスト）表 14.2 に従った、構造オプションのタグ付けのためのオプション群

11.2 テキスト行・テキストフローブロック

C++ Java C# ***int fill_textblock(int page, String blockname, String text, String optlist)***

Perl PHP ***int fill_textblock(int page, string blockname, string text, string optlist)***

C ***int PDF_fill_textblock(PDF *p,***
int page, const char *blockname, const char *text, int len, const char *optlist)

テキスト行またはテキストフローブロックへ、そのプロパティに従って、可変データを流し込みます。

page PDFlib ブロックを入れてあるページに対する有効な PDF ページハンドル。これより前に、ブロックを持った入力 PDF ページを、*PDF_fit_pdi_page( )* で直接、または *PDF_fit_table( )* で表セル内へ間接的に、あるいは *PDF_fill_pdfblock( )* で PDF ブロックの内容として、ページ上に配置しておく必要があります。

blockname （名前文字列）ブロックの名前。

text （内容文字列）ブロックへ流し込みたいテキスト。空文字列にすると、デフォルトテキスト（ブロックのプロパティで定義してある）を使えます。*textflowhandle* オプションを与えているときは、その中身が有効なテキストフローハンドルならば、この引数は無視されます。

len （C 言語バインディングのみ）*text* の長さ（バイト単位）。*len=0* にすると null 終了文字列を与える必要があります。

optlist テキストブロック流し込みオプションを指定したオプションリスト。以下のオプションが使えます：

- 一般オプション：*errorpolicy*（表 1.5 参照）
- 表 11.1 に従ったブロック流し込みメソッドのための長方形オプション：*Rect*・*Status*・*backgroundcolor*・*bordercolor*・*linewidth*
- はめ込みオプション（135 ページ「6.1 オブジェクトのはめ込み」参照）
- テキスト行ブロック、すなわち *textflow* プロパティまたはオプションが *false*：すべてのテキスト行オプション（99 ページ「5.1 テキスト行による一行テキスト」参照）。
- テキストフローブロック、すなわち *textflow* プロパティまたはオプションが *true*：*PDF_add/create_textflow( )* のすべてのオプション（106 ページ「5.2 テキストフローによる複数行テキスト」参照）と *PDF_fit_textflow( )* のすべてのオプション（表 5.13 参照）
- 表 11.3 に従ったテキストブロックオプション：*textflow*・*textflowhandle*
- デフォルト内容のためのオプション：*defaulttext*（PDFlib チュートリアル参照）

戻り値 指定した名前のブロックがページ上にないか、またはブロックへ流し込めないか（フォントの問題などにより）、またはブロックが誤った型を持っているか、またはブロックがもっと新しい PPS バージョンを必要とするときは -1（PHP では 0）。ブロックの処理が成功したときは 1。

textflowhandle オプションを与えている場合には、テキストフローハンドルが返されるので、以後の呼び出しで使えます。プロパティ *Status* の値が *ignore* の場合には、*textflowhandle=-1* であるなら、空のテキストフローへのハンドルが返されます。そうでないなら、与えたテキストフローハンドルが、そのテキストフローに変更を加えることなく（すなわち出力を一切生成せずに）返されます。返されたテキストフローハンドルを最後に *PDF_delete_textflow( )* を用いて削除するのはユーザー側の役割です。

PDF 文書が破損していることがわかったときは、このメソッドは *errorpolicy* オプションに従って、例外を発生させるか、-1 を返します。

詳細 与えたテキストが、ブロックのプロパティに従って、ブロックの中に組版されます。*text* を空にすると、このメソッドは、ブロックのデフォルトテキストがあればそれを使い（*Status=ignoredefault* でなければ）、それ以外の場合は警告を出さずに返ります。これは、塗り色や描線色など、他のブロックプロパティを活用するうえで有用です。

フォント選択：*font* オプションを与えておらず、オプションに基づく暗黙的フォント読み込みも用いられていないときは、フォントはブロックのプロパティ群に基づいて暗黙的に読み込まれます。

フォントに対するエンコーディングは、オプションとしてのみ指定することができ、ブロックプロパティとしては指定できませんので、デフォルトでは以下のように設定されます：

- フォントが記号フォントであり、かつ *charref=false*、かつ（Unicode 非対応言語についてのみ）*textformat=auto* か *bytes* ならば *builtin*
- それ以外の場合は *unicode*。

defaulttext を用いる場合には、*encoding*・*charref*・*textformat* オプションを避けることを推奨します。

テキストフローブロックを連結 テキストフローがブロックに収まりきらないときは、*textflowhandle* オプションを使って、複数のブロックを連結し、同じテキストフローの各部分をそれぞれに入れることもできます。

- 最初の呼び出しでは、値 -1（PHP では 0）を与える必要があります。内部的に作成されたテキストフローハンドルが *PDF_fill_textblock()* によって返されるので、ユーザー側でそれを保管する必要があります。
- 次の呼び出しでは、前の段階で返されたテキストフローハンドルを、*textflowhandle* オプションに与えることができます（*text* 引数で与えるテキストはこの場合無視されるので、空にするべきです）。ブロックへはテキストフローの残りが流し込まれます。
- この処理を、他のテキストフローブロックについても繰り返すことができます。
- 返されたテキストフローハンドルは、*PDF_info_textflow()* に与えて、テキストの終了位置等、ブロック流し込みの結果を知ることができます。

この処理を、任意の数のブロックに対して繰り返すことができます。

PDF/UA ブロック装飾、すなわち、*backgroundcolor*・*bordercolor*・*linewidth* プロパティに従って作成される罫線と背景色は、自動的に *Artifact* としてタグ付けされます。

スコープ ページ・パターン・テンプレート・グリフ

表 11.3 PDF_fill_textblock( ) の追加オプション

オプション	説明
textflow	（論理値）一行処理か複数行処理かを制御します。このプロパティを使うと、テキスト行ブロックとテキストフローブロックを切り替えることができます： ***false*** テキストは一行にわたり、PDF_fit_textline( ) で処理されます。 ***true*** テキストは複数行にわたり、PDF_fit_textflow( ) で処理されます。 デフォルトはブロックの種別に依存します：テキストフローブロックの場合は `true`、テキスト行ブロックの場合は `false`

表 11.3 PDF_fill_textblock( ) の追加オプション

オプション	説明
textflow-handle	（テキストフローハンドル。PDF_fill_textblock( ) で textflow=true の場合にのみ可）このオプションを用いると、テキストフローブロックどうしを連結することができます。連結ブロック群の最初のブロックに対しては値 –1（PHP では 0）を与える必要があります。このメソッドによって返された値を、それ以後の連結される他のブロック群に対する呼び出しでテキストフローハンドルとして与えることができます。このオプションに –1（PHP では 0）以外の値を与えると、fitmethod のデフォルトは clip へ変更されます。 ブロックのテキスト作成・テキスト組版・書式プロパティグループのすべてのプロパティは、textflowhandle が与えられている場合には無視されます。なぜなら、テキストフローを作成するために用いられた、それらに対応する値群が適用されるからです。

11.3 イメージブロック

C++ Java C# ***int fill_imageblock(int page, String blockname, int image, String optlist)***

Perl PHP ***int fill_imageblock(int page, string blockname, int image, string optlist)***

C ***int PDF_fill_imageblock(PDF *p,***
int page, const char *blockname, int image, const char *optlist)

イメージブロックへ、そのプロパティに従って、可変データを流し込みます。

page PDFlib ブロックを入れてあるページに対する有効な PDF ページハンドル。これより前に、ブロックを持った入力 PDF ページを、*PDF_fit_pdi_page()* で直接、または *PDF_fit_table()* で表セル内へ間接的に、あるいは *PDF_fill_pdfblock()* で PDF ブロックの内容として、ページ上に配置しておく必要があります。

blockname （名前文字列）ブロックの名前。

image ブロックへ流し込みたい画像に対する有効な画像ハンドル。-1 にすると、デフォルト画像（ブロックのプロパティで定義してある）を使えます。

optlist イメージブロック流し込みオプションを指定したオプションリスト。以下のオプションが使えます：

- 一般オプション：*errorpolicy*（表 1.5 参照）
- 表 11.1 に従ったブロック流し込みメソッドのための長方形オプション：
  Rect・*Status*・*backgroundcolor*・*bordercolor*・*linewidth*
- 流し込みオプション（135 ページ「6.1 オブジェクトのはめ込み」参照）
- 表 9.3 に従った画像処理のためのオプション。
- デフォルト内容のためのオプション：*defaultimage*（PDFlib チュートリアル参照）

戻り値 指定した名前のブロックがページ上にないか、またはブロックへ流し込めないか、またはブロックが誤った種別を持っているか、またはブロックがもっと新しい PPS バージョンを必要とするときは -1（PHP では 0）。ブロックの処理が成功したときは 1。問題の性質に関する情報をもっと得たいときは、*PDF_get_errmsg()* を使います。

詳細 与えた画像ハンドルが参照する画像が、ブロックのプロパティに従って、ブロックの中に配置されます。*image* を -1（PHP では 0）にすると、このメソッドは、ブロックのデフォルト画像があればそれを使い（*Status=ignoredefault* でなければ）、それ以外の場合は警告を出さずに返ります。

PDF 文書が破損していることがわかったときは、このメソッドは *errorpolicy* オプションに従って、例外を発生させるか、-1 を返します。

PDF/UA すべてのラスタ画像は、事前の *PDF_begin_item()* への呼び出しを用いて *Artifact* か *Figure* としてタグ付けしておく必要があります。

ブロック装飾、すなわち、*backgroundcolor*・*bordercolor*・*linewidth* プロパティに従って作成される罫線と背景色は、自動的に *Artifact* としてタグ付けされます。

スコープ ページ・パターン・テンプレート・グリフ

11.4 PDF ブロック

C++ Java C# ***int fill_pdfblock(int page, String blockname, int contents, String optlist)***

Perl PHP ***int fill_pdfblock(int page, string blockname, int contents, string optlist)***

C ***int PDF_fill_pdfblock(PDF *p, int page, const char *blockname, int contents, const char *optlist)***

PDF ブロックへ、そのプロパティに従って、可変データを流し込みます。

page PDFlib ブロックを入れてあるページに対する有効な PDF ページハンドル。これより前に、ブロックを持った入力 PDF ページを、*PDF_fit_pdi_page()* で直接、または *PDF_fit_table()* で表セル内へ間接的に、あるいは *PDF_fill_pdfblock()* で PDF ブロックの内容として、ページ上に配置しておく必要があります。

blockname （名前文字列）ブロックの名前。

contents ブロックへ流し込みたい PDF ページに対する有効な PDF ページハンドル。-1 にすると、デフォルト PDF ページ（ブロックのプロパティで定義してある）を使えます。

optlist PDF ブロック流し込みオプションを指定したオプションリスト。以下のオプションが使えます：

- 一般オプション：*errorpolicy*（表 1.5 参照）
- 表 11.1 に従ったブロック流し込みメソッドのための長方形オプション：
  Rect・*Status*・*backgroundcolor*・*bordercolor*・*linewidth*
- 流し込みオプション（135 ページ「6.1 オブジェクトのはめ込み」参照）
- 表 9.3 に従ったページ処理のためのオプション。
- デフォルト内容のためのオプション：*defaultpdf*・*defaultpdfpage*（PDFlib チュートリアル参照）

戻り値 指定した名前のブロックがページ上にないか、またはブロックへ流し込めないか、またはブロックが誤った種別を持っているか、またはブロックがもっと新しい PPS バージョンを必要とするときは -1（PHP では 0）。ブロックの処理が成功したときは 1。問題の性質に関する情報をもっと得たいときは、*PDF_get_errmsg()* を使います。

詳細 与えたページハンドル *contents* が参照する PDF ページが、ブロックのプロパティに従って、ブロックの中に配置されます。*contents* を -1（PHP では 0）にすると、このメソッドは、ブロックのデフォルト PDF ページがあればそれを使い（*Status=ignoredefault* でなければ）、それ以外の場合は警告を出さずに返ります。

PDF 文書が破損していることがわかったときは、このメソッドは *errorpolicy* オプションに従って、例外を発生させるか、-1 を返します。

PDF/UA ブロック装飾、すなわち、*backgroundcolor*・*bordercolor*・*linewidth* プロパティに従って作成される罫線と背景色は、自動的に *Artifact* としてタグ付けされます。

スコープ **ページ・パターン・テンプレート・グリフ**

11.5 グラフィックブロック

C++ Java C# **int fill_graphicsblock(int page, String blockname, int contents, String optlist)**

Perl PHP **int fill_graphicsblock(int page, string blockname, int contents, string optlist)**

C **int PDF_fill_graphicsblock(PDF *p, int page, const char *blockname, int contents, const char *optlist)**

グラフィックブロックへ、そのプロパティ群に従って、可変データを流し込みます。

page PDFlib ブロック群を含むページに対する有効な PDF ページハンドル。これより前に、ブロック群を持った入力 PDF ページを、*PDF_fit_pdi_page()* で直接、または *PDF_fit_table()* で表セル内へ間接的に、あるいは *PDF_fill_pdfblock()* で PDF ブロックの内容として、ページ上に配置しておく必要があります。

blockname （名前文字列）ブロックの名前。

contents ブロックへ流し込みたいグラフィックに対する有効なグラフィックハンドル、あるいはデフォルトグラフィック（ブロックプロパティ群によって定義されている通りの）を使用したいときは -1。

optlist グラフィックブロック流し込みオプションを指定したオプションリスト。以下のオプションを使えます：

- 一般オプション：*errorpolicy*（表 1.5 参照）
- 表 11.1 に従った、ブロック流し込みメソッド群に対する長方形オプション：*Rect*・*Status*・*backgroundcolor*・*bordercolor*・*linewidth*
- 流し込みオプション（135 ページ「6.1 オブジェクトのはめ込み」参照）
- 表 9.3 に従った、グラフィック処理のためのオプション
- デフォルト内容のためのオプション：*defaultgraphics*（PDFlib チュートリアル参照）

戻り値 指定した名前のブロックがページ上にないか、またはブロックへ流し込めないか、またはブロックが誤った種別を持っているか、またはブロックがもっと新しい PPS バージョンを必要とするときは -1（PHP では 0）。ブロックの処理が成功したときは 1。問題の性質に関する情報をもっと得たいときは、*PDF_get_errmsg()* を使います。

詳細 与えたグラフィックハンドルが参照するグラフィックが、ブロックのプロパティ群に従って、ブロックの中に配置されます。*graphics* を -1（PHP では 0）にすると、このメソッドは、ブロックのデフォルトグラフィックがあればそれを使い（*Status=ignoredefault* でなければ）、それ以外の場合は警告を出さずに返ります。

PDF 文書が破損していることがわかったときは、このメソッドは *errorpolicy* オプションに従って、例外を発生させるか、-1 を返します。

PDF/UA ブロック装飾、すなわち、*backgroundcolor*・*bordercolor*・*linewidth* プロパティに従って作成される罫線と背景色は、自動的に *Artifact* としてタグ付けされます。

スコープ ページ・パターン・テンプレート・グリフ

12 インタラクティブ機能

この章の API メソッド：

- *PDF_create_bookmark()*
- *PDF_create_annotation()*
- *PDF_create_field()*
- *PDF_create_fieldgroup()*
- *PDF_create_action()*
- *PDF_add_nameddest()*

12.1 しおり

C++ Java C# ***int create_bookmark(String text, String optlist)***

Perl PHP ***int create_bookmark(string text, string optlist)***

C ***int PDF_create_bookmark(PDF *p, const char *text, int len, const char *optlist)***

しおりを、さまざまなオプションに従って作成します。

text （ハイパーテキスト文字列）しおりに対するテキスト。

len （C 言語バインディングのみ）*text* の長さ（バイト単位）。*len=0* にすると null 終了文字列を与える必要があります。

optlist しおりのプロパティ群を指定したオプションリスト。以下のオプションが使えます：

- 一般オプション：*errorpolicy*（表 1.5 参照）
- 表 12.1 に従ったしおり制御オプション：
  action・*destination*・*destname*・*fontstyle*・*index*・*item*・*open*・*parent*・*textcolor*
- C の場合と、Perl・PHP・Ruby で *stringformat=legacy* の場合のためのエンコーディングオプション：
  hypertextencoding・*hypertextformat*（表 2.1 参照）

戻り値 作成したしおりのハンドル。以後の呼び出しの *parent* オプションで使えます。

詳細 このメソッドは、与える *text* を持つ PDF しおりを追加します。*destination* オプションを指定しないと、しおりはカレントページ（**文書**スコープの中で使うと一番最近に作成されたページ、先頭ページより前に使うと先頭ページ）を指します。

しおりを作成すると、*PDF_begin/end_document()* の *openmode* オプションが、たとえ他のモードを明示的に設定してあっても、*bookmarks* に設定されます。

PDF/UA PDF/UA-1 ではしおりを作成することが推奨されます。

スコープ **文書・ページ**：*target* オプションで作成した表現アクションを与える場合、このメソッドを呼び出せるのは**ページ**スコープ内、かつ、そのターゲット注釈が位置しているページにおいてだけです。

表 12.1 PDF_create_bookmark( ) のオプション

オプション	説明
action	（アクションリスト）以下のイベントに対するしおりアクションのリスト（デフォルト：destination オプションで指定している移動先への GoTo アクション）： ***activate*** しおりがアクティブにされた時に実行させたいアクション。あらゆる種別のアクションが使えます。
destination	（オプションリスト。activate アクションを指定している場合には無視されます）表 12.11 に従った、しおりの移動先。デフォルト：destination・destname・action を指定していないときは {type fitwindow page 0}。
destname	（ハイパーテキスト文字列。空でも可。destination オプションを指定している場合には無視されます）PDF_add_nameddest( ) で定義してある移動先の名前。destination・destname アクションはこのオプションよりも優先されます。destname を空文字列（すなわち {}）にすると、destination と action のどちらも指定していないときは、しおりは全くアクションを持ちません。これは区切りしおりのために有用でしょう。
fontstyle	（キーワード）しおりテキストのフォントスタイル：normal・bold・italic・bolditalic。デフォルト：normal
index	（整数）しおりを親の中に挿入したい位置の番号。0 から、同じ階層のしおりの数までの値を使って、しおりを親の中のその位置に挿入します。値 -1 を使うと、しおりをカレント階層の末尾に挿入することができます。デフォルト：-1。ただし、挿入・再開したページについては、すべてのページを物理的な順序どおりに作成したかのようにしおりが配置されます（すなわち、しおりがページ順序を反映します）。
item	（アイテムハンドルかキーワード。このハンドルは、アクティブな構造エレメントを参照している必要がありますが、インラインまたは疑似エレメントを参照していてはいけません。タグ付き PDF のみ）このしおりに紐付けられる構造アイテムに対するハンドル。値 0 はつねに構造ツリールートを参照します。値 1 と、これに等価なキーワード current は、カレントでアクティブなエレメントを参照します。
open	（論理値）false にすると、子しおりは表示されません。true にすると、すべての子が展開表示されます。デフォルト：false
parent	（しおりハンドル）新規しおりが、ハンドルで指定するしおりの子であることを指定します。parent=0 にすると、新たな最上位階層のしおりが作成されます。デフォルト：0
textcolor	（色）しおりテキストの色。使える色空間：none・gray・rgb。 デフォルト：rgb {0 0 0 }（＝黒）

12.2 注釈

C++ Java C# ***void create_annotation(double llx, double lly, double urx, double ury, String type, String optlist)***

Perl PHP ***create_annotation(float llx, float lly, float urx, float ury, string type, string optlist)***

C ***void PDF_create_annotation(PDF *p,***
double llx, double lly, double urx, double ury, const char *type, const char *optlist)

注釈を、カレントページ上に作成します。

llx・lly・urx・ury 注釈の長方形の左下隅と右上隅の *x・y* 座標を、デフォルト座標で（*usercoordinates* オプションを *false* にしたとき）、またはユーザー座標で（*usercoordinates* を *true* にしたとき）指定します。

注釈の座標は、*PDF_rect()* 関数の引数とは違うので注意してください。すなわち、*PDF_create_annotation()* では 2 個の隅の座標をとるのに対し、*PDF_rect()* では、1 個の隅の座標とそれから幅と高さを指定します。

usematchbox オプションを指定しているときは、*llx/lly/urx/ury* は無視されます。注釈長方形は、注釈種別とオプションの組み合わせによっては、若干拡大されることがあります。

type 表 12.2 に従った注釈種別。マークアップ注釈には表内で特記しています。オプションのなかにはマークアップ注釈にのみ適用されるものがあるからです。

表 12.2 注釈の種別

種別	マークアップ注釈[1]	この種別に関連するオプション（共通のオプションに加えて）
3D		（PDF 1.6。3D アートワークは RichMedia 注釈を用いて作成することもでき、またそちらのほうが 3D 注釈よりも多くの機能を提供します）3D モデル：3Dactivate・3Ddata・3Dinteractive・3Dshared・3Dinitialview
Caret	○	（PDF 1.5）rectdiff・symbol
Circle	○	cloudy・createrichtext・inreplyto・interiorcolor・replyto
File-Attachment	○	***attachment***・createrichtext・iconname・inreplyto・replyto
FreeText	○	（テキスト注釈と異なり、テキストが常に表示されていて、ポップアップを必要としません）alignment・calloutline・cloudy・createrichtext・endingstyles・fillcolor・font・fontsize・inreplyto・orientate・replyto
Highlight	○	createrichtext・inreplyto・polylinelist・replyto
Ink	○	createrichtext・inreplyto・polylinelist・replyto
Line	○	captionoffset・captionposition・createrichtext・endingstyles・font・interiorcolor・inreplyto・leaderlength・leaderoffset・line・showcaption・replyto
Link		destination・destname・highlight
Polygon	○	（PDF 1.5。頂点群を直線で結んだもの）：cloudy・createrichtext・inreplyto・interiorcolor・polylinelist・replyto

表 12.2 注釈の種別

種別	マークアップ注釈[1]	この種別に関連するオプション（共通のオプションに加えて）
PolyLine	○	（PDF 1.5。多角形に似ていますが、最初と最後の頂点が結ばれないことが違います）createrichtext・endingstyles・inreplyto・interiorcolor・polylinelist・replyto
Popup		open・parentname
RichMedia		（PDF 1.7ext3。動画・音声には Screen 注釈を推奨します）richmedia
Screen		（PDF 1.5）メディアの詳細については type=Rendition のアクションで指定します。
Square	○	cloudy・createrichtext・inreplyto・interiorcolor・replyto
Squiggly	○	（波型下線注釈）createrichtext・inreplyto・polylinelist・replyto
Stamp	○	createrichtext・iconname・inreplyto・orientate・replyto スタンプのテキストは、オプション createrichtext・contents・iconname・template のいずれかを用いて作成できます。これらのオプションをどれも与えていない場合には iconname=draft がデフォルトとして使用されます。
StrikeOut	○	createrichtext・inreplyto・polylinelist・replyto
Text	○	（テキストが別窓に表示され、この別窓は開いたり閉じたりできます）Acrobat ではこの種別はノート注釈と呼ばれます）createrichtext・iconname・inreplyto・open・replyto・state・statemodel
Underline	○	createrichtext・inreplyto・polylinelist・replyto

1. Popup 注釈といくつかのオプションで意味を持ちます（表 12.9 参照）。

optlist 注釈のプロパティ群を指定したオプションリスト：

- 表 12.3 に従った共通オプションがすべての注釈種別で使えます：
  action・*annotcolor*・*associatedfiles*・***blendmode***・*borderstyle*・*cloudy*・*contents*・*createdate*・*custom*・*dasharray*・*datestring*・*display*・*lang*・*layer*・*linewidth*・*locked*・*lockedcontents*・*name*・*opacity*・*opacityfill*・*readonly*・*rotate*・*subject*・*template*・*title*・*usematchbox*・*usercoordinates*・*zoom*
- 表 12.2 に従ったいくつかの注釈種別に対する、表 12.3 に従った種別特有オプション：
  alignment・*calloutline*・*createrichtext*・*destname*・*endingstyles*・*fillcolor*・*font*・*fontsize*・*highlight*・*iconname*・*inreplyto*・*interiorcolor*・*open*・*orientate*・*parentname*・*polylinelist*・*rectdiff*・*replyto*・***richmedia***・*state*・*statemodel*・*symbol*
- 表 12.4 に従った、*type=Line* に対するオプション：
  captionoffset・*captionposition*・*leaderlength*・*leaderoffset*・*line*・*showcaption*
- 表 13.12 に従った、*type=3D* に対するオプション：
  3Dactivate・*3Ddata*・*3Dinteractive*・*3Dshared*・*3Dinitialview*
- 表 14.4 に従った、短縮構造エレメントタグ付けのためのオプション：*tag*
- C の場合と、Perl・PHP・Ruby で ***stringformat=legacy*** の場合に対するエンコーディングオプション：*hypertextencoding*（表 2.1 参照）

詳細 このメソッドは、与えられた枠座標を用いて、または *matchbox* オプションで与えられた枠を用いて、カレントページ上に注釈を作成します。

たいていの種類の注釈は、その長方形の各辺がページの各辺と並行でなければなりませんが、以下の種類の注釈では、回転または斜形化されることも可能です（たとえば座標系が回転されている場合などに）：*Highlight*・*Link*・*Squiggly*・*StrikeOut*・*Underline*。

以下の注釈種別は、*PDF_create_annotation()* で使うことができませんが、ただし、他の PDF 文書から *PDF_fit_pdi_page()* を用いて取り込むことはできます：*Movie*・*Projection*・*Redact*・*Sound*。

font または *template* オプションを与えない場合には、*type=Caret*・*FreeText*・*Line*（*showcaption=true* の場合に限る）に対してはフォント *NotoSans-Regular* を構成する必要があり、*type=Stamp* に対してはフォント *NotoSans-Bold* を構成する必要があります。どちらのフォントも PDFlib ディストリビューションに含まれています（PDFlib チュートリアル参照）。これらのフォントは日中韓グリフを含んでいませんので、*type=FreeText* または *Line* で日中韓テキストを使用する場合には、然るべき日中韓フォントを *font* オプションに与えるべきです。

タグ付き PDF モードでは、このメソッドは、注釈に対して *OBJR* エレメントを自動的に作成します。ユーザーは、このメソッドを呼び出す前に、対応する *Link* または *Annot* コンテナエレメント（PDFlib チュートリアル参照）を作成する必要があります。

PDF/A PDF/A-1：*type=FileAttachment*・*Highlight* は許されません。以下の種別は、RGB 出力インテントを与えている場合のみ許されます：*Freetext*・*Text*・*Stamp*・*Underline*・*Squiggly*・*Strikeout*・*Polygon*・*Polyline*・*Line*・*Square*・*Circle*・*Ink*。
PDF/A-2/3：*type=3D*・*Screen* は許されません。

オプション *annotcolor*・*fillcolor*・*interiorcolor* に対する色の制約：
PDF/A-1/2/3：グレースケールカラーは、出力インテント（種別を問わず）を指定している場合にのみ許されます。CMYK カラーは、CMYK 出力インテントを指定している場合にのみ許されます。

PDF/A-1a/2a/3a：テキストを表示しない注釈に対しては *contents* オプションを推奨します。
いくつかのオプションは制約されています。表 12.3 を参照してください。

PDF/UA *type=Link* の注釈は *Link* 構造エレメントの内容とする必要があります。他のすべての種類の注釈は、*Popup* を除き、*Annot* 構造エレメントの内容とする必要があります。

可視注釈に対しては、オプション *contents* かオプション *tag* にサブオプション *ActualText* を付ける必要があります。

PDF/X 注釈は、完全に BleedBox（BleedBox がないときは TrimBox/ArtBox）の外に置く場合のみ許されます。以下の種別については、注釈の長方形を、少なくとも線幅の半分は、関連するページ枠の外に置く必要があります：*Square*・*Circle*・*Ink*・*PolyLine*・*Polygon*。

Line 注釈のキャプションは、完全の長方形の内部に置く必要があります。
いくつかのオプションは制約されています。表 12.3 を参照してください。
PDF/X-3：*type=Caret*・*FileAttachment*・*Highlight* は許されません。
PDF/X-4/5：*type=Caret*・*Highlight*・*Screen* は許されません。

スコープ ページ

表 12.3 PDF_create_annotation( ) のオプション

オプション	説明
action	（アクションリスト。type=Popup では不可）以下のイベントに対する注釈アクションのリスト（デフォルト：空リスト）。あらゆる種類のアクションが使えますが、ただし表現アクションは type=Screen の場合にのみ許されます： ***activate*** （type=Link・Screen でのみ可）その注釈がアクティブにされた時に実行させたいアクション。 ***close*** （PDF 1.5）ページが閉じられた時に実行させたいアクション。 ***invisible*** （PDF 1.5）ページがもう見えなくなった時に実行させたいアクション。 ***open*** （PDF 1.5）ページが開かれた時に実行させたいアクション。 ***visible*** （PDF 1.5）ページが見えた時に実行させたいアクション。
alignment	（キーワード。type=FreeText でのみ可）注釈の中のテキストの整列：left・center・right。デフォルト：left
annotcolor	（色。type=Popup では不可。上述のとおり PDF/A では色の制約が課せられます）注釈のアイコンが閉じている時の背景と、注釈のポップアップウインドウのタイトルバーと、マークアップと、リンク注釈の枠の色。使える色空間：none・gray・rgb・（PDF 1.6 で）cmyk。デフォルト：注釈種別によっていろいろな RGB カラー
associatedfiles	（アセットハンドルのリスト。PDF 2.0・PDF/A-3 でのみ可）注釈に連携させたいファイル群に対するアセットハンドル群。このファイル群は、PDF_load_asset( ) で type=attachment を用いて読み込まれている必要があります。
attachment	（アセットハンドル。type=FileAttachment でのみ可。必須）ファイル添付。このファイル添付は、PDF_load_asset( ) で type=attachment を用いて読み込まれている必要があります。
blendmode	（キーワード。PDF 2.0）透過操作に対するブレンドモード（表 7.2 の blendmode を参照）。デフォルト：None
borderstyle	（キーワード）注釈の枠と、type=Polygon・PolyLine・Line・Square・Circle・Ink での線のスタイル：solid・beveled・dashed・inset・underline。ただし、beveled・inset・underline スタイルは Acrobat で正しく働きません。デフォルト：solid
calloutline[1]	（float 4 個か 6 個のリスト。PDF 1.6。type=FreeText でのみ可。PDF/X モードでは許されません）注釈に付ける引き出し線を指定した 4 個か 6 個の float 値。6 個の座標 {x1 y1 x2 y2 x3 y3} は、線の開始・折れ点・終了点（これはたいてい注釈の枠の上に配置されます）を表します。4 個の座標 {x1 y1 x2 y2} は、線の開始・終了点を表します。開始点は、endingstyles オプションの 1 番目のキーワードで指定した記号で修飾されます。
cloudy	（範囲 0 ～ 2 の float。PDF 1.6 では type=FreeText でのみ可であり、PDF 1.5 では type=Square でのみ可。PDF/X の場合には不可）多角形の変形に用いられる「雲型」効果の強度を、0（効果なし）から 2（最大の効果）で。このオプションは、非常に小さい注釈長方形に対しては予期しない効果を及ぼすことがあります。このオプションを用いると、borderstyle オプションは無視されます。デフォルト：0
contents[2]	（type=FreeText の場合と、Line で showcaption=true の場合には文字列、それ以外の場合にはハイパーテキスト文字列。type=Popup では不可。PDF/UA-1：tag オプションで ActualText を与えていない場合には必須、また type=Link ではつねに必須）注釈で表示させたいテキストか、または（テキストを表示しない注釈の場合）注釈の内容の人間が読める形での代替説明。キャリッジリターンまたはラインフィードキャラクターを使うと、改行を強制することができます。
createdate	（論理値。PDF 1.5 以上）true にすると、注釈に対して日付・時刻項目が作成されます。デフォルト：マークアップ注釈では true、そうでないなら true

表 12.3 PDF_create_annotation( ) のオプション

オプション	説明
createrich-text[2]	（オプションリスト。マークアップ註釈でのみ可。PDF/A・PDF/X モードでは不可。PDF 1.5）テキストフローからリッチテキスト内容を作成します。これは、組版されたテキストを注釈のために作成するのに有用でしょう（テキストフローの一部をページ上に配置し、一部を注釈内に配置することもできます）。ここで与えるリッチテキストは、contents オプションを与えている場合にはそれと等しくするべきです。 ***textflow*** （テキストフローハンドル。必須）注釈にリッチテキストとして付けるテキストフロー。このテキストフローの中の色指定は、グレースケールまたは RGB カラーのみを使用している必要があります。このテキストフローハンドルを、PDF_create_annotation( ) への呼び出しの前に PDF_fit_textflow/table( ) に与えていた場合は、残りのテキストだけが注釈に用いられます。テキストがそれ以上得られないときは、テキストフローが先頭から再開されます。テキストフローを注釈に用いても、その後の PDF_fit_textflow/table( ) への呼び出しには影響を与えません。 ***userunit*** （キーワード）スカラー値（firstlinedist・fontsize 等）に対する長さ単位：cm（センチメートル）・in（インチ）・mm（ミリメートル）・pt（ポイント）。デフォルト：pt リッチテキストを作成する際には、以下のテキストフローオプション群が効力を持ち、これ以外はすべて無視されます：nextline・alignment・fillcolor・underline・strikeout・font・fontsize・textrise・テキスト組版オプション群 リッチテキストは、XML 言語の一種である XFA 形式で出力されます。ですので、注釈リッチテキストのためのテキストフローの中に XML 文法キャラクターがあるときは、それらを < など適切にクォートする必要があります。ただこれは、そのテキストフローを PDF_fit_textflow( ) と PDF_create_annotation( ) に使っていると、どこが両者の境目かたいてい判定しようがなく厄介です（テキストフロー向けにはキャラクターをクォートする必要はないが、注釈用 XFA 向けには必要）。font と alignment を設定しても、Acrobat では期待通りには動作しません。OpenType 機能はリッチテキスト内では表現できないので、テキストフローにおいてテキストオプション features={_none} を指定することによって OpenType 機能を無効化することを推奨します。
custom	（オプションリストのリスト）カスタム注釈項目群を作成します。これは、NexPress 出力機のための処理命令を挿入するなどの応用において有用でしょう。各オプションリストは注釈辞書内の各項目を記述します： ***key*** （文字列。必須）辞書キーの名前。あらゆる非標準の PDF キーのほか、以下の標準キーも指定できます：Contents・Name（iconname オプション）・NM（name オプション）・Open。 ***type*** （キーワード。必須）その値の型：boolean・name・string のいずれか ***value*** （type=string にしているときはハイパーテキスト文字列、そうでなければ文字列。必須）項目の値
dasharray	（非負 float 2 個のリスト。borderstyle=dashed の場合にのみ可）破線枠の短線と間隙の長さ。デフォルト：3 3
datestring	（文字列。強制的に createdate=true になります。type=Popup では不可）その注釈に対して設定したい作成日の PDF 日付文字列。この日付文字列はその注釈辞書に変更なく書き込まれます。無効な日付文字列を与えることはエラーです。デフォルト（createdate=true の場合のみ意味を持ちます）：現在の日付
destination	（オプションリスト。type=Link でのみ可。activate アクションを指定している場合には無視されます）リンクの移動先を表 12.11 に従って定義したオプションリスト
destname	（ハイパーテキスト文字列。type=Link でのみ可。destination オプションを指定している場合には無視されます）PDF_add_nameddest( ) で定義しておいた移動先の名前。PDF_create_action( ) の destination または destname オプションで作成したアクションは、このオプションよりも優先されます。

表 12.3 PDF_create_annotation( ) のオプション

オプション	説明
display	（キーワード。PDF/A では visible のみ許されます）画面と紙の上での表示・非表示。visible・hidden・noview・noprint。デフォルト：visible
endingstyles	（キーワードのリスト。type=FreeText・Line・PolyLine でのみ可）線端スタイルを指定したキーワード 2 個のリスト。type=FreeText では 2 番目のキーワードは無視されます（デフォルト：{none none}）： butt（PDF 1.5）・circle・closedarrow・diamond・none・openarrow・rclosedarrow（PDF 1.5）・ropenarrow（PDF 1.5）・slash（PDF 1.6）・square
fillcolor	（色。type=FreeText でのみ可。上述のとおり PDF/A では色の制約が課せられます）テキストの枠線色と塗り色。使える色空間：none・gray・rgb・（PDF 1.6 で）cmyk。デフォルト：{rgb 0 0 0}
font	（フォントハンドル。type=Caret・FreeText・Line・Stamp でのみ可）注釈で使いたいフォント。デフォルト：type=Stamp の場合には NotoSans-Bold、そうでないなら NotoSans-Regular。これらのフォントへのアクセスを構成する必要があります。
fontsize[1]	（文字サイズ。type=Caret・FreeText・Line でのみ可。必須）文字サイズ。値 0 またはキーワード auto を指定すると、文字サイズが長方形に合わせて調節されることを意味します。
highlight	（キーワード。type=Link でのみ可）ユーザーが注釈をクリックした時の、その注釈のハイライトモード：none・invert・outline・push。デフォルト：invert
iconname[2]	（文字列。type=Text・Stamp・FileAttachment でのみ可）注釈を表示するためのアイコンの名前（目に見えるアイコンを一切持たない注釈を作成するには、opacity=0 と設定します）： type=Text の場合（デフォルト：note）： checkmark・circle・comment・cross・crosshairs・help・insert・key・newparagraph・note・paragraph・rightarrow・rightpointer・star・uparrow・upleftarrow type=Stamp の場合（デフォルト：draft。オプションを用いると任意のスタンプテキストを作成できます）： approved・asis・confidential・departmental・draft・experimental・expired・final・forcomment・forpublicrelease・notapproved・notforpublicrelease・sold・topsecret type=FileAttachment の場合（デフォルト：pushpin）： graph・paperclip・pushpin・tag
inreplyto	（ハイパーテキスト文字列。PDF 1.5。マークアップ注釈でのみ可。replyto を与えている場合には必須）この注釈の返信先である注釈の名前（name オプション参照）。両方の注釈が同一ページ上にある必要があります。2 個の注釈の間の関係を、replyto オプションを用いて指定することもできます。
interiorcolor	（色。type=Line・Polygon・PolyLine・Square・Circle でのみ可。上述のとおり PDF/A では色の制約が課せられます）注釈の内部領域のための色。使える色空間：none・gray・rgb・（PDF 1.6 で）cmyk。デフォルト：none
lang	（文字列。PDF 2.0）注釈の自然言語を、表 3.3 でオプションについて記述しているフォーマットで。言語指定のフォーマットは PDF_begin_document( ) の lang オプションと同一です（表 3.3 参照）。
layer	（レイヤーハンドル。PDF 1.5）注釈を属させたいレイヤー。注釈が可視になるのは、そのレイヤーを可視にしているときだけになります。
linewidth	（float）注釈種別 Line・PolyLine・Polygon・Square・Circle・Ink の注釈の枠または線の幅を、デフォルト単位で指定します。linewidth=0 にすると、枠は見えなくなります。デフォルト：1
locked	（論理値）true にすると、注釈のプロパティ（位置・寸法等）を編集できなくなります。ただしこの場合でも、その内容は変更できます。デフォルト：false

表 12.3 PDF_create_annotation( ) のオプション

オプション	説明
locked-contents	（論理値。PDF 1.7）true にすると、注釈の内容を編集できなくなります。ただしこの場合でも、注釈を削除することと、そのプロパティ（位置・寸法等）を変更することは可能です。デフォルト：false
name	（ハイパーテキスト文字列。type=Screen では必須）注釈の名前。ページ上のすべての注釈の中で一意である必要があります。この名前は、注釈を以下の状況で使用する場合には必須です： ▸ PDF_create_annotation( )：オプション inreplyto・parentname ▸ PDF_create_action( ) で type=Hide の場合：オプション namelist ▸ PDF_create_action( ) で type=RichMediaExecute か GoTo3DView か Rendition の場合：オプション target ▸ PDF_create_action( ) で type=GoToE の場合：オプション targetpath でサブオプション annotation 文書に署名を行うつもりの場合にはこのオプションを推奨します。
opacity	（float またはパーセント値。PDF/A-1 の場合には不可。PDF/VT-1 では使用するべきではありません。PDF 1.7ext8 およびそれ以前ではマークアップ注釈に対してのみ可。PDF 2.0 ではすべての注釈すべてに対して可）注釈を描く際の描線・（opacityfill が指定されていない場合には）塗り操作の不透明度（0 ～ 1 または 0% ～ 100%）。デフォルト：1
opacityfill	（float またはパーセント値。PDF 2.0）注釈を描く際の塗り操作の不透明度。デフォルト：1
open	（論理値。type=Text・Popup でのみ可）true にすると、注釈ははじめ開いた状態で表示されます。デフォルト：false
orientate	（キーワード。type=FreeText・Stamp でのみ可）注釈の、その長方形の中での向き。使えるキーワード：north（直立）・east（右倒し）・south（上下逆さま）・west（左倒し）。デフォルト：north
parentname	（文字列。type=PopUp でのみ可、かつこの場合には必須）Popup 注釈の親注釈の名前。この名前は、FreeText 以外の種別のマークアップ注釈を指している必要があります。 このオプションが与えられた場合には、ポップアップ注釈の contents・annotcolor・title は、親注釈から継承されます。
polylinelist[1]	（折れ線か四辺形のリスト。type=Polygon・PolyLine・Ink・Highlight・Underline・Squiggly・Strikeout でのみ可）デフォルト：注釈長方形の頂点群を結んだ折れ線。 type=Polygon・PolyLine・Ink の場合：1 個または複数の点から成る折れ線 1 個を内容とするリスト。 その他の種別：リストに、float 値を 8 個ずつ持ったサブリストを n 個入れて、n 個の四辺形を指定します。四辺形はそれぞれ、注釈の対象としたいテキストの単語、ないし連続する複数の単語を囲みます。四辺形はジグザグ順（右上・左上・右下・左下）で与える必要があります。
readonly	（論理値）true にすると、注釈がユーザーの操作に反応することを許しません。注釈は、表示と印刷は許されますが、マウスクリックに反応したり、マウスの動きに反応して表示を変えたりすることはできません。デフォルト：false
rectdiff	（長方形。type=Caret でのみ可）注釈長方形と背後のキャレットの境界との距離を記述する 4 個の非負数。
replyto	（キーワード。PDF 1.6。マークアップ註釈のみ）この注釈と、inreplyto オプションで指定した注釈との間の関係（返信種別）（デフォルト：reply）： ***reply*** 注釈は、inreplyto で指定した注釈に対する返信と見なされます。 ***group*** 注釈は、inreplyto で指定した注釈とグループ化される必要があります。
richmedia	（オプションリスト。type=RichMedia でのみ可、かつその場合には必須）表 13.5 に従ったリッチメディアオプション群

表 12.3 PDF_create_annotation( ) のオプション

オプション	説明
rotate	（論理値。テキスト注釈では PDF/A では true に設定してはいけません）true にすると、ページの回転に合わせて注釈を回転させます。そうでなければ、注釈の回転は固定されたままになります。このオプションは、テキスト注釈のアイコンでは無視されます。デフォルト：PDF/A におけるテキスト注釈では false、それ以外では true
state	（文字列。type=Text でのみ可。PDF 1.5）inreplyto オプションで指定した注釈を設定したいステート。 statemodel=Marked の場合のためのキーワード（デフォルト：Unmarked）：Marked・Unmarked statemodel=Review の場合のためのキーワード（デフォルト：None）：Accepted・Rejected・Cancelled・Completed・None
statemodel	（文字列。state オプションを与えている場合には必須。type=Text でのみ可。PDF 1.5）使えるモデル： ***Marked*** ステート値 Marked・Unmarked を使えます ***Review*** ステート値 Accepted・Rejected・Cancelled・Completed・None を使えます
subject	（ハイパーテキスト文字列。PDF 1.5）注釈が述べている主題の簡単な説明
symbol	（キーワード。type=Caret でのみ可）キャレット記号の種類（デフォルト：none）： ***paragraph*** 段落記号 ***none*** 記号なし
template[2]	（オプションリスト。type=Popup では不可）注釈の各ステータスの体裁： ***normal/rollover/down*** （テンプレートハンドル。PDF/A-3 モードでは rollover・down は許されません）注釈の通常・マウスロールオーバー・マウスボタン押下時のためのテンプレート。normal の場合のデフォルト：内部的に生成される体裁。rollover・down の場合のデフォルト：体裁なし ***fitmethod*** （キーワード）テンプレートを注釈長方形内にはめ込む方式。fitmethod を entire 以外にすると、注釈長方形はテンプレート枠に合わせて調節されます（デフォルト：entire）： ***nofit*** テンプレートを置くだけです。拡縮も切り抜きも行われません。 ***meet*** テンプレートを position オプションに従って置き、長方形内にまるごと収まるよう、縦横比を保って拡縮します。一般に、テンプレートの少なくとも 2 つの辺が、対応する長方形の辺と重なることになります。 ***entire*** テンプレートを position オプションに従って置き、かつ、長方形全体を覆うように拡縮します。一般に、この方式ではテンプレートはつぶされます。 ***position*** （float かキーワードのリスト）テンプレート内の左下隅を {0 0}、右上隅を {100 100} としたときの、参照点 (x, y) の位置を指定した 1 個か 2 個の値。これらの値は、テンプレートの幅と高さに対するパーセント値として表します。両方のパーセント値が等しいときは、1 個の float 値だけを与えれば充分です。 キーワード left・center・right（x 方向で）または bottom・center・top（y 方向で）を、値 0・50・100 と同等なものとして用いることができます。キーワードを 1 個だけ指定すると、もう 1 つの方向でそれに対応するキーワードが追加されます。デフォルト：{left bottom}。
title	（ハイパーテキスト文字列。type=Popup では不可）注釈のポップアップウインドウが開いていてアクティブな時にタイトルバーの中のラベル。この文字列は Acrobat の作成者フィールドに対応します。最大長は、シングルバイトなら 255 キャラクター、Unicode なら 126 キャラクターです。ただし title の実用上の上限としては、32 キャラクターを推奨します。デフォルト：なし

表 12.3 PDF_create_annotation( ) のオプション

オプション	説明
usematchbox	（文字列のリスト。範囲枠が構造エレメントとともに作成されている場合には tag オプションと組み合わせることはできません）引数 llx/lly/urx/ury を無視して、かわりに名前付き範囲枠を用います。オプションリストの中の 1 番目の要素は、範囲枠を指定する名前文字列です。2 番目の要素は、使いたい長方形の番号（1 から始まる）を指定する整数か、またはキーワード all で、選んでいる範囲枠のすべての長方形を指定します。2 番目の要素を指定しないときは、デフォルトとして all になります。複数の範囲枠の長方形を選んだ場合にはそれぞれの長方形について別々の注釈が作成されます。 範囲枠か、または指定した長方形が、ページに存在しないときは、メソッドは注釈を作成せずに、警告を出さずに返ります。範囲枠を用いて表セル内に注釈を作成する場合には、PDF_fit_table( ) の後に PDF_create_annotation( ) を呼び出す必要があります。
user-coordinates	（論理値）false にすると、注釈の座標と文字サイズはデフォルト座標系で表されていると見なされます。そうでなければ、カレントユーザー座標系が用いられます。デフォルト：グローバル usercoordinates オプションの値
zoom	（論理値。テキスト注釈では PDF/A では true に設定してはいけません）true にすると、ページの表示倍率に合わせて注釈を拡縮させます。そうでなければ、注釈のサイズは固定されたままになります。このオプションは、テキスト注釈のアイコンでは無視されます。デフォルト：PDF/A におけるテキスト注釈では false、それ以外では true

1. 座標は、デフォルト座標（usercoordinates オプションが false の場合）かユーザー座標（それが true の場合）で解釈されます。
2. type=Stamp の場合には、オプション createrichtext・contents・iconname・template のうち指定できるのは 1 つだけです。

表 12.4 PDF_create_annotation( ) で type=Line の場合の追加のオプション

オプション	説明
captionoffset	（float 2 個。PDF 1.7）キャプションテキストの通常位置からの変位。1 番目の値は、注釈線の中点からの、線に沿った横変位を指定し、正の値は右への変位を、負の値は左への変位を表します。2 番目の値は、注釈線に垂直な縦変位を指定し、正の値は上への変位を、負の値は下への変位を表します。デフォルト：{0, 0}、すなわち通常位置からの変位なし
caption-position	（キーワード。PDF 1.7。showcaption=false の場合には無視されます）注釈のキャプション位置（デフォルト：Inline）： ***Inline*** キャプションは線の内側で中央揃えされます。 ***Top*** キャプションは線の上方に置かれます。
leaderlength	（float 1 個か 2 個のリスト。2 番目の float は負の値にしてはいけません。PDF 1.6）補助線の長さを、デフォルト座標で（usercoordinates オプションを false にしたとき）、またはユーザー座標で（true にしたとき）指定します。補助線とは、線の各終点から、その線自体に垂直に引く追加の線群です。この長さは 2 個の数値で指定します（デフォルト：{0 0}）： 1 番目の数値は、線の両端から、その線自体に垂直に引く線の長さです。正の値は、線を始点から終点（line オプションで指定している）へたどったときに時計回りである向きに補助線を引くことを意味し、負の値はその反対の向きを表します。 2 番目の値は、省略することもでき、その線の反対側に引く補助線の延長の長さを表します。1 番目の値を 0 にすると、正の値は無視されます。
leaderoffset	（非負 float。PDF 1.7）補助線のオフセットの長さ、すなわち、注釈の端点と補助線の始点との間のアキ幅。デフォルト：0
line[1]	（線。必須。PDF/X モードでは、線は完全に注釈の四角形の内部に位置している必要があります）線の始点と終点を指定した 4 個の座標のリスト。
showcaption	（論理値。PDF 1.6）true にすると、contents または createrichtext オプションで指定した内容が、線の体裁の中のキャプションとして複製されます。デフォルト：false

1. 座標は、デフォルト座標（usercoordinates オプションが false の場合）かユーザー座標（それが true の場合）で解釈されます。

12.3 フォームフィールド

C++ Java C# ***void create_field(double llx, double lly, double urx, double ury, String name, String type, String optlist)***

Perl PHP ***create_field(float llx, float lly, float urx, float ury, string name, string type, string optlist)***

C ***void PDF_create_field(PDF *p, double llx, double lly, double urx, double ury, const char *name, int len, const char *type, const char *optlist)***

新規フォームフィールドを作成するか、取り込まれたフォームフィールドに流し込みを行います。

llx・lly・urx・ury フィールドの長方形の左下隅と右上隅の *x・y* 座標を、デフォルト座標で（*usercoordinates* オプションを *false* にしたとき）、またはユーザー座標で（*true* にしたとき）指定します。PDF フォームフィールドの辺は常にページの辺に並行になります。

フォームフィールドの座標は、*PDF_rect()* メソッドの引数とは違うことに注意してください。*PDF_create_field()* は 2 個の隅に対する座標を直接とるのに対し、*PDF_rect()* では、1 個の隅の座標に幅と高さの値をあわせて指定します。

name （ハイパーテキスト文字列）フォームフィールドの名前。場合によっては、*PDF_create_fieldgroup()* で作成しておいた 1 個ないし複数のグループの名前を頭につけます。グループ名どうしの間と、グループ名とフィールド名との間は、ピリオドキャラクター「.」で区切る必要があります。フィールド名は、文書内で一意にする必要があり、かつ、ピリオドキャラクター「.」で終わってはいけません。

len （C 言語バインディングのみ）*name* の長さ（バイト単位）。*len=0* にすると null 終了文字列を与える必要があります。

type フィールドの種別を表 12.5 に従って指定します。

表 12.5 フォームフィールド種別

種別	アイコン	この種別に関連するオプション（一般オプションに加えて）
checkbox		buttonstyle・currentvalue・itemname
combobox		commitonselect・charspacing・currentvalue・editable・itemnamelist・itemtextlist・sorted・spellcheck
listbox		charspacing・commitonselect・currentvalue・itemnamelist・itemtextlist・multiselect・sorted・topindex
pushbutton		buttonlayout・caption・captiondown・captionrollover・charspacing・fitmethod・icon・icondown・iconrollover・position・submitname
radiobutton		buttonstyle・currentvalue・itemname・toggle・unisonselect fieldtype=radiobutton を持つグループを必要とします。
signature		fieldcontent・lockmode

表 12.5 フォームフィールド種別

種別	アイコン	この種別に関連するオプション（一般オプションに加えて）
textfield		comb・charspacing・currentvalue・fileselect・maxchar・multiline・password・richtext・scrollable・spellcheck
		（廃止。PDF 2.0 では利用できません）テキストフィールドはバーコードにも使われます：barcode

optlist フィールドのプロパティ群を指定したオプションリスト：

- 表 12.6 に従った、フィールドプロパティ群のためのオプション。以下のオプションがすべてのフィールド種別で使えます：
  action・*alignment*・*backgroundcolor*・*barcode*・*bordercolor*・*borderstyle*・*calcorder*・*dasharray*・*defaultvalue*・*display*・*exportable*・*fieldtype*・*fillcolor*・*font*・*fontsize*・*highlight*・*layer*・*linewidth*・*locked*・*orientate*・*readonly*・*required*・*strokecolor*・*taborder*・*tooltip*・*usercoordinates*
- 特定のフィールド種別群で使えるオプション群を表 12.5 に挙げます。これらについては表 12.6 でも詳述しています。
- （*PDF_create_fieldgroup()* の場合とフォーム流し込みモードの場合には不可）表 14.4 に従った、短縮構造エレメントタグ付けのためのオプション：*tag*
- 一般オプション：*hypertextencoding*・*hypertextformat*（表 2.1 参照）

詳細 ページ上のフィールドのタブ順序（タブキーが押された時にフォーカスを得る順序）は、デフォルトでは *PDF_create_field()* を呼び出した順序で決まりますが、*taborder* オプションを使ってそれ以外の順序を指定することもできます。タブ順序は、フィールドを作成した後に変えることはできません。ただしこの動作は、*PDF_begin/end_page_ext()* の *taborder* オプションでオーバーライドすることもできます。

Acrobat では、テキストフィールドにフォーマット（数値・パーセントなど）を割り当てることも可能です。ただしこれは PDF レファレンスには記載されておらず、カスタム JavaScript によって実装されています。これと同じ効果を得るには、フィールドに、Acrobat 内の定義済みの JavaScript 関数を参照する JavaScript アクションを関連づければよいでしょう。

タグ付き PDF モードでは、このメソッドは、作成されるフォームフィールドに対して然るべき *OBJR* エレメントを自動的に作成します。ユーザーは、このメソッドを呼び出す前に、対応する *Form* コンテナエレメント（PDFlib チュートリアル参照）を作成する必要があります。

フォームフィールドで使用するフォントは、Acrobat における制約を回避するために、注意して選択する必要があります。推奨事項：

- そのフォントは、閲覧するマシンにインストールされているべきです。
- TrueType・OpenType フォントは、オプション *nosubsetting* を用いて読み込まれているべきです。記号フォントにはオプション *encoding=builtin* を使用できます。

取り込まれたフォームフィールドに流し込み このメソッドを使うと、フォームフィールドを作成すること以外に、*PDF_fit_pdi_page()* で *usefields* オプションを用いて取り込んだフォームフィールドの体裁と内容を変更することもできます。フォーム流し込みモードは以下のように動作します：

- *name* 引数の内容は、取り込まれたフィールドの完全修飾名にする必要があります。存在しないフォール度の名前を与えるとエラーになります。取り込まれた文書の中のフィールド名は、pCOS疑似オブジェクト *pages[...]/fields[...]/fullname* を用いてクエリーすることもできます。
- *type* 引数は無視されます：取り込まれた文書の中のフィールド種別は、pCOS 疑似オブジェクト *pages[...]/fields[...]/type* を用いてクエリーすることもできます。
- フィールド座標 *llx/lly/urx/ury* は無視されます。
- 各フィールド種別で使用可能なオプションはたいてい与えることができます。すべてのオプションのデフォルトは「元のフィールドプロパティそのまま」に変わります。最も重要なこととして、フィールドの内容はオプション *currentvalue* を用いて変更できます。
- 許されないオプションもあります（表 12.6 参照）。

PDF/A 制約がかかるオプションがあります。詳しくは表 12.6 を参照してください。オプション *backgroundcolor*・*bordercolor*・*fillcolor*・*strokecolor* は以下の条件に縛られます：*RGB* カラーはつねに許され、*Grayscale* カラーは出力インテント（種類はどれでも）とともにのみ許され、*CMYK* カラーは CMYK 出力インテントとともにのみ許されます。

PDF/UA *PDF_begin_item()* を用いて、またはこのメソッドを呼び出す際に *tag* オプションを用いて、種別 *Form* の構造エレメントを作成する必要があります。*tooltip* オプションが必須です。

PDF/X フォームフィールドは、完全に BleedBox（BleedBox がないときは TrimBox/ArtBox）の外に置く場合のみ許されます。

スコープ **ページ**。*target* オプションを用いて作成した体裁アクションを与えた場合、このメソッドは、ターゲット注釈が位置しているページにおいてのみ呼び出せます。

フォーム流し込みモードにおいては、すなわち、フォームフィールドが *PDF_fit_pdi_page()* の *usefields* オプションを用いて他の文書から取り込まれている場合には、このメソッドを使って新規フィールドを作成することはできず、取り込まれたフィールドに流し込みを行うことしかできません。このメソッドは、取り込まれたフィールドを内容としているページを配置するために *PDF_fit_pdi_page()* を呼び出したページにおいてのみ呼び出せます。

C++ Java C# ***void create_fieldgroup(String name, String optlist)***

Perl PHP ***create_fieldgroup(string name, string optlist)***

C ***void PDF_create_fieldgroup(PDF *p, const char *name, int len, const char *optlist)***

フォームフィールドグループを、さまざまなオプションに従って作成します。

name （ハイパーテキスト文字列）フォームフィールドグループの名前。さらに別のグループの名前を頭につけることもできます。フィールドグループは任意の深さにネストさせることが可能です。グループ名どうしはピリオドキャラクター「.」で区切る必要があります。グループ名は、文書内で一意にする必要があり、また、ピリオドキャラクター「.」で終わってはいけません。

len （C 言語バインディングのみ）*name* の長さ（バイト単位）。*len=0* にすると null 終了文字列を与える必要があります。

optlist *PDF_create_field()* のフィールドオプション群を持ったオプションリスト。

詳細 フィールドグループは、*fieldtype* オプションに応じて、以下の目的に役立ちます：

表 12.6 PDF_create_field( )・PDF_create_fieldgroup( ) によるフィールドプロパティオプション

オプション	説明
action	（アクションリスト。PDF/A では不可）以下のイベントのいずれか 1 個ないし複数に対するフィールドアクションのリスト。activate イベントはすべてのフィールド種別で使えますが、それ以外のイベントは type=pushbutton・checkbox・radiobutton では使えません。デフォルト：空リスト ***activate*** フィールドがアクティブにされた時に実行させたいアクション。 ***blur*** フィールドが入力フォーカスを失った時に実行させたいアクション。 ***calculate*** 他のフィールドの値が変わった時にこのフィールドの値を再計算するために実行させたい JavaScript アクション。 ***close*** （PDF 1.5）フィールドを含むページが閉じられた時に実行させたいアクション。 ***down*** マウスボタンがフィールドの領域内で押された時に実行させたいアクション。 ***enter*** マウスがフィールドの範囲内に入った時に実行させたいアクション。 ***exit*** マウスがフィールドの範囲外に出た時に実行させたいアクション。 ***focus*** フィールドが入力フォーカスを得た時に実行させたいアクション。 ***format*** フィールドがその時点の値を表示するためにフォーマットされる前に実行させたい JavaScript アクション。これを使うと、フィールドの値をフォーマットされる前に変更することができます。 ***invisible*** （PDF 1.5）フィールドを含むページがもう見えなくなった時に実行させたいアクション。 ***keystroke*** ユーザーがテキストフィールドかコンボボックスにキー入力した時か、またはスクロール可能なリストボックスの選択を変更した時に実行させたい JavaScript アクション。 ***open*** （PDF 1.5）フィールドを含むページが開かれた時に実行させたいアクション。 ***up*** マウスボタンがフィールドの領域内で放された時に実行させたいアクション（フィールドをアクティブにするにはこれが通常使われます）。 ***validate*** フィールドの値が変更された時に実行させたい JavaScript アクション。これを使うと、新しい値の有効性を検証することができます。 ***visible*** （PDF 1.5）フィールドを含むページが見えた時に実行させたいアクション。
alignment	（キーワード）フィールド内のテキストの整列：left・center・right。デフォルト：left
background-color bordercolor	（色。PDF/A の色の制約が課されます）フィールドの背景か枠の色。使える色空間：none・gray・rgb・cmyk。デフォルト：none
barcode	（オプションリスト。type=textfield でのみ可。readonly を暗示。PDF 1.7ext3。廃止。PDF 2.0 では利用できません）表 12.7 のオプション群に従ってバーコードフィールドを作成します。このフィールドは、他のフィールド群の内容に基づいてバーコード内容を決定するか固定値を与える calculate イベントスクリプトを持つ action オプションを持っている必要があります：action={calculate=...}。 バーコードフィールドは、フルバージョンの Acrobat でのみ動作し、Acrobat Reader やサードパーティーの PDF ビューアーでは動作しません。
borderstyle	（キーワード）フィールドの枠のスタイル：solid・beveled・dashed・inset・underline。デフォルト：solid
button-layout	（キーワード。type=pushbutton でのみ可）ボタンのラベルの、ボタンのアイコンに対する位置を、両方とも指定しているときに限り指定します：below・above・right・left・overlaid。デフォルト：right
buttonstyle	（キーワード。type=radiobutton・checkbox でのみ可）フィールドで使いたい記号：check・cross・diamond・circle・star・square。デフォルト：check

表 12.6 PDF_create_field( ) ・ PDF_create_fieldgroup( ) によるフィールドプロパティオプション

オプション	説明
calcorder	（整数。フィールドが calculate イベントに対する JavaScript アクションを持つときのみ）フィールドの、他のフィールド群に対する計算順序。小さい数を持つフィールドは、大きい数を持つフィールドより先に計算されます。デフォルト：10 ＋カレントページで使われている最大の calcorder（はじめは 10）
caption	（内容文字列。type=pushbutton でのみ可。押しボタンでは、caption と icon オプションのどちらか一方を与える必要があります）ボタンが入力フォーカスを持たない時に表示させたいラベルテキスト。空文字列（すなわち caption {}）は、キャプションもアイコンも生み出しません。デフォルト：なし
caption-down	（内容文字列。type=pushbutton でのみ可。要 highlight=push。PDF/A では不可）ボタンがアクティブにされた時に表示させたいラベルテキスト。デフォルト：なし
caption-rollover	（内容文字列、type=pushbutton でのみ可。要 highlight=push。PDF/A では不可）ボタンが入力フォーカスを持つ時に表示させたいラベルテキスト。デフォルト：なし
charspacing	（float。type=radiobutton・checkbox・signature では不可）フィールド内のテキストの字間を、カレントユーザー座標系の単位で指定します。デフォルト：0
comb	（論理値。type=textfield でのみ可。PDF 1.5）true にすると、multiline・fileselect・password オプションを false にしている場合には、maxchar オプションに整数値を与えていれば、フィールドは、キャラクターごとに等間隔に、多数のサブフィールドに分割されます（maxchar オプションに従って）。デフォルト：false
commit-onselect	（論理値。type=listbox・combobox でのみ可。PDF 1.5）true にすると、リストの中で選択された項目は、選択後ただちに確定されます。false にすると、項目はフィールドから出た時に確定されます。デフォルト：false
currentvalue	（type=pushbutton・signature では不可）フィールドの初期値。型とデフォルトは、フィールドの種別によって異なります： ***checkbox・radiobutton***（文字列）文字列 Off にすると、ボタンは非アクティブになります。Off 以外の任意の文字列にすると、ボタンはアクティブになります。このオプションは、最初のボタンに対して設定する必要があります。デフォルト：Off ***combobox***（内容文字列）コンボボックス内で選択されている値。itemtextlist オプションで与えた文字列群のうちのひとつである必要があります。デフォルト：空 ***listbox***（整数のリスト）itemtextlist（sorted オプションを与えている場合、ソート前の）の中で選択されている項目のゼロベースの番号。デフォルト：なし ***textfield***（内容文字列）テキストフィールドの内容。デフォルト：空
dasharray	（非負 float 2 個のリスト。borderstyle=dashed でのみ可）破線枠の短線と間隙の長さを、デフォルト単位で指定します（表 7.1 参照）。デフォルト：3 3
defaultvalue	ResetForm アクション後のフィールドの値。型とデフォルトは、currentvalue オプションと同じです。例外：リストボックスでは、1 個の整数値しか指定できません。
display	（キーワード。PDF/A では強制的に visible になります）画面と紙の上での表示・非表示：visible・hidden・noview・noprint。デフォルト：visible
editable	（論理値。type=combobox でのみ可）true にすると、枠の中で選択中のテキストを編集できます。デフォルト：false
exportable	（論理値）フィールドは、SubmitForm アクションが起きた時に書き出されます。デフォルト：true

表 12.6 PDF_create_field( ) ・ PDF_create_fieldgroup( ) によるフィールドプロパティオプション

オプション	説明
fieldcontent	（テンプレートハンドル：type=signature でのみ可）与えたテンプレートがこの署名フィールドの中に表示されます。署名のやり方の指示を書いたりできます。テンプレートの幅と高さがフィールドと同じになっている必要があります。
fieldtype	（キーワード。PDF_create_fieldgroup( ) でのみ可）このグループの中のフィールド群の種別（デフォルト：mixed）： ***radiobutton*** このグループにはラジオボタン群だけが入ります。オプション unisonselect を指定する必要があります。itemtextlist・itemnamelist・currentvalue・defaultvalue オプションは、PDF_create_fieldgroup( ) で指定する必要があり、PDF_create_field( ) で指定してはいけません。 ***mixed*** このグループの中には任意の種類のフィールド群を混在させることができます。 ***pushbutton・checkbox・listbox・combobox・textfield*** このグループには、指定した種類のフィールド群だけが入ります。そのカレント値が、すべてのグループフィールドに同時に表示されます。このフィールド群が別々のページにあっても同様です。currentvalue など共通のオプション群は、PDF_create_fieldgroup( ) で指定する必要があり、PDF_create_field( ) で指定してはいけません。
fileselect	（論理値。type=textfield でのみ可）true にすると、フィールドの中のテキストは、ファイル名として扱われます。デフォルト：false
fillcolor	（色。PDF/A の色の制約が課されます）テキストの塗り色。使える色空間：gray・rgb・cmyk。デフォルト：PDF/A で出力インテントがない場合には {rgb 0 0 0}、それ以外なら {gray 0}（すなわち黒）
fitmethod	（キーワード。type=pushbutton でのみ可）icon・icondown・iconrollover オプションで与えているテンプレートを、ボタンの中に配置させたい方式。とりうるキーワード（デフォルト：meet）： ***auto*** テンプレートがボタンに収まりきるときは meet と同じ、そうでなければ clip ***nofit*** clip と同じ ***clip*** テンプレートを拡縮せずに、フィールドの端で切り落とします ***meet*** テンプレートを、縦横比を保ちつつ、ボタンに収まるよう拡縮します ***slice*** meet と同じ ***entire*** テンプレートを、ボタン全体を覆うように拡縮します
font	（フォントハンドル。type=listbox・combobox・textfield の場合と、type=pushbutton で caption・captionrollover・captiondown のいずれか 1 つないし複数を指定している場合には必須。type=checkbox・radiobutton・signature の場合には不可）フィールドで使いたいフォント。フォントとエンコーディングに関する推奨事項について、上述の「詳細」を参照してください。Acrobat はキャラクターを、それがそのフォントのエンコーディングに含まれていなくても表示することができます。たとえば、encoding=winansi を使っておきながら winansi 外の Unicode キャラクターを与えることが可能です。
fontsize	（文字サイズ。type=signature では不可）文字サイズを、ユーザー座標で指定します。値 0 かキーワード auto は、文字サイズが長方形に合わせて調整されることを意味します。デフォルト：auto
highlight	（キーワード）ユーザーがフィールドをクリックした時の、そのハイライトモード：none・invert・outline・push。デフォルト：invert
icon	（テンプレートハンドル[1]。type=pushbutton でのみ可。押しボタンでは、caption と icon オプションのどちらか一方を与える必要があります）ボタンが入力フォーカスを持たない時に表示されたいテンプレートのハンドル。デフォルト：caption・font オプションから体裁が生成されます

表 12.6 PDF_create_field( ) ・ PDF_create_fieldgroup( ) によるフィールドプロパティオプション

オプション	説明
icondown	（テンプレートハンドル[1]。type=pushbutton でのみ可。要 highlight=push。PDF/A では不可）ボタンがアクティブにされた時に表示させたいテンプレートのハンドル。デフォルト：captiondown・font オプションから体裁が生成されます
iconrollover	（テンプレートハンドル[1]。type=pushbutton でのみ可。要 highlight=push。PDF/A では不可）ボタンが入力フォーカスを持つ時に表示させたいテンプレートのハンドル。デフォルト：captionrollover・font オプションから体裁が生成されます
itemname	（ハイパーテキスト文字列。type=radiobutton・checkbox でのみ可。書き出し値が Latin 1 文字列でないときには用いる必要があります）フィールドの書き出し値。グループ内の複数のラジオボタンの項目名は同一にすることができます。デフォルト：フィールド名
item-namelist	（ハイパーテキスト文字列のリスト。type=listbox・combobox でのみ可）リスト項目群の書き出し値。複数の項目が同じ書き出し値を持つことができます。デフォルト：なし
itemtextlist	（内容文字列のリスト。type=listbox・combobox でのみ可、かつその場合には必須。itemnamelist を与えている場合にはフォーム流し込みモードにおいても必須）リストのすべての項目のテキスト内容。itemnamelist と itemtextlist を両方指定するときは、両方の文字列の数を同じにする必要があります。
layer	（レイヤーハンドル。PDF 1.5）フィールドを属させたいレイヤー。フィールドが可視になるのは、そのレイヤーが可視のときだけになります。
linewidth	（float）フィールドの枠の線幅を、デフォルト座標で指定します。デフォルト：1
locked	（論理値）true にすると、フィールドのプロパティを Acrobat で編集できなくなります。デフォルト：false
lockmode	（キーワード。type=signature でのみ可。PDF 1.5）フィールドが署名された時にロックさせたいフィールド群を示します： ***all*** 文書のすべてのフィールドがロックされます。
maxchar	（整数またはキーワード。type=textfield でのみ可）フィールドの中のテキストのキャラクター数の上限か、または制限なしにしたいときはキーワード unlimited。デフォルト：unlimited
multiline	（論理値。type=textfield でのみ可）true にすると、テキストは必要に応じて折り返されて複数行になります。デフォルト：false
multiselect	（論理値。type=listbox でのみ可）true にすると、リストで複数の項目が選択できるようになります。デフォルト：false
orientate	（キーワード）フィールドの中での内容の向き：north・west・south・east。デフォルト：north
password	（論理値。type=textfield でのみ可）true にすると、テキストが入力時に見えなくなります。パスワードフィールドに対して currentvalue を与えるべきではありません。デフォルト：false

表 12.6 PDF_create_field( ) ・ PDF_create_fieldgroup( ) によるフィールドプロパティオプション

オプション	説明
position	(float かキーワードのリスト。type=pushbutton でのみ可) icon/icondown/iconrollover オプション群で与えているテンプレートの、フィールド長方形内における相対位置を指定する 1 個か 2 個の値。{0 0} はフィールドの左下隅で、{100 100} は右上隅です。値は、フィールド長方形の幅と高さに対するパーセント値で指定します。両方のパーセント値が等しいときは、1 個の float 値を指定するだけで充分です。 キーワード left・center・right (x 方向で)、または bottom・center・top (y 方向で) を、値 0・50・100 と同じ意味で使うこともできます。1 個のキーワードだけを指定したときは、他方の方向についてはそれに対応するキーワードが追加されます。デフォルト : {center}。例 : {0 50} または {left center}　テンプレートを左揃え {50 50} または {center}　テンプレートを中央揃え {100 50} または {right center}テンプレートを右揃え
readonly[2]	(論理値) true にすると、フィールドに何も入力できなくなります。デフォルト ; false
required	(論理値) true にすると、フィールドは、フォームが送信される時に値を持つ必要があります。デフォルト : false
richtext	(論理値。type=textfield でのみ可。PDF 1.5) リッチテキスト組版を可能にします。true にするときは、fontsize は 0 にしてはならず、fillcolor は色空間 cmyk を用いてはいけません。デフォルト : false
scrollable	(論理値。type=textfield でのみ可) true にすると、テキストがフィールドに収まりきらないときは、テキストはフィールドの外の見えない領域へ移行します。false にすると、テキストがフィールドを満たしたときは、もう入力を受け付けなくなります。デフォルト : true
sorted	(論理値。type=listbox・combobox でのみ可) true にすると、リストの内容がソートされます。デフォルト : false
spellcheck	(論理値。type=textfield・combobox でのみ可) true にすると、フィールドの中でスペルチェック機能がアクティブになります。デフォルト : type=textfield かつ password=true なら false、そうでないなら true
strokecolor	(色。PDF/A の色の制約が課されます) テキストの描線色。使える色空間 : gray・rgb・cmyk。デフォルト : PDF/A で出力インテントがない場合には {rgb 0 0 0}、それ以外なら {gray 0} (すなわち黒)。
submitname	(ハイパーテキスト文字列。type=pushbutton でのみ推奨) フィールドデータを書き出す際に用いたいマッピング名。
taborder	(整数。フォーム流し込みモードにおいては許されません) フィールドの、他のフィールド群に対するタブ順序。小さい数を持つフィールドは、大きい数を持つフィールドより先にフォーカスされます。デフォルト : 10 +カレントページで使われている最大の taborder (ページの最初のフィールドでは 10)。このデフォルトでは結果として、作成順によってタブ順序が決まります。
toggle	(論理値。PDF_create_fieldgroup( ) で type=radiobutton の場合にのみ可) true にすると、グループのラジオボタンをクリックしてアクティブにすることも、非アクティブにすることもできます。false にすると、クリックしてアクティブにすることしかできず、非アクティブにするにはほかのボタンをクリックする必要があります。デフォルト : false
tooltip[2]	(ハイパーテキスト文字列。PDF/UA-1 では空でない文字列が必須) フィールドのツールヒントに表示させたいテキスト。スクリーンリーダによっても使用されます。ラジオボタンとグループでは、Acrobat はグループの最初のボタンのツールヒントを使って、他は無視されます。デフォルト : なし
topindex	(整数。type=listbox でのみ可) 先頭に表示させたい項目の番号。最初の項目の番号は 0 です。デフォルト : 0

表 12.6 PDF_create_field( ) ・ PDF_create_fieldgroup( ) によるフィールドプロパティオプション

オプション	説明
unisonselect	（論理値。PDF_create_fieldgroup( ) で type=radiobutton かつ PDF 1.5 の場合にのみ可）true にすると、同じフィールド名か項目名を持つラジオボタン群が、同時に選択されます。デフォルト：false
user-coordinates	（論理値。フォーム流し込みモードにおいては許されません）false にすると、フィールドの座標はデフォルト座標系で表されていると見なされます。そうでなければ、カレントユーザー座標系が使われます。デフォルト：usercoordinates グローバルオプションの値

1. アイコンのテンプレートは、PDF_begin_template( ) メソッドで作成することができます。アイコンが画像だけでできているときは、PDF_load_image( ) に template オプションを与えてテンプレートを作成することもできます。
2. type=radiobutton にしているときは、このオプションは PDF_create_field( ) では使えず、PDF_create_fieldgroup( ) でのみ使えます。

- *fieldtype=radiobutton* を持つグループは、ラジオボタン群のためのコンテナとして必須です。それ以外のすべての種類のフィールドについては、グループのメンバーとすることは必須ではありません。
- 任意のフィールド群を、その種類が同じであっても異なっていても、*fieldtype=mixed* を持つ論理グループの中にまとめることができます。このグループの中のすべてのフィールドの名前は、頭にグループの名前が付きます。例：*name.prefix*・*name.first*・*name.initial*・*name.last*。
- その他すべての *fieldtype* の値を使って、同期フィールドを作成できます。すなわち、同一種別の 1 個ないし複数のグループフィールドに同一の値を表示させることが可能です。

PDF_create_field( ) でフィールド名を作成する際に頭にフィールドグループ（*button* 等）の名前を付けると（*button.1* 等）、その新しいフィールドはそのグループの一部になります。グループに対する *optlist* で与えたフィールドオプション群は、そのグループの中のすべてのフィールドへ継承されます。

PDF/A *PDF_create_field( )* を参照してください。

PDF/UA *PDF_create_field( )* を参照してください。

スコープ **オブジェクト**以外任意

フォームフィールド群が *PDF_fit_pdi_page( )* の *usefields* オプションを用いて他の文書から取り込まれている場合にはこのメソッドは使えません。

表 12.7 PDF_create_field( )・PDF_create_fieldgroup( ) の barcode オプションのサブオプション

オプション	説明
caption	（ハイパーテキスト文字列）バーコードの下に表示されるキャプション。デフォルトでは、Acrobat はその文書の file: URL をキャプションとして作成します。
dataprep	（整数）データの生成方式。使える値（デフォルト：0）： ***0*** バーコード内のデータをエンコードする前に何の圧縮も適用しません。 ***1*** データをエンコードする前にデータを Flate 圧縮アルゴリズムで圧縮します。
ecc	（整数。symbology=PDF417・QRCode の場合には必須）誤り訂正係数。値が大きいほど、冗長性を通じた、より良い誤り訂正が生成されますが、より大きなバーコードが必要となります。symbology=PDF417 の場合、この値は範囲 0 ～ 8 内でなければなりません。symbology=QRCode の場合、この値は範囲 0 ～ 3 内でなければなりません。
resolution	（正の整数）バーコードが表示される解像度を dpi 単位で（デフォルト：300）
symbology	（キーワード。必須）使いたいバーコード技術： ***PDF417*** ISO 15438 に従った PDF417 バーコード ***QRCode*** ISO 18004 に従った QR コード 2005 バーコード ***DataMatrix*** ISO 16022 に従ったデータマトリックスバーコード
xsymheight	（整数。symbology=PDF417 の場合にのみ可。かつその場合は必須）2 個のバーコードモジュール間の縦間隔をピクセル単位で。比 xsymheight/xsymwidth が整数値である必要があります。この比に許される範囲は 1 ～ 4 です。
xsymwidth	（整数。必須）2 個のバーコードモジュール間の横間隔をピクセル単位で

12.4 アクション

C++ Java C# ***int create_action(String type, String optlist)***

Perl PHP ***int create_action(string type, string optlist)***

C ***int PDF_create_action(PDF *p, const char *type, const char *optlist)***

アクションを作成します。アクションは、さまざまなオブジェクトやイベントに適用することができます。

type 表 12.8 に従ったアクション種別。

表 12.8 アクション種別

種別	説明：この種別に関連するオプション（一般オプションに加えて）
GoTo	カレント文書の中の移動先へ行きます：destination・destname
GoTo3DView	（PDF 1.6）3D モデルのカレントビューを設定します：3Dview・target
GoToE	（PDF 1.6）埋め込まれた文書の中の移動先へ行きます：targetpath
GoToR	別の（リモート）文書の中の移動先へ行きます：destination・destname・filename・newwindow・remotestructdest
Hide	フォームフィールドを隠すか表示させます：hide・namelist
ImportData	フォームフィールド群の値をファイルから取り込みます。
JavaScript	JavaScript コードによるスクリプトを実行します：javascript・script・scriptname
Launch	アプリケーションまたは文書を起動します：filename・newwindow
Named	Acrobat のメニュー項目を、その名前で同定して実行します：menuname
Rendition	（PDF 1.5）Screen 注釈内におけるマルチメディア内容の再生を制御します：javascript・operation・rendition・script・target
ResetForm	文書内のフィールドをいくつか、ないしすべて、デフォルト値に設定します。
RichMedia-Execute	（PDF 1.7ext3）RichMedia 注釈へコマンドを送ります：functionname・instance・richmediaargs・target
SetOCGState	（PDF 1.5）レイヤーを隠すか、または表示させます：layerstate・preserveradio
SubmitForm	データを URL（uniform resource locator）へ、すなわちインターネットのアドレスへ送信します：canonicaldate・exclude・exportmethod・submitemptyfields・url
Trans	（PDF 1.5）表示を何らかの視覚効果を使って更新します。これは、連続する複数のアクションの最中に表示を制御するために有用でしょう：duration・transition
URI	URI（uniform resource identifier）を解決します。すなわち、インターネットのアドレスへ飛びます：ismap・url

optlist アクションの特性群を指定したオプションリスト：

- 一般オプション：***errorpolicy***（表 1.5 参照）
- 表 12.9 に従った、以下の種別独自オプション：
  3Dview・*richmediaargs*・*canonicaldate*・*destination*・*destname*・*duration*・*exclude*・*exportmethod*・*filename*・*functionname*・*hide*・*instance*・*ismap*・*javascript*・*layerstate*・

menuname・*namelist*・*newwindow*・*operation*・*preserveradio*・*rendition*・*remotestructdest*・*script*・*scriptname*・*submitemptyfields*・*target*・*targetpath*・*transition*・*url*

- C の場合と、Perl・PHP・Ruby で *stringformat=legacy* の場合のためのエンコーディングオプション：*hypertextencoding*（表 2.1 参照）

戻り値 アクションハンドル。文書の中のオブジェクトにアクションを関連づけるのに使えます。アクションハンドルは、カレントの**文書**スコープを終えるまで使えます。

詳細 このメソッドは、ただ 1 つのアクションを作成します。さまざまなオブジェクトには（ページ、フォームフィールドのイベント、しおり等）、複数のアクションも与えることができますが、アクションは 1 つずつ、個々に *PDF_create_action( )* を呼び出して作成する必要があります。1 つのアクションを、複数のオブジェクトに使うことも可能です。同じオプション群を持ったアクションをそれまでにすでに作成している場合は、既存のハンドルを再利用することを推奨します。

PDF/A 以下のアクション種別のみが許容されます：
GoTo・*GoToE*・*GoToR*・*Named*・*SubmitForm*・*URI*

PDF/UA *ismap=true* オプションは許容されません。

PDF/X このメソッドを呼び出してはいけません。

スコープ **オブジェクト**以外任意。もし *type=Rendition* かつ *target* オプションを与えている場合には、このメソッドは**ページ**スコープ内でのみ許され、かつ、ターゲット注釈が位置しているページにおいて呼び出す必要があります。返されたハンドルは、次に *PDF_end_document( )* を呼び出すまで使えます。

表 12.9 PDF_create_action( ) のオプション

オプション	説明
3Dview	（キーワードか 3D ビューハンドル。GoTo3DView。必須）3D 注釈のビューを選びます。キーワード first・last・next・previous（注釈の views オプションの中の各項目を参照する）・default（注釈の defaultview オプションを参照する）のいずれか、または PDF_create_3dview( ) で作成した 3D ビューハンドル。
canonical-date	（論理値。SubmitForm）true にすると、日付を表すフィールドの送信される値は、すべて標準形式に変換されます。フィールドを日付として解釈することは、フィールド自体の中ではなく、それを処理する JavaScript コードの中でだけ明示的に指定することができます。デフォルト：false
destination	（オプションリスト。GoTo・GoToE・GoToR。destname を与えていない場合には必須）飛ばしたい移動先を表 12.11 に従って指定したオプションリスト。structdest サブオプションが許されるのは type=GoTo の場合だけです。
destname	（ハイパーテキスト文字列。destination を与えていない場合には必須）GoTo：PDF_add_nameddest( ) で定義しておいた移動先の名前。この移動先は、それを参照する前に作ることも、後に作ることもできます。名前付き構造移動先が許されるのは type=GoTo の場合だけです。 GoToR・GoToE：別の文書の、または埋め込まれている文書の中の移動先の名前。
duration	（float。Trans）カレントページの表示遷移効果の継続時間を秒単位で設定します．デフォルト：1

表 12.9 PDF_create_action( ) のオプション

オプション	説明
exclude	（論理値）SubmitForm：true にすると、namelist オプションは、どのフィールドを除外したいかを指定します。文書の中のフィールドは、namelist 配列に挙げてあるものと、exportable オプションを false にしてあるもの以外がすべて送信されます。false にすると、namelist オプションは、どのフィールドを送信に含めたいかを指定します。フィールドグループを指定しているときは、そのメンバーもすべて送信されます。デフォルト：false ResetForm：true にすると、namelist オプションは、どのフィールドを除外したいかを指定します。文書内のフィールドは、namelist 配列に挙げてあるもの以外がすべてリセットされます。false にすると、namelist オプションは、どのフィールドをリセットに含めたいかを指定します。フィールドグループを指定しているときは、そのメンバーもすべてリセットされます。デフォルト：false
export-method	（キーワードのリスト。SubmitForm）フィールドの名前と値の送信方法を制御します（デフォルト：fdf） ***html*・*fdf*・*xfdf*・*pdf*** それぞれ、HTML・FDF・XFDF・PDF 形式で ***annotfields***（fdf のみ）注釈とフィールドをすべて含めます。 ***coordinate***（html のみ）submitform アクションを引き起こしたマウスクリックの座標を、フォームデータに含めて送信します。座標の値は、フィールドの長方形の左上隅から測ったものです。 ***exclurl***（fdf のみ）送信する FDF から url 文字列を除外します。 ***getrequest***（html・pdf のみ）HTTP GET を使って送信。指定しないと HTTP POST ***onlyuser***（fdf・annotfields のみ）リモートサーバーによって決定されるカレントユーザー名に一致する名前の注釈だけを送信に含めます。 ***updates***（fdf のみ）その PDF 文書に入っている増分アップデートをすべて含めます オプションの組み合わせ例：exportmethod={fdf updates onlyuser}
filename	（ハイパーテキスト文字列）GoToR・Launch（必須）：アクションがトリガーされた時に開かれる、外部（PDF でもそれ以外でも）ファイルまたはアプリケーションの名前。UNC ファイル名は、\\server\volume と書く必要があります。 ImportData（必須）：フォームデータの入っている外部ファイルの名前。 GoToE：移動先のルート文書の名前を、移動元のルート文書から相対的に指定します。この項目を与えないときは、移動元と移動先は同じルート文書を共有します。
functionname	（ハイパーテキスト文字列。RichMediaExecute。必須）JavaScript メソッド名を指定した文字列（フルスクリプトではない）。
hide	（論理値。Hide）フィールドを隠すか（true）、それとも表示するか（false）を指定します。デフォルト：true
instance	（整数。RichMediaExecute）その RichMedia 注釈の、そのスクリプトを実行する対象としたい Flash または 3D インスタンスを指定する、PDF_create_annotation( ) の richmedia オプションの configuration サブオプションの instances サブオプション内のオプションリストの番号（1 から始まる）。
ismap	（論理値。URI。PDF/UA-1 では true は不可）true にすると、url が解決された時に、マウス位置の座標が移動先 URI に追加されます。デフォルト：false
javascript	（type=JavaScript を用いて読み込んでおいたアセットハンドル。JavaScript・Rendition：script と javascript のどちらか 1 つだけを与えることができます）アクションの JavaScript

表 12.9 PDF_create_action( ) のオプション

オプション	説明
layerstate	（オプションリスト。SetOCGState。必須）キーワードとレイヤーハンドルのペアのリスト： ***on*** レイヤーを表示します ***off*** レイヤーを隠します ***toggle*** レイヤーのステータスを反転させます。これを使うときは、preserveradio オプションを false に設定している必要があります。
menuname	（文字列。Named。必須）実行したいメニュー項目の名前。PDF/A では、nextpage・prevpage・firstpage・lastpage だけが許されます。それ以外の場合は、他の名前も受け付けます。他のメニュー項目の名前を見つけるためのコードがクックブックの interactive/acrobat_menu_items トピックにあります。
namelist	（文字列のリスト。Hide。必須）隠したいか、あるいは表示させたい注釈群かフィールド群の名前（グループ名を含む）。 （SubmitForm）exclude オプションの設定によって、送信に含めたいか、あるいは除外したいフォームフィールド群の名前（グループ名を含む）。デフォルト：exportable オプションを false にしてあるフィールド以外がすべて送信されます。 （ResetForm）exclude オプションの設定によって、リセットに含めたいか、あるいは除外したいフォームフィールド群の名前（グループ名を含む）。デフォルト：すべてのフィールドがリセットされます。
newwindow	（論理値。GoToE・GoToR）移動先の文書を新しいウインドウで開くかどうかを指定するフラグ。このフラグを false にすると、移動先の文書はカレント文書から同じウインドウの中で切り換わります。 Launch：この項目は、ファイルが PDF 文書でないときは無視されます。デフォルト：Acrobat は、カレントの環境設定に従って動作します。
operation	（キーワード。Rendition。javascript か script を与えていないかぎりは必須）表現に適用したい操作： ***play*** 表現を頭から再生します。他の表現が注釈に紐付いている場合にはそれは停止します。 ***stop*** 表現が再生中か一時停止中なら停止します。 ***pause*** 表現が再生中なら一時停止します。たいていのストリーミング形式に対して効力がありません。 ***resume*** 表現が一時停止中なら再開します。 ***playfrombeginning*** もし同じ表現か別の表現が注釈に紐付いていてかつ一時停止中ならそれを再開します。そうでないなら、与えられた表現を頭から再生します。表現が一時停止中ならこれは play よりも速いです。
preserve-radio	（論理値。SetOCGState）true にすると、レイヤー間のラジオボタンステータス関係を保持します。デフォルト：true
rendition	（表現アセットハンドルのリスト。Rendition。operation=play か playfrombeginning の場合には必須）JavaScript か操作を適用する対象としたい 1 個ないし複数の表現。複数の表現を与えると、PDF ビューアーは対応している最初のものを選びます。
remote-structdest	（文字列。タグ付き PDF かつ type=GoToR の場合にのみ可。PDF 2.0）移動先にさせたいリモート文書の中の構造エレメントの識別子。PDFlib で作成する文書においてはこの識別子は構造エレメントに id タグ付けオプションを用いて割り当てる必要があります（elementid ではありません。それは PDF に書き込まれません）。与える ID が必ず実際にターゲット文書の中に存在するようにするのはユーザー側の役割です。

表 12.9 PDF_create_action( ) のオプション

オプション	説明
richmediaargs	（オプション usage=richmediaargs を用いて作成した POCA コンテナハンドル。RichMediaExecute）そのコマンドに対する任意の数の引数を指定した配列コンテナに対するハンドル。有効な引数は、種別が文字列か整数か float か論理値のオブジェクトです。デフォルト：引数なし
script	（ハイパーテキスト文字列。JavaScript・Rendition。script と javascript のどちらかが必須であり、かつ両方を与えることはできません）アクションの JavaScript コードを内容とする文字列。キャラクター \ や { } などについて、オプションリスト文法に対するクオーティング（エスケーピング）の諸要請に従う必要があります。任意のコードを渡すには、10 ページ「括弧で囲われていない文字列」で説明しているオプションリスト文法が有用でしょう。代替オプション javascript を用いてコードを渡す際にはクオーティングをする必要はありません。
scriptname	（ハイパーテキスト文字列。JavaScript）指定すると、script または javascript オプションで与えている JavaScript は、文書レベルの指定名の JavaScript として挿入されます。文書内で同じ scriptname を複数回与えると、最初のスクリプトだけが使われます。文書レベルの JavaScript は、文書が読み込まれた後に実行されます。これは、フォームフィールドで使いたいスクリプトで有用でしょう。
submit-emptyfields	（論理値。SubmitForm）true にすると、namelist・exclude オプションで決まるすべてのフィールドが、値を持つかどうかにかかわらず、送信されます。値のないフィールドについては、フィールド名だけが送信されます。false にすると、値のないフィールドは送信されません。デフォルト：false
target	（文字列。GoTo3DView・Rendition・RichMediaExecute。必須。表現アクションの場合には、operation を与えていて、かつ、type=Screen を用いた PDF_create_annotation( ) 以外のメソッドでこのアクションを使うときにのみ必須）アクションを実行する対象としたい、PDF_create_annotation( ) の name オプションで指定された通りの、ターゲット注釈の名前。Screen 注釈に表現アクションを与えると、この注釈が暗黙的にその表現アクションのターゲットとして使われます。
targetpath	（オプションリスト。GoToE。filename を指定していない場合には必須）移動先文書のパス情報を指定した移動先オプションリスト（表 12.10 参照）。各移動先オプションリストは、移動先へのフルパスの中の 1 個の要素を指定し、追加の要素群を持ったネストされた移動先オプションリスト群を持つこともできます。
transition	（キーワード。Trans）表示遷移効果を設定します。キーワードの一覧は表 3.9 を参照。デフォルト：replace
url	（文字列。URI・SubmitForm。必須）リンク先を指定する（type=URI の場合）、または送信内容を処理させたい Web サーバー上のスクリプトのアドレスを指定する（type=SubmitForm の場合）URL（Uniform Resource Locator）を、7 ビット ASCII または EBCDIC（ただし ASCII 文字だけを含む）でエンコードしたもの。任意のリソース（Web でもローカルでも）を指し示すことができ、先頭にプロトコル識別子（http:// など）が必要です。URL 内で特別な意味を持つキャラクター（「%」等）については、RFC 3986 に従ってクォートする必要があります。

表 12.10 PDF_create_action( ) の targetpath オプションのサブオプション

オプション	説明
annotation	（ハイパーテキスト文字列。relation=child かつ移動先がファイル添付注釈に関連づけられている場合には必須）pagenumber か destname で指定したページ上の移動先のファイル添付注釈の名前。
destname	（ハイパーテキスト文字列。pagenumber を与えていて、かつ relation=child で、しかも移動先がファイル添付注釈に関連づけられている場合以外には必須。pagenumber を指定すると無視されます）移動先のファイル添付注釈を内容として持つカレント文書内のページに対する名前付き移動先。

表 12.10 PDF_create_action( ) の targetpath オプションのサブオプション

オプション	説明
name	（ハイパーテキスト文字列。relation=child かつ移動先が添付リスト内に置かれている場合には必須。それ以外の場合には指定してはいけません。annotation を指定すると無視されます）PDF_begin/end_document( ) の添付リスト内の移動先の名前。
pagenumber	（整数。destname を与えていて、かつ relation=child で、しかも移動先がファイル添付注釈に関連づけられている場合以外には必須。destname を指定すると無視されます）移動先のファイル添付注釈を内容として持つカレント文書内のページの番号。
relation	（キーワード。必須）カレント文書と移動先（中間移動先でも可）の関係。使えるキーワード： ***parent*** 移動先はカレント文書の親です。 ***child*** 移動先はカレント文書の子です。
targetpath	（オプションリスト）追加の移動先文書のパス情報を表 12.10 に従って指定した移動先オプションリスト。このオプションを指定しないと、カレント文書が、移動先を含む移動先ファイルとなります。

12.5 名前付き移動先

C++ Java C# ***void add_nameddest(String name, String optlist)***

Perl PHP ***add_nameddest(string name, string optlist)***

C ***void PDF_add_nameddest(PDF *p, const char *name, int len, const char *optlist)***

名前付き移動先を、文書のページ上に作成します。

name （ハイパーテキスト文字列）移動先の名前。リンク、しおり、その他のトリガーの対象として使えます。移動先名は、文書内で一意にする必要があります。文書内で同じ名前を複数回与えると、最初の定義だけが使われて、他は無警告で無視されます。

len （C 言語バインディングのみ）*name* の長さ（バイト単位）。*len=0* にすると null 終了文字列を与える必要があります。

optlist 移動先を指定したオプションリスト。空リストにすると、*{type=fitwindow page=0}* と同義になります。以下のオプションが使えます：

- 一般オプション：*errorpolicy*（表 1.5 参照）・
- 表 12.11 に従った移動先制御オプション：
  bottom*・*group*・*left*・*page*・*right*・**structdest**・*top*・*type*・*zoom
  これらのオプションにおける座標はつねにデフォルト座標系で解釈されます。ユーザー座標は使えません。なぜならターゲットページの座標系が未知であるからです。
- C の場合と、Perl・PHP・Ruby で *stringformat=legacy* の場合のためのエンコーディングオプション：*hypertextencoding*・*hypertextformat*（表 2.1 参照）

詳細 *optlist* で移動先の詳細を指定する必要があります。移動先は、カレント文書のどのページにあってもかまいません。与える *name* は、*PDF_create_action()*・*PDF_create_annotation()*・*PDF_create_bookmark()*・*PDF_begin/end_document()* の *destname* オプションで使えます。このやり方では、移動先の定義と利用とを、2 つの別々のステップに分けることができます。

あるいは、移動先が利用の時にわかる場合は、これらのメソッドで *destination* オプションを使って、名前付き移動先の定義と利用を一度に行うこともできますので、その場合は *PDF_add_nameddest()* は必要ありません。

スコープ **オブジェクト**以外任意

表 12.11 PDF_add_nameddest( ) の移動先オプション。PDF_create_action( )・PDF_create_annotation( )・PDF_create_bookmark( )・PDF_begin/end_document( ) の destination オプションでも使えます。

オプション	説明
bottom	（float。type=fitrect でのみ可）ウインドウの下端に合わせたい、ページの y 座標を、デフォルト座標系で表したもの。デフォルト：0
group	（文字列。page オプションを指定しているとき、文書がページグループを使っている場合には必須。そうでない場合には禁止）移動先のページが属するページグループの名前。
left	（float。type=fixed・fitheight・fitrect・fitvisibleheight でのみ可）ウインドウの左端に合わせたい、ページの x 座標を、デフォルト座標系で表したもの。デフォルト：0
page	（整数）移動先ページのページ番号（先頭ページは 1）。ページは、移動先 PDF に存在している必要があります。page 0 にすると、ページスコープの中ではカレントページを、文書スコープの中では page 1 を意味します。デフォルト：0

表 12.11 PDF_add_nameddest( ) の移動先オプション。PDF_create_action( )・PDF_create_annotation( )・PDF_create_bookmark( )・PDF_begin/end_document( ) の destination オプションでも使えます。

オプション	説明
right	(float。type=fitrect でのみ可) ウインドウの右端に合わせたい、ページの x 座標を、デフォルト座標系で表したもの。デフォルト：1000
structdest	(文字列。タグ付き PDF でのみ可。group・page オプションとともに与えてはいけません。PDF 2.0) 移動先は、文字列によって指定する構造エレメントから成ります。この文字列は、elementid タグ付けオプションを用いて移動先エレメントに割り当てる必要があります。この割り当てを行うのは、この structdest オプションを用いて参照するより前でも後でもかまいません。文書が終了するまでにこの ID をどの構造エレメントにも全く割り当てなかった場合には、この ID は警告なく構造ツリーのルートに割り当てられます。 この構造エレメントが決定するのは移動先のページ番号であり、ページ上の正確な位置や表示倍率については他のオプション群で指定できます。名前付き構造移動先を PDF_create_action( ) で使う場合には、そのアクション種別は GoTo である必要があります。
top	(float。type=fixed・fitwidth・fitrect・fitvisiblewidth でのみ可) ウインドウの上端に合わせたい、ページの y 座標を、デフォルト座標系で表したもの。デフォルト：1000
type	(キーワード) 対象ページ上でのウインドウの位置。とりうるキーワード (デフォルト：fitwindow)： ***fitheight*** ページの高さをウインドウに収めて、x 座標 left をウインドウの左端に合わせます。 ***fitrect*** left・bottom・right・top で指定している長方形をウインドウに収めます。 ***fitvisible*** ページの描画領域 (ArtBox) をウインドウに収めます。 ***fitvisibleheight*** ページの描画領域をウインドウに収めて、x 座標 left をウインドウの左端に合わせます。 ***fitvisiblewidth*** ページの描画領域をウインドウに収めて、y 座標 top をウインドウの上端に合わせます。 ***fitwidth*** ページの幅をウインドウに収めて、y 座標 top をウインドウの上端に合わせます。 ***fitwindow*** ページ全体をウインドウに収めます。 ***fixed*** left・top・zoom オプションで指定している、固定した移動先表示を使います。これらのいずれかを指定しないと、そのカレント値が保たれます。
zoom	(float またはパーセント値。type=fixed でのみ可) ページ内容を表示させたい倍率 (1 が 100% を意味します)。このオプションを指定しないか、または 0 にすると、リンクが押された時に適用されていた表示倍率が保たれます。

13 マルチメディア・3D・地理空間機能

この章の API メソッド：

- *PDF_load_asset()*
- *PDF_load_3ddata()*
- *PDF_create_3dview()*

13.1 マルチメディアアセットと添付

さまざまなマルチメディア機能を、以下の API メソッドとオプションを用いて実装します：

- マルチメディアについては、それが埋め込まれている場合も、外部ファイルまたは URL ベースの場合も、まず *PDF_load_asset()* で *type=Rendition* を用いてその映像か音声を読み込む必要があり、それから *PDF_create_action()* を用いて表現アクションを作成してから（265 ページ「12.4 アクション」を参照）、そのアクションを Screen 注釈に与える必要があります（245 ページ「12.2 注釈」を参照）。
- マルチメディアアセットを読み込むには、*PDF_load_asset()* で *type=3D/Sound/Video* を用います。このアセットを RichMedia 注釈に与えるには *richmedia* オプションを用います。このオプションのサブオプション群を表 13.5 で説明します。紐付けたいアクションを作成するには、*PDF_create_action()* で *type= RichMediaExecute* を用います。ただし音声と動画のために RichMedia 注釈を使うことは推奨しませんので、かわりに表現注釈を使ってください。
- JavaScript を読み込むには、*PDF_load_asset()* で *type=JavaScript* を用います。これは 3D 注釈・リッチメディア注釈・JavaScript アクションのために使います。
- *PDF_load_asset()* で *type=Attachment* を用いると、さまざまな API メソッドで使う添付または関連ファイルを読み込めます。

13.1.1 アセットを読み込む

C++ Java C# ***int load_asset(String type, String filename, String optlist)***

Perl PHP ***int load_asset(string type, string filename, string optlist)***

C ***int PDF_load_asset(PDF *p, const char *type, const char *filename, int len, const char *optlist)***

マルチメディアアセットか添付を、ファイルか URL から読み込みます。

type 読み込まれるアセットの種別を表 13.1 に従って指定したキーワード。

表 13.1 アセット種別

種別	アセットを使える API メソッドとオプション / サブオプション
3D	PDF_create_annotation( ) で type=RichMedia とオプション richmedia/configurations/instances/asset

表 13.1 アセット種別

種別	アセットを使える API メソッドとオプション / サブオプション
Attachment	▸ PDF_end_document( ) でオプション attachments ▸ PDF_create_annotation( ) で type=FileAttachment とオプション attachment ▸ PDF/A-3・PDF 2.0：PDF_end_document( )・PDF_begin/end_page_ext( )・PDF_create_annotation( )・PDF_begin_item( )・PDF_begin/end_dpart( ) でオプション associatedfiles と共通の XObject オプション associatedfiles ▸ PDF 2.0：PDF_begin/end_document( )・PDF_begin/end_page_ext( )・PDF_load_font( ) と共通の XObject オプション associatedfiles ▸ PDF 2.0：PDF_end_document( ) でオプション structureassociatedfiles
JavaScript	▸ PDF_create_annotation( ) で type=RichMedia とオプション richmedia/activate/scripts ▸ PDF_create_action( ) でオプション javascript ▸ PDF_load_3ddata( ) でオプション javascript
Rendition	▸ PDF_create_action( ) で type=rendition とオプション rendition
Sound	▸ PDF_create_annotation( ) で type=rendition とオプション richmedia/assets/asset

filename （名前文字列。*filenamehandling* グローバルオプションに従って解釈または参照されます。表 2.1 参照）PDF ファイル内に埋め込まれるディスクベースまたは仮想ファイルの名前。Unicode ファイル名が使えますが、ただし PDF 1.7 が必要です。*external=true* の場合にはこの引数にはファイル名でなく URL を与えます。

len （C 言語バインディングのみ）*filename* の長さ（バイト単位で）。*len* = 0 の場合には、ヌル終端文字列を与える必要があります。

optlist 以下のオプションを内容とすることができるオプションリスト：

- すべての種別に対する一般オプション：*errorpolicy*（表 1.5 参照）
- *type=Rendition* の場合には、表 13.2 に従って以下のオプションも可：
  autoplay・*alttext*・*baseurl*・*controller*・*duration*・*external*・*mimetype*・*monitor*・*name*・*nametree*・*opacity*・*permissions*・*style*・*url*・*volume*・*window*
- *type=3D/Attachment/JavaScript/Sound/Video* の場合には、表 13.4 に従って以下のオプションも可：
  description・*documentattachment*・*external*・*filename*・*mimetype*・*name*・*password*・*relationshipthumbnail*

戻り値 アセットハンドル。このハンドルは、それを囲う**文書**スコープの終了までの間、表 13.1 に挙げるメソッドとオプションで使用することができます。

errorpolicy=return の場合には、戻り値が -1 であるかどうかを呼び出し側でチェックする必要があります。なぜなら -1 はエラーを知らせているからです。この呼び出しが失敗した場合には、その失敗の理由を、*PDF_get_errmsg( )* を用いて要求することもできます。

詳細 *filename* の内容にかかる要件は、アセット種別によって異なります：

- *type=Rendition*（PDF 1.5）：*filename* は、メディアファイルを参照するか、リモートのメディアファイルの URL を内容とする必要があります。
- *type=3D*（PDF 1.7ext3）：U3D または PRC データ。この種別は、*RichMediaExecute* アクションで使うためだけのものです。3D 注釈と *GoTo3DView* アクションで使う 3D アセットを読み込むには *PDF_load_3ddata( )* を用いる必要があります。

- ***type=Attachment***：汎用の任意の内容、ただし PDF/A では制約あり
- ***type=JavaScript***：PDF 1.7 は ECMAScript 3 に対応しています。PDF 2.0 は ECMAScript 5.1 に対応しているほか、XML 拡張群と「Adobe Acrobat 3D JavaScript Reference」にも対応しています。両者とも、「Adobe JavaScript for Acrobat API Reference」に従った拡張群に対応しています。スクリプトのエンコーディングは、ISO 8859-1 か、BOM 付き UTF-16 LE または BE にする必要があります。
- ***type=Sound・Video***（PDF 1.7ext3）：MP3 形式の音声データか、Acrobat が対応している形式の動画データ。これらの種別を使えるのは ***RichMedia*** 種別と ***RichMediaExecute*** アクションでだけです。Screen 注釈と表現アクションで使う音声・動画アセットは、***type=Rendition*** を用いて読み込む必要があります。

PDF/A PDF/A-1：このメソッドを呼び出してはいけません。
PDF/A-2：***type=Attachment*** のみが許されます。***filename*** は、PDF/A-1 または PDF/A-2 文書を参照している必要があります。いくつかのオプションは制約されます。
PDF/A-3：***type=Attachment*** のみが許されます。いくつかのオプションは制約されます。***type=Attachment*** を用いて生成したアセットハンドルは、少なくとも 1 つの ***associatedfiles*** オプションへ与える必要があります。

PDF/X PDF/X-3/4：***type=Attachment*** のみが許されます。

スコープ **オブジェクト**以外任意

13.1.2 表現アセットのためのオプション

表 13.2 PDF_load_asset( ) で type=Rendition の場合のオプション

オプション	説明
autoplay	（論理値）true にすると、メディアがアクティブにされた時に自動再生されます。そうでない場合には、メディアは一時停止のままとなります。デフォルト：true
alttext	（ハイパーテキスト文字列）JavaScript を通じてアクセスできる、表現に対する代替テキスト
baseurl	（ハイパーテキスト文字列）メディアデータ内で見つかるすべての相対 URL を解決する際のベース URL として使いたい絶対 URL。デフォルト：PDF 文書の URL
controller	（論理値）true にすると、メディアを再生する際にコントローラーユーザーインターフェイスを表示します。デフォルト：false
duration	（非負 float かキーワード）メディアの再生時間（その本来の長さ以下）。秒単位で、または以下のキーワードを用いて指定できます（デフォルト：intrinsic）： ***infinite*** メディアの再生を止めません。 ***intrinsic*** メディアの通常の長さを再生します。
external	（論理値。url を与えている場合には無視されます）true にすると、ファイルの内容が PDF に埋め込まれずに、外部ファイルへの参照が作成されます。絶対ファイル名を避け、相対名だけを使うことを推奨します。デフォルト：false
mimetype	（文字列。必須）ファイルか URL の MIME タイプ。正しく再生させるにはここで表現の内容種別を正しく記述する必要があります。
monitor	（キーワード。style=floating・fullscreen の場合にのみ可。マルチモニター環境でのみ意味を持ちます）ウインドウを表示させたいモニター（デフォルト：document）： ***bestcolor*** 色深度が最も大きいモニター ***document*** 文書ウインドウの最も大きな部分を含んでいるモニター ***largest*** ピクセルが最も多いモニター ***nondocument*** 文書ウインドウの最も小さな部分を含んでいるモニター ***primary*** プライマリーモニターが得られるならそれ。得られないなら documentlarge と同じ ***tallest*** 縦のピクセルが最も多いモニター ***widest*** 横のピクセルが最も多いモニター
name	（ハイパーテキスト文字列。必須）表現名。文書内のすべての表現のうちで一意である必要があります。
nametree	（論理値）表現アセットを、文書の renditions 名称ツリーに挿入します。そうすると JavaScript からアクセス可能になります。デフォルト：false
opacity	（float かパーセント値。style=fullscreen・hidden の場合には無視されます）背景不透明度。デフォルト：1.0
permissions	（キーワード。external か url を与えている場合には許されません）メディアを再生するための一時ファイルを PDF ビューアーが生成することを許される条件（デフォルト：access）： ***access*** アクセシビリティー目的を含め内容抽出を文書が許可している場合にのみ許されます（表 3.3 のオプション permissions を参照） ***always*** つねに許されます ***extract*** 内容抽出を文書が許可している場合にのみ許されます（表 3.3 のオプション permissions を参照） ***never*** 全く許されません

表 13.2 PDF_load_asset( ) で type=Rendition の場合のオプション

オプション	説明
style	（キーワード）表現を再生するウインドウスタイル（デフォルト：annotation）： ***annotation*** 表現に紐付いているスクリーン注釈の領域 ***floating*** フローティングウインドウ（window オプションが必須となります） ***fullscreen*** フルスクリーンウインドウ。同じモニターにある他のウインドウをすべて覆い隠すことになります。 ***hidden*** 隠しウインドウ。音声再生に有用です
url	（論理値）true にすると、filename 引数がビデオファイルかストリーミングメディアを参照している URL として解釈されます。Acrobat DC は RTMP（Real Time Messaging Protocol）・HTTP・HTTPS プロトコルに対応しています。HTTP・HTTPS サーバー上では H.264 準拠 MOV・MP4 動画に対応しています。デフォルト：false
volume	（範囲 0 ～ 1 の float かパーセント値）音量を、録音されている音量レベルに対する割合として指定します。0% はミュート、100% は最大となります。デフォルト：100%
window	（オプションリスト。style=floating の場合にのみ可、かつその場合には必須）フローティングウインドウのためのサブオプション群を表 13.3 に従って指定します。

表 13.3 PDF_load_asset( ) で type=Rendition かつ style=floating の場合のオプション window のサブオプション

オプション	説明
align	（キーワード 2 個のリスト）over オプションで指定したウインドウを基準としてフローティングウインドウをどこに位置させるかを指定します。1 個目のキーワードは横位置を表し、left か center か right にする必要があります。2 個目のキーワードは縦位置を表し、bottom か center か top にする必要があります。デフォルト：{center center}
canresize	（キーワード）フローティングウインドウをリサイズできるか、またどのようにできるかを指定します（デフォルト：no）： ***no*** フローティングウインドウをリサイズできない。 ***keepratio*** フローティングウインドウは、アスペクト比が温存されるかぎりにおいてリサイズできる。 ***yes*** フローティングウインドウは、アスペクト比が温存されなくてもリサイズできる。
hasclose	（論理値。hastitle=true の場合にのみ可）true にすると、フローティングウインドウをユーザーが閉じることができます。デフォルト：true
hastitle	（論理値。Windows 版 Acrobat でのみ意味を持ちます）true にすると、フローティングウインドウにタイトルバーが表示されます。デフォルト：true
height	（正の整数。必須）フローティングウインドウの高さ（タイトルバーを含めず）をピクセル単位で
ifoffscreen	（キーワード）フローティングウインドウが完全にか部分的に画面外に位置したら、すなわち見えないならどうするか（デフォルト：）： ***allow*** 特にどうもしない ***forceonscreen*** ウインドウが画面に収まるよう移動させるかリサイズ。 ***cancel*** 表現の再生をキャンセル。

表 13.3 PDF_load_asset( ) で type=Rendition かつ style=floating の場合のオプション window のサブオプション

オプション	説明
over	（キーワード）フローティングウインドウが位置の基準とするべきウインドウ（デフォルト：pagewindow）： ***appwindow*** アプリケーションウインドウ ***desktop*** デスクトップ全体 ***monitor*** monitor オプションで指定したモニター ***pagewindow*** 文書ウインドウ
title	（文字列。hastitle=true の場合にのみ可）フローティングウインドウのタイトルバーのためのテキスト
width	（正の整数。必須）フローティングウインドウの幅（タイトルバーを含めず）をピクセル単位で

13.1.3 その他のアセット種別のためのオプション

表 13.4 PDF_load_asset( ) で type=3D/Attachment/JavaScript/Sound/Video の場合の、または PDF_add_portfolio_folder/file( ) の場合の、または PDF_begin/end_document( ) の attachments オプションのサブオプションとして用いるためのオプション

オプション	説明
description	（ハイパーテキスト文字列。PDF 1.6。PDF/A-2/3・PDF/UA-1 では推奨。type=3D・JavaScript の場合には不可。）そのファイルに紐付けられる説明テキスト。
document-attachment	（論理値。PDF/A-3・PDF 2.0 でのみ可。type=Attachment でのみ可。PDF_add_portfolio_file/folder( ) では不可）その連携ファイルを、文書レベル添付としても保管します。これは、PDF ビューアーのユーザーインターフェイスで添付群をリストするために有用でしょう。デフォルト：false
external	（論理値。PDF/A では false とする必要があります。type=3D・JavaScript と attachments オプションと PDF_add_portfolio_file/folder( ) では不可）true にすると、そのファイルの内容は PDF 内に埋め込まれず、外部ファイルへの参照が作成されます。絶対ファイル名を避けて相対ファイル名だけを使うことを推奨します。デフォルト：false
filename	（名前文字列）そのファイルの名前。その内容は、指定する種別によって、「詳細」の項の諸要件に準拠している必要があります。このオプションは、PDF_load_asset( )・PDF_add_portfolio_file/folder( ) のメソッド引数 filename を通じて与えることもできますが、PDF_begin/end_document( ) の attachments オプションとともに用いられる場合には必須です。external=true でない限りは、この filename の、ディレクトリ構成要素を除いたベース部分だけが PDF 出力へ書き出されます。
mimetype	（文字列。type=Sound・Video では必須。type=3D・JavaScript と PDF_add_portfolio_folder( ) では不可）そのファイルの MIME タイプ。デフォルト：application/octet-stream
name	（ハイパーテキスト文字列。PDF_add_portfolio_folder( ) では不可。type=3D・JavaScript では不可）その添付の名前。デフォルト：filename からパス構成要素を除いたもの
password	（最長 127 キャラクターの文字列。PDF/A-2 では不可。type=Attachment でのみ可。PDI が利用可能な場合にのみ可。PDF_add_portfolio_folder( ) では不可）保護された PDF 文書を、その日付エントリー群を取得するために開くために必要なマスターパスワード。

表 13.4 PDF_load_asset( ) で type=3D/Attachment/JavaScript/Sound/Video の場合の、または PDF_add_portfolio_folder/file( ) の場合の、または PDF_begin/end_document( ) の attachments オプションのサブオプションとして用いるためのオプション

オプション	説明
relationship	（ハイパーテキスト文字列。PDF 2.0・PDF/A-3 でのみ可。type=Attachment でのみ可。PDF_add_portfolio_folder( ) では不可）そのファイルの、それが紐付けられる文書の部分に対する関係。この値は任意の文字列にできますが、以下のキーワードが定義済みです（デフォルト：Unspecified）： ***Alternative*** そのファイルは代替表現である（オーディオなど） ***Data*** そのファイルは視覚表現を導出するために用いられる情報を表している（表やグラフのための CSV データなど） ***FormData*** そのファイルは、その文書の中のフォームフィールド群に紐付けられたデータを表している。 ***Schema*** そのファイルは、XMP メタデータのための XML スキーマなどスキーマ定義を内容としている。 ***Source*** そのファイルは元ソース素材である（その文書に紐付けられたワードプロセッサー文書や、画像に紐付けられたスプレッドシートなど）。 ***Supplement*** そのファイルは、元ソースまたはデータの、より容易に消費可能であろう補足表現を表している（画像内の数式の MathML 表現など）。 ***Unspecified*** その関係は未知である、ないしは他のキーワードで表現できない。
thumbnail	（画像ハンドル。type=Attachment でのみ可。PDF 1.7ext3）そのファイルかフォルダーのためのサムネールとして使用したい画像。このハンドルは、PDF_load_image( ) を用いて作成されている必要があり、かつこの画像は、ICC プロファイルなしのグレースケールまたは RGB 画像である必要があります。ですので RGB 画像は、iccprofile=none を用いて読み込まれている必要があります。Acrobat は、添付のためのサムネールを無視します。

13.2 リッチメディア注釈のためのサブオプション

表 13.5 PDF_create_annotation( ) の type=RichMedia の場合の richmedia オプションのサブオプション

オプション	説明
activate	（オプションリスト）表 13.6 に従った、表現のスタイル、デフォルトスクリプト動作、デフォルトビュー情報、およびそのアニメーションが起動された際のアニメーションスタイルを指定したオプションリスト。
assets	（1 個ないし複数のオプションリストのリスト。必須）参照されることのできる名前付きアセット： ***asset*** （アセットハンドル。必須）3D・JavaScript・Sound・Video いずれかのアセット ***name*** （1 ～ 255 キャラクターのハイパーテキスト文字列。キャラクター: * ″ < > | を使ってはいけません。末尾キャラクターをピリオド「.」としてはいけません。必須）アセットの名前
configuration	（オプションリスト。必須）この構成リストは、1 個ないし複数のインスタンスオプションリストを内容とします： ***instances*** （オプションリストのリスト。必須）リストはそれぞれ、アセット 1 個を、注釈 1 個のアートワークを入れ込むための設定とともに指定します： ***asset*** （ハイパーテキスト文字列。必須）assets オプションで指定されたアセットの名称。種別 3D・Sound・Video いずれかのリッチメディアアセットの名前のみをここでは指定できます。 ***name*** （ハイパーテキスト文字列）この構成の一意名。 ***type*** （キーワード）この構成に対する主要内容種別。使えるキーワードは 3D・Sound・Video です。デフォルト：instances サブオプションの 1 個目の要素の種別
deactivate	（オプションリスト）読み込み解除（リスタートか一時停止）の条件： ***condition*** （キーワード）この注釈がいつ起動解除されるかを指定します（デフォルト：clicked）： ***clicked*** この注釈は、ユーザーアクションかスクリプトによって明示的に起動解除される。 ***closed*** この注釈は、この注釈を内容とするページがカレントページとしてのフォーカスを失った時にただちに起動解除される。 ***invisible*** この注釈は、ページが見えなくなった時にただちに起動解除される。
views	（3D ビューハンドルのリスト）PDF_create_3dview( ) によって返された 3D ビューハンドル。ビューが指定されない場合には、3D ビューの、レンダリング / ライトニングモード群、背景色、カメラデータといった構成要素群に対して、デフォルト値群が使用されます。デフォルト：空リスト

表 13.6 PDF_create_annotation( ) の richmedia オプションの activate サブオプションのサブオプション

オプション	説明
animation	（オプションリスト）そのアートワーク内に存在するキーフレームアニメーション群をドライブさせる方式として望ましいもの： ***playcount*** （整数）非負数は、このアニメーションが再生される回数を表します。負数は、このアニメーションが無限に繰り返されることを示します。デフォルト：-1 ***speed*** （正の float）1 より大きい値は、このアニメーションを再生するためにかかる時間を短縮、つまりこのアニメーションをスピードアップします。これを利用すると、作成者は、アニメーションの内容をオーサリングしなおすことなく、その望む速度を変えることができます。デフォルト：1 ***style*** （キーワード）アニメーションスタイル（デフォルト：none）： ***none*** キーフレームアニメーション群は、ビューアーによって駆動されるべきではない。この値は、JavaScript を通じてアニメーションを駆動する文書によって用いられます。この animation オプションの残りのサブオプション群は無視されます。 ***linear*** キーフレームアニメーションは最初から最後まで線形に駆動される。これは繰り返し再生になります。 ***oscillating*** キーフレームアニメーションはその時間範囲に沿って往復する。これは往復再生になります。
condition	（キーワード）この注釈がいつ起動されるかを指定します（デフォルト：clicked）： ***clicked*** この注釈は、ユーザーアクションかスクリプトによって明示的に起動される。 ***opened*** この注釈は、この注釈を内容とするページがフォーカスを得た時に起動される。 ***visible*** この注釈は、そのページの任意の部分が見えた時に起動される。
presentation	（表 13.7 に従ったオプションリスト）この注釈とユーザーインターフェイス要素群がレイアウトされ描画される方式を指定します
scripts	（ハイパーテキスト文字列のリスト）この richmedia オプションの assets サブオプションで指定している JavaScript アセット群の名前群。
view	（キーワードか 3D ビューハンドル）3D リッチメディアの起動ビュー。ハンドルは、richmedia オプションリストの views サブオプションにも含まれている必要があります。この views オプションが指定されない場合には、3D ビューの構成要素群に対するデフォルト値群が用いられます。使えるキーワード（デフォルト：first）： ***first*** richmedia オプションリストの views サブオプションの 1 番目の要素

表 13.7 PDF_create_annotation( ) の richmedia オプションの activate サブオプションの presentation サブオプションのサブオプション

オプション	説明
navigation-pane	（論理値）ナビゲーションパネルユーザーインターフェイス要素のデフォルト動作。true にすると、この内容が初期起動される際にナビゲーションパネルは表示されます。デフォルト：false
passcontext-click	（論理値）このリッチメディア注釈上でのコンテキストクリックがメディアプレーヤーランタイムへ渡されるか、それとも PDF ビューアーによって処理されるかを指定します。false にすると、PDF ビューアーがこのコンテキストクリックを処理します。true にすると、PDF ビューアーのコンテキストメニューは表示されず、ユーザーはメディアプレーヤーランタイムによって生成されたコンテキストメニューと任意のカスタムアイテム群を目にします。デフォルト：false
style	（キーワード）このリッチメディアがどのように表示されるかを指定します（デフォルト：embedded）： ***embedded*** PDF ページ内に埋め込まれて ***windowed*** window オプションリストによって指定された別ウインドウ内で

表 13.7 PDF_create_annotation( ) の richmedia オプションの activate サブオプションの presentation サブオプションのサブオプション

オプション	説明
toolbar	（論理値）この注釈に付いているインタラクティブツールバーのデフォルト動作。true にすると、この注釈が起動されてフォーカスを与えられた時にツールバーが表示されます。デフォルト：構成種別 3D の場合には true、それ以外なら false
transparent	（論理値）ページ内容がこのリッチメディア内容の透過領域を通して表示されるかどうかを指定します。true にすると、このリッチメディアアートワークはページ内容の上にアルファチャンネルを用いて合成されます。そうでないなら、このリッチメディアアートワークは、不透明な背景の上に描画されます。デフォルト：false
window	（オプションリスト）style=windowed に対するフローティングウインドウのサイズと位置。座標は usercoordinates オプションに応じてユーザーまたはデフォルト座標で表されます。 その動作は注釈オプション zoom・rotate がに設定されている場合と同様です： ***heightdefault*** （float）このウインドウのデフォルト高さ。デフォルト：216（デフォルト座標で） ***widthdefault*** （float）このウインドウのデフォルト幅。デフォルト：288（デフォルト座標で）

13.3 3D アートワーク

3D 内容を実装するには、*3D* 注釈か *RichMedia* 注釈を用います。*RichMedia* 注釈のほうが新しく、*3D* 注釈よりも多くのスクリプティングイベントを提供しています。この 2 つの方式の違いと共通点を表 13.8 にまとめます。

表 13.8 PDF_create_3dview( ) のオプション

	3D 注釈	3D 内容を持つ RichMedia 注釈
PDF 互換性	PDF 1.6	PDF 1.7ext3
3D データを読み込む	PDF_load_3ddata( )	PDF_load_asset( ) で type=3D
注釈	PDF_create_annotation( ) で type=3D。関連するオプション群を 288 ページ「13.3.3 3D 注釈のためのオプション」で説明しています。	PDF_create_annotation( ) で type=RichMedia。richmedia オプションの関連するサブオプション群を 280 ページ「13.2 リッチメディア注釈のためのサブオプション」で説明しています。
3D データを適用	PDF_create_annotation( ) : オプション 3Ddata	PDF_create_annotation( ) : オプション richmedia/assets/asset
3D ビューを作成	PDF_create_3dview( ) （283 ページ「13.3.1 RichMedia・3D 注釈のための 3D ビュー」参照）	
3D ビューを適用	PDF_load_3ddata( ) : オプション defaultview・views PDF_create_annotation( ) : オプション 3Ddinitialview PDF_create_action( ) で type=GoTo3DView : オプション 3Dview	PDF_create_annotation( ) : オプション richmedia、サブオプション views
JavaScript を読み込む	PDF_load_asset( ) で type=JavaScript	
JavaScript を適用	PDF_load_3ddata( ) : オプション javascript	PDF_create_annotation( ) : オプション richmedia/activate/scripts
3D アクションを作成	PDF_create_action( ) で type=3Dview	PDF_create_action( ) で type=RichMediaExecute
3D アクションを適用	PDF_create_annotation( ) : オプション action	

13.3.1 RichMedia・3D 注釈のための 3D ビュー

C++ Java C# ***int create_3dview(String username, String optlist)***

Perl PHP ***int create_3dview(string username, string optlist)***

C ***int PDF_create_3dview(PDF *p, const char *username, int len, const char *optlist)***

3D ビューを作成します。

username （ハイパーテキスト文字列）3D ビューのユーザーインターフェイス名。

len （C 言語バインディングのみ）*username* の長さ（バイト単位）。*len=0* にすると null 終了文字列を与える必要があります。

optlist 3D ビューのプロパティ群を指定したオプションリスト：

- 一般オプション：*errorpolicy*（表 1.5 参照）

- 3D ビューのプロパティ群を表 13.9 に従って指定するオプション：*background*・*camera2world*・*cameradistance*・*lighting*・*name*・*rendermode*・*type*・*U3Dpath*
- C の場合と、Perl・PHP・Ruby で *stringformat=legacy* の場合のためのエンコーディングオプション：*hypertextencoding*（表 2.1 参照）

戻り値 3D ビューハンドル。カレントの**文書**スコープを終えるまで使えます。*errorpolicy=return* の場合、戻り値 -1（PHP では 0）はエラーを示すので、そうでないかを呼び出し側で検査する必要があります。

詳細 3D ビューハンドルは、*PDF_load_3ddata()* の *views* オプションで 3D モデルに関連づけることもできますし、あるいは *PDF_create_annotation()* で 3D 注釈を作成したり、*PDF_create_action()* で 3D 関連のアクションを作成したりするのに使うこともできます。

スコープ **オブジェクト**以外任意。返されたハンドルは、次に *PDF_end_document()* を呼び出すまで使えます。要 PDF 1.6

表 13.9 PDF_create_3dview( ) のオプション

オプション	説明
background	（オプションリスト）3D モデルの背景： ***fillcolor*** （色）背景色を RGB 色空間で指定します。デフォルト：白 ***entire*** （論理値）true にすると、背景は注釈全体に適用されます。そうでなければ、注釈の 3Dbox オプションで指定してある長方形にだけ適用されます。デフォルト：false
camera2world	（float 12 個のリスト）カメラの位置と向きを世界座標で指定した 3D 変換行列（以下説明参照）。デフォルト：3D モデルの中で内部的に定義された初期ビュー
camera-distance	（float。負値は不可。camera2world を指定していない場合には無視されます）カメラと軌道中心の距離。詳しくは、ISO 32000-1 の section 13.6.4「3D Views」の C0 キーの説明を参照。デフォルト：3D データの中で内部的に定義
lighting	（オプションリスト。PDF 1.7）3D アートワークの照明方式： ***type*** （キーワード）照明方式を指定します。使えるキーワード（デフォルト：Artwork）： ***Artwork*** 光は 3D アートワーク内で指定される。 ***None*** 光なし。3D アートワーク内で指定されている光は無視されます。 ***White*** 3 個のライトグレー無限遠光、アンビエント項なし ***Day*** 3 個のライトグレー無限遠光、アンビエント項なし ***Night*** 黄色 1 個・アクア色 1 個・青 1 個の無限遠光、アンビエント項なし ***Hard*** 3 個のグレー無限遠光、中程度のアンビエント項 ***Primary*** 赤 1 個・緑 1 個・青 1 個の無限遠光、アンビエント項なし ***Blue*** 3 個の青色無限遠光、アンビエント項なし ***Red*** 3 個の赤色無限遠光、アンビエント項なし ***Cube*** 長軸に沿って配置された 6 個のグレー無限遠光、アンビエント項なし ***CAD*** 3 個のライトグレー無限遠光とカメラに付いた光 1 個、アンビエント項なし ***Headlamp*** カメラに付いた 1 個の無限遠光、低いアンビエント項
name	（ハイパーテキスト文字列）3D ビューの名前。GoTo アクションで使えます。これは必須ではない内部的な名前であり、必須の username 引数とは別個に扱われます。
rendermode	（オプションリスト。PDF 1.7）3D アートワークを表示するためのレンダーモードを表 13.10 に従って指定します。

表 13.9 PDF_create_3dview( ) のオプション

オプション	説明
type	（キーワード。このビューが PDF_load_3ddata( ) で type=PRC で使われる場合には必須）3D データの種別（デフォルト：U3D）： ***PRC*** このビューは、PDF_load_3ddata( ) で type=PRC で使用される。 ***U3D*** このビューは、PDF_load_3ddata( ) で type=U3D で使用される。
U3Dpath	（ハイパーテキスト文字列。camera2world オプションが指定されている場合には無視されます。type=U3D の場合み）3D アートワーク内のビューノードにアクセスするために用いられるビューノード名。

カメラ位置　カメラの位置は *camera2world* オプションで指定することができます。あるいは、JavaScript コードを付けて、カメラのモデルに対する位置と向きを指定することもできます。PDFlib クックブックに、JavaScript コードを 3D モデルに付けるサンプルコードがあります。

以下の値を *camera2world* オプションに与えると、代表的なカメラ位置を実現することができます。*x・y・z* は、カメラの位置を記述する適切な値です。これらの値は、定められた条件を満たす必要があります：

前からのビュー：

```
{-1 0 0 0 0 1 0 1 0 x y z} x小、y大負、z小
```

左からのビュー：

```
{ 0 1 0 0 0 1 1 0 0 x 0 z} x大負、z小
```

上からのビュー：

```
{-1 0 0 0 1 0 0 0 -1 x 0 z} x小、z大負
```

後ろからのビュー：

```
{ 1 0 0 0 0 1 0 -1 0 x y z} x小、y大正、z小
```

下からのビュー：

```
{-1 0 0 0 -1 0 0 0 1 x 0 z} x小、z大負
```

右からのビュー：

```
{ 0 -1 0 0 0 1 -1 0 0 x 0 z} x大正、z小
```

等角ビュー、すなわち投影の方向が 3 軸すべてと等しい角度で交わる場合。このようなビューは 8 個、八分空間ごとに 1 個ずつあります：

```
{0.707107 -0.707107 0 -0.5 -0.5 0.707107 -0.5 -0.5 -0.707107 x y z}
x, y, z large positive
```

x・y・z 値は、モデルの位置とサイズに応じて選ぶ必要があります。「大」は、カメラとモデルの間に充分大きな距離をとるために、その値をモデルのサイズよりかなり大きくする必要があることを意味します。値が大きすぎると、モデルは非常に小さく表示され、そしてビューを回転させるとすぐに視界から外れてしまいます。値が小さすぎると、モデルは

ビューに収まりきりません。「小」は、その絶対値を、大きな値に比べて小さし、モデルのサイズをあまり大きく超えないようにする必要があることを意味します。

表 13.10 PDF_create_3dview( ) の rendermode オプションのサブオプション

オプション	説明
crease	（float で範囲 0 ～ 180）折り目値
facecolor	（RGB カラーまたはキーワード、type=Illustration のみ）表面色。キーワード backgroundcolor はカレント背景色を意味します。デフォルト：backgroundcolor
opacity	（float で範囲 0 ～ 1、またはパーセント値）いくつかのレンダーモードにおける不透明値。デフォルト：0.5
rendercolor	（RGB カラー）いくつかのレンダーモードで用いられる補助色。デフォルト：黒
type	（キーワード。PDF 1.7）3D アートワークを表示するためのレンダーモード（デフォルト：Artwork）： ***Artwork*** レンダーモードは 3D アートワーク内で指定される。rendermode オプションのこれ以外のサブオプションはすべて無視されます。 ***Solid*** テクスチャ付きで照明された幾何図形輪郭群を表示。 ***SolidWireframe*** テクスチャ付きで照明された、単色の辺を持った幾何図形輪郭（三角形）群を表示。 ***Transparent*** テクスチャ付きで照明された、追加の透過の水準を持った幾何図形輪郭（三角形）群を表示。 ***TransparentWireframe*** テクスチャ付きで照明された、追加の透過の水準を持った幾何図形輪郭（三角形）群を表示。 ***BoundingBox*** テクスチャ付きで照明された、追加の透過の水準を持った、単色の不透明な辺を持った幾何図形輪郭（三角形）群を表示。 ***TransparentBoundingBox*** 各節点の、その節点に対する局所座標空間の軸に沿った、追加の透過の水準を持った境界枠の面を表示。 ***TransparentBoundingBoxOutline*** 各節点の、その節点に対する局所座標空間の軸に沿った、追加の透過の水準を持った境界枠の辺と面を表示。 ***Wireframe*** 各節点の、その節点に対する局所座標空間の軸に沿った、追加の透過の水準を持った境界枠の辺と面を表示。 ***ShadedWireframe*** 辺のみを、ただしその 2 個の頂点の間でその色を補間し、かつ照明を適用して表示。 ***HiddenWireframe*** 辺を単色で、ただし後ろ向きの辺と隠された辺を除いて表示。 ***Vertices*** 頂点のみを単色で表示。 ***ShadedVertices*** 頂点のみを、ただしその頂点色を用いて、かつ照明を適用して表示。 ***Illustration*** 表面を持ったシルエット辺を、隠線を除いて表示。 ***SolidOutline*** 照明されテクスチャ付きの表面を持ったシルエット辺を、隠線を除いて表示。 ***ShadedIllustration*** 照明されテクスチャ付きの表面を持った、アートワークの弱く照明された領域を除くための追加の放射項目を持ったシルエット辺を表示。

13.3.2 3D 注釈の内容を読み込む

C++ Java C# ***int load_3ddata(String filename, String optlist)***

Perl PHP ***int load_3ddata(string filename, string optlist)***

C ***int PDF_load_3ddata(PDF *p, const char *filename, int len, const char *optlist)***

3D モデルを、ディスクベースまたは仮想ファイルから読み込みます。

filename （名前文字列。*filenamehandling* グローバルオプションに従って解釈されます。表 2.1 参照）3D モデルの入った、ディスクベースまたは仮想ファイルの名前。

len （C 言語バインディングのみ）*filename* の長さ（バイト単位）。*len=0* にすると null 終了文字列を与える必要があります。

optlist 3D モデルのプロパティ群を指定したオプションリスト：

- 一般オプション：*errorpolicy*（表 1.5 参照）
- 表 13.11 に従って 3D モデルのプロパティ群を指定するオプション：*animation*・*defaultview*・*javascript*・*script*・*type*・*views*
- C の場合と、での場合のエンコーディングオプション：*hypertextencoding*（表 2.1 参照）

戻り値 3D ハンドル。*PDF_create_annotation( )* で *type=3D* を用いて 3D 注釈を作成するのに使えます。3D ハンドルは、カレントの**文書**スコープを終えるまで使えます。*errorpolicy=return* の場合、戻り値 -1（PHP では 0）はエラーを示すので、そうでないかを呼び出し側で検査する必要があります。

詳細 ファイルは、PRC または U3D 形式の 3D データを内容として持つ必要があります。

スコープ **オブジェクト**以外任意。返されたハンドルは、次に *PDF_end_document( )* を呼び出すまで使えます。要 PDF 1.6

表 13.11 PDF_load_3ddata( ) のオプション

オプション	説明
animation	（オプションリスト）そのアートワーク内に存在するキーフレームアニメーション群をドライブさせる方式として望ましいもの。使えるオプションは playcount・speed・style。PDF_create_annotation( ) の richmedia オプションの activate サブオプションの animation サブオプションのためのものと同じ。表 13.6 参照）
defaultview	（キーワードまたは 3D ビューハンドル）3D 注釈の初期ビュー。キーワード first・last（views オプションの中の各項目を参照する）のいずれか、または PDF_create_3dview( ) で作成した 3D ビューハンドル。デフォルト：first
javascript	（type=JavaScript を用いて読み込んであるアセットハンドル。オプション script・javascript をどちらか 1 つだけ指定することが必須）3D モデルがインスタンス化された時に実行させたい JavaScript
script	（ハイパーテキスト文字列）3D モデルが実体化された時に実行させたい JavaScript コードを入れた文字列。デフォルト：スクリプトなし

表 13.11 PDF_load_3ddata( ) のオプション

オプション	説明
type	（キーワード）3D データの種類（デフォルト：U3D）： ***PRC*** （PDF 1.7ext3）Product Representation Compact（PRC）形式（ISO 14739–1） ***U3D*** （PDF 1.6）Universal 3D File 形式（U3D）。以下の種類が使えます：Universal 3D File Format (U3D), 1st Edition・ECMA–363, Universal 3D File Format (U3D), 3rd Edition。
views	（3D ビューハンドルのリスト）3D モデルに対するビューのリスト。リストの各要素は、PDF_create_3dview( ) で作成した 3D ビューハンドルです。ビュー群を PDF_create_3dview( ) で作成した際に用いられた type オプションは、PDF_create_3ddata( ) 内の type オプションと一致している必要があります。デフォルト：空リスト

13.3.3 3D 注釈のためのオプション

表 13.12 PDF_create_annotation( ) で type=3D の場合の 3D オプション

オプション	説明
3Dactivate	（オプションリスト）3D 注釈をアクティブにするタイミングと、アクティブ・非アクティブにしたときの状態を指定します： ***enable*** （キーワード）注釈をいつ起動するべきかを指定します（デフォルト：click）： ***open*** ページを開いた時に起動。 ***visible*** ページが見えた時に起動。 ***click*** 注釈は、スクリプトかユーザーアクションで明示的に起動する必要がある。 ***enablestate*** （キーワード）初期アニメーションステータス（デフォルト：play）： ***pause*** 3D モデルが実体化されるが、スクリプトアニメーションは無効になる。 ***play*** 3D モデルが実体化される。スクリプトアニメーションがあるときはそれが有効になります。 ***disable*** （キーワード）注釈をいつ起動解除するべきかを指定します（デフォルト：invisible）： ***close*** ページを閉じた時に起動解除。 ***invisible*** ページが見えなくなった時に起動解除。 ***click*** 注釈は、スクリプトかユーザーアクションで明示的に起動解除する必要がある。 ***disablestate*** （キーワード）注釈を起動解除した際の状態（デフォルト：reset）： ***pause*** 3D モデルはレンダリングできるが、アニメーションは無効になる。 ***play*** 3D モデルはレンダリングでき、アニメーションは有効になる。 ***reset*** 3D モデルを使う前の任意の初期状態。 ***modeltree*** （論理値。PDF 1.6）true にすると、注釈のアクティベーション時にモデルツリーナビゲーションタブが開かれます（デフォルト：false） ***toolbar*** （論理値。PDF 1.6）true にすると、注釈のアクティベーション時に 3D ツールバー（注釈の上部の）が表示されます（デフォルト：true）
3Ddata	（3D ハンドル。必須）PDF_load_3ddata( ) で作成した 3D ハンドル。
3Dinteractive	（論理値）true にすると、3D モデルはインタラクティブな用途を想定します。false にすると、JavaScript で動かされることを想定します。デフォルト：true
3Dshared	（論理値）true にすると、3Ddata オプションで指定している 3D データは、間接的に参照されます。同じデータを参照する複数の 3D 注釈は、そのモデルのただ 1 つの動作時実体を共有します。これはすなわち、変更はそのような注釈すべてにおいて同時に見えることを意味します。デフォルト：false

表 13.12 PDF_create_annotation( ) で type=3D の場合の 3D オプション

オプション	説明
3Dinitialview	（キーワードまたは 3D ビューハンドル）3D モデルの初期ビューを、キーワード first・last・（モデルの views オプションの中の各項目を参照する）・default（モデルの defaultview オプションを参照する）のいずれか 1 つか、または PDF_load_3ddata( ) で作成した 3D ビューハンドルで指定します。デフォルト：default

13.4 地理空間機能

地理空間機能は、以下のメソッドとオプションで実装されます：

- *PDF_begin/end_page_ext()* の *viewports* オプションを用いて、ページに1個ないし複数の地理参照付き領域を割り当てることができます。
- *PDF_load_image()* の *georeference* オプションを用いて、画像に地球ベースの座標系を割り当てることができます。この方式は、Acrobat では動作しますが、Avenza PDF Maps アプリでは対応していません（試験したバージョン：iOS 用 Avenza 2.7・Android 用 1.7）。
- *PDF_open_pdi_page()*・*PDF_load_graphics()*・*PDF_begin_template_ext()* の *georeference* オプションを用いて地球ベースの座標系をフォーム XObject に割り当てることもできます。しかしこの方式は、Acrobat DC を含むいずれの既知のビューアーにおいても対応していないため、推奨しません。

地理空間機能のためのオプション群を、表 13.13 以降に詳しく示します。

表 13.13 PDF_begin/end_page_ext( ) の viewports オプションのサブオプション

オプション	説明
bounding-box	（長方形。必須）ページ上のビューポートの位置をデフォルト座標で指定した長方形。
georeference	（表 13.14 に従ったオプションリスト。必須）地理空間計測に用いるためビューポートに関連付けられる世界座標系の記述
hypertext-encoding	（キーワード）name オプションのエンコーディング。空文字列は unicode と同等です。デフォルト：hypertextencoding グローバルオプションの値
name	（ハイパーテキスト文字列）ビューポートの説明タイトル（地図名）。ただし、Acrobat はこのビューポート名をユーザーインターフェイスに表示しません。

表 13.14 PDF_load_image( )PDF_open_pdi_page( )・PDF_load_graphics( )・PDF_begin_template_ext( ) の georeference オプションのサブオプション。PDF_begin/end_page_ext( ) の viewports オプションの georeference サブオプションのサブオプションとしても用いられます

オプション	説明
angularunit	（キーワード）求める角度表示単位（デフォルト：deg）： *degree* 度 *grad* グラード（全円の 1/400、すなわち 0.9 度）
areaunit	（キーワード）求める面積表示単位（デフォルト：sqm）： *sqm* 平方メートル *ha* ヘクタール（10,000 平方メートル） *sqkm* 平方キロメートル *sqft* 平方フィート *a* エーカー *sqmi* 平方マイル ここで指定した単位は、以下の Acrobat の設定を無効にしているときのみ、表示に用いられます：「環境設定」→「ものさし（地図情報）」→「デフォルトの面積単位を使用」。
bounds	（折れ線で点 2 個以上）地理空間変換が有効となる領域の境界（地図では、この境界折れ線は図郭線として知られています）。点群は、ページビューポートの境界枠に対して、またはテンプレートか画像の寸法に対して相対的に表現します。デフォルト：{0 0 0 1 1 1 1 0}、すなわちビューポート・テンプレート・画像領域全体が地図に用いられます。

表 13.14 PDF_load_image( )PDF_open_pdi_page( )・PDF_load_graphics( )・PDF_begin_template_ext( ) の georeference オプションのサブオプション。PDF_begin/end_page_ext( ) の viewports オプションの georeference サブオプションのサブオプションとしても用いられます

オプション	説明
displaysystem	（オプションリスト）緯度・経度といった位置の値をユーザーに見せる表示に用いる座標系を表 13.15 に従って指定します。この項目を利用すると、地図を指定するために coords オプションで与えた座標系以外の座標系で座標を表示することができます。
linearunit	（キーワード）求める距離表示単位（デフォルト：m）： ***m*** メートル ***km*** キロメートル ***ft*** 国際フィート ***usft*** 米国測量フィート ***mi*** 国際マイル ***nm*** 海里 ここで指定した単位は、以下の Acrobat の設定を無効にしているときのみ、表示に用いられます：「環境設定」→「ものさし（地図情報）」→「デフォルトの距離単位を使用」。
mappoints	（float ペア 2 個以上のリスト。必須）数値のリストで、各ペアが 2D 単位正方形内の点を定義します。この単位正方形は、ページビューポートの、または georeference オプションを含むテンプレートページか PDI ページか画像の長方形境界へマップされます。この mappoints リストは、worldpoints リストと同じ数の点を内容として持つ必要があり、各点は、worldpoints リスト内の地理空間位置に対応する単位正方形内の地図位置となります。
worldpoints	（float ペア 2 個以上のリスト。必須）座標ペアのリストで、各ペアが、mappoints オプション内のそれぞれの点の世界座標を指定します。このペアの数は、mappoints オプション内のペアの数と一致する必要があります。座標値は、worldsystem オプションで指定した座標系に基づきます：type=geographic のときは、緯度・経度値を度単位で与える必要があります。type=projected のときは、投影された x・y 値を与える必要があります。
worldsystem	（オプションリスト。必須）表 13.15 に従った世界座標系（worldpoints の解釈のための）。

表 13.15 PDF_load_image( )・PDF_open_pdi_page( )・PDF_load_graphics( )・PDF_begin_template_ext( ) の georeference オプションの mapsystem・displaysystem サブオプションのサブオプション。PDF_begin/end_page_ext( ) の viewports オプションの georeference サブオプションのサブオプションとしても用いられます

オプション	説明
epsg	（整数。epsg か wkt のいずれか 1 つだけを必ず与える必要があります）座標系を EPSG 参照コードで指定します。なお、Acrobat は type=geographic のときは EPSG コードに対応していませんので、その場合は wkt を用いてください。
type	（キーワード。必須）座標系の種類： ***geographic*** 測地座標系（wkt のみ対応） ***projected*** 投影座標系（wkt にも epsg にも対応）
wkt	（文字列で最大 1024 ASCII キャラクター。epsg か wkt のいずれか 1 つだけを必ず与える必要があります）座標系を「Well Known Text」（WKT）の文字列で指定します。WKT は、EPSG コードを全く持たないカスタム座標系に対して推奨されます。

14 タグ付き PDF とマーク付きコンテンツ

この章の API メソッド：

- *PDF_begin_item()*
- *PDF_end_item()*
- *PDF_activate_item()*
- *PDF_begin_mc()*
- *PDF_end_mc()*

14.1 タグ付けメソッド

タグ付き PDF を生成するには、*PDF_begin_document()* で *tagged* オプションが *true* に設定されている必要があります。タグ付き PDF モードは、PDF/A-1a/2a/3a・PDF/UA 規格では自動的に起動されます。タグ付き PDF を作成する際には PDF/UA 規則群に従うことを強く推奨します。

C++ Java C# ***int begin_item(String tagname, String optlist)***

Perl PHP ***int begin_item(string tagname, string optlist)***

C ***int PDF_begin_item(PDF *p, const char *tagname, const char *optlist)***

タグ付き PDF のために構造エレメントかその他のコンテンツエレメントを開きます。

tagname そのアイテムのエレメント構造種別の名前。表 14.1 に従って、以下のグループの構造種別を使えます（説明は PDFlib チュートリアルを参照）：

- 標準種別群（標準種別群の詳しい説明は PDFlib チュートリアルにあります）
- 構造エレメントではない擬似種別群
- タグ名 *Plib_custom_tag* は、カスタム種別の使用を暗黙に前提します（これは *customtag=true* と等価です）。実際のタグ名を *tagname* オプションで与える必要があります。カスタム種別群は *rolemap* 文書オプションを必要とします。

あるいはタグ名を、この引数をオーバーライドする *tagname* オプションを通じて与えることもできます。

optlist アイテムの詳細を指定したオプションリスト。継承可能な設定はすべて子エレメントへ継承されますので、繰り返す必要はありません。アイテムのプロパティは後で変更できないので、すべてここで設定する必要があります。以下のオプションが使えます：

- 表 14.2 に従った処理オプション：
  associatedfiles・bookmark・contents・customtag・direct・elementid・id・index・metadata・namespace・parent・ref・tag・tagname・title・usercoordinates
- 表 14.3 に従った、構造エレメント属性のためのオプション：
  ActualText・Alt・ARIARole・artifactsubtype・artifacttype・Attached・BBox・Checked・ColSpan・ContinuedFrom・ContinuedList・Desc・E・Headers・Height・Lang・ListNumbering・Placement・PrintFieldRole・RowSpan・Scope・Short・Summary・Width・WritingMode

表 14.1 PDF_begin_item( ) ・ PDF_begin_mc( ) ・ PDF_mc_point( ) の、およびさまざまなメソッドの tag オプションを用いた短縮タグ付けの、標準・擬似・カスタム構造種別

カテゴリー	タグ
標準構造種別群	
グループ化	Document ・ DocumentFragment[1]
グループ化	Art ・ Aside[1] ・ BlockQuote ・ Caption ・ Div ・ Document ・ Index ・ NonStruct ・ Part ・ Private ・ Sect ・ TOC ・ TOCI
ブロックとサブブロック	FENote[1] ・ H ・ H1 ・ H2 ・ H3 ・ H4 ・ H5 ・ H6 H7[1] ・ H8[1] など（PDF/UA-1 でロールマッピングを用いる場合にも許されます） Sub[1] ・ Title[1]
ラベルと箇条書き	L ・ LI ・ Lbl ・ LBody
テーブル	Table ・ TR ・ TH ・ TD ・ THead[2] ・ TBody[2] ・ TFoot[2] （テーブルタグはすべて、PDF_fit_table( ) によって自動的に作成されることもできます。PDFlib チュートリアル参照）
インライン	Em[1] ・ BibEntry ・ Code ・ Lbl ・ Note ・ Quote ・ Reference ・ Span ・ Strong[1]
インタラクティブ	Annot[2] ・ Form ・ Link
イラストレーション	Figure ・ Formula
日本語	Ruby[2]（グループ化）・ RB[2] ・ RT[2] ・ RP[2] ・ Warichu[2]（グループ化）・ WT[2] ・ WP[2]
ページ装飾	Artifact[1]（直接出力される疑似エレメント Artifact ではなく標準アイテムとして）
擬似構造種別群	
非構造化エレメント群	***Artifact*** ページの本文から区別するべきページ装飾。 ***ASpan*** （アクセシビリティースパン。PDF には Span として書き込まれますが、直接的アイテム Span とは区別する必要があります）これを使うと、構造エレメントに属さないコンテンツや、構造エレメントの一部分に似たコンテンツに対して、アクセシビリティー属性群を与えることができます。 ***ReversedChars*** （廃止）文字の並び順が反対になっている右書きの言語のテキスト。 ***Clip*** （廃止）マーク付き切り抜きシーケンス。すなわち、クリッピングパスまたは表現モード 7 のテキストのみを指定します。表示されるグラフィックまたは PDF_save( ) / PDF_restore( ) を含みません。
カスタム構造種別群	
ユーザー定義種別群	tag 引数ではタグ名 Plib_custom_tag を与える必要があります。PDF へ書き込まれる実際のタグ名を tagname オプションで与える必要があります。カスタム種別群は rolemap 文書オプションを必要とします。

1. 要 PDF 2.0
2. 要 PDF 1.5 以上

- C の場合と、Perl・PHP・Ruby で ***stringformat=legacy*** の場合のためのエンコーディングオプション：***hypertextencoding***（表 2.1 参照）

戻り値　アイテムハンドル。以後のアイテム関連の呼び出しで使えます。

詳細　新規構造エレメントまたはページ装飾（まとめて**アイテム**といいます）を開始します。デフォルトでは、この新規エレメントは、文書構造ツリー内へ、カレントでアクティブなアイテムの子として挿入されます。しかし、*parent*・*index* オプションを用いて、構造ツリー内の別の場所を指定することもできます。構造エレメントは、任意のレベルにネストできます。標準エレメント種別（構造ツリー内にエントリーのない直接的エレメントではないということ）は、それが開かれたページに縛られず、任意の数のページへ続くこともできます。

構造エレメントと *Alt/ActualText* 属性は、PDFlib チュートリアル内の諸規則に従って適切にネストされる必要があります。いくつかの装飾的要素は自動的に**ページ装飾**としてタグ付けされます。詳しくは PDFlib チュートリアルを参照してください。

PDF/A　PDF/A-1a/2a/3a ではタグ付けは必須ですが、タグ用途やネスト化について具体的な必要条件はありません。PDF/UA-1 の必要条件に従うことを推奨します。

PDF/UA　すべての画像・グラフィック内容は *Artifact* か *Figure* としてタグ付けされる必要があります。
追加の規則群がさまざまな構造種別とオプションに適用されます（PDFlib チュートリアル参照）。

スコープ　**ページ**。グループ化エレメントの場合は**文書**も。対応する *PDF_end_item()* と必ずペアにして呼び出す必要があります。このメソッドはタグ付き PDF モードでのみ使えます。

表 14.2　PDF_begin_item( ) の構造種別の、およびさまざまなメソッドの tag オプションを用いた短縮タグ付けのための処理オプション。いくつかのオプションは PDF_begin_mc( )・PDF_mc_point( ) でも使用可能です。

オプション	説明
associatedfiles	（アセットハンドルのリスト。標準エレメントでのみ可。PDF 2.0・PDF/A-3 でのみ可）構造エレメントに紐付けたいファイル（複数可）のアセットハンドル（複数可）。このファイル群は、PDF_load_asset( ) で type=attachment を用いて読み込んである必要があります。
bookmark	（しおりハンドル。標準エレメントでのみ可）この構造エレメントと紐付けられるしおりに対するハンドル。
contents	（文字列。PDF_begin_mc( ) でのみ可）マーク付きコンテンツシーケンスを記述するコンテンツ文字列。
customtag	（論理値。rolemap 文書オプションを必要とします）true にすると、tagname オプションで与えられた種別名は、rolemap 文書オプションを通じて標準種別へマップされる必要のあるカスタム種別です。デフォルト：false このオプションを true に設定することは、引数 tagname=Plib_custom_tag を与えることと等価です。
direct	（論理値。tagname=Artifact・Code・BibEntry・Em・FENote・Note・Quote・Reference・Span・Strong・Sub でのみ可。BibEntry・TOCI・Note の子である Lbl では可、疑似アイテム ASpan・ReversedChars・Clip では可）true にすると、構造ツリー内のエントリーなしでマーク付きコンテンツシーケンス（「直接」エレメント）のみが出力されます。そうでない場合には、そのアイテムをマーク付きコンテンツシーケンスとして出力し、かつ構造ツリー内にエントリーを生成します（「標準」エレメント）。デフォルト：Title オプションが与えられているなら false、そうでないなら true
elementid	（文字列。標準エレメントでのみ可。PDF 2.0）構造エレメントに対する一意な文字列識別子。このエレメントを脚注または移動先として参照するために使用できます。id オプションと異なり、この識別子は内部的にのみ使用され、PDF 出力へ書き出されません。

表 14.2 PDF_begin_item( ) の構造種別の、およびさまざまなメソッドの tag オプションを用いた短縮タグ付けのための処理オプション。いくつかのオプションは PDF_begin_mc( )・PDF_mc_point( ) でも使用可能です。

オプション	説明
id	（文字列。疑似エレメントと、Note 以外のインラインエレメントでは不可。PDF/UA-1 では、Note としてタグ付けされている脚注・後注に対しては必須）このエレメントに、Headers オプションで使うための識別子を割り当てます。この文字列は、すべての構造エレメントの中で一意である必要があります。
index	（整数。擬似タグでは不可）エレメントを親の中に挿入したい位置の、ゼロから始まる番号。この位置から開始して、その親の既存の子孫群が上方へシフトされていきます。0 から、カレントの子の数までの値を与えることができます。値 -1 は、このエレメントを末尾に、すなわち新規の最近アイテムとして追加します。これは、カレントのエレメント数を index として与えることと等価です。デフォルト：-1
metadata	（オプションリスト。標準エレメントでのみ可）構造エレメントに対するメタデータ（307 ページ「15.2 XMP メタデータ」を参照）。
namespace	（文字列。PDF 2.0。PDF 1.7/PDF 2.0 標準タグセット外のタグでは必須）この種別を含んでいるタグセットを識別する名前空間。tagsets 文書オプション内で指定した名前空間群のうちのいずれかにする必要があります。
parent	（アイテムハンドル。擬似・インライン構造種別では不可）このエレメントの親。以前に PDF_begin_item( ) を、または PDF_get_option( ) の activeitemid キーワードを呼び出して返されたハンドルです。値 0 は、構造ツリーのルートを参照します。-1 は、カレントでアクティブなエレメントを参照します。いいかえれば parent=-1 は、カレントエレメントの子を開きます。デフォルト：-1
ref	（文字列のリスト。標準エレメントでのみ可。PDF 2.0）カレントの構造エレメントが参照する、脚注・後注・サイドバーなど 1 個ないし複数の構造エレメントに対する参照 ID（複数可）。参照されるエレメントは、すでに存在していてもよいですし、後で作成しても構いません。参照された ID が、文書の終了までにどの構造エレメントにも全く割り当てられなかった場合には、その参照は警告なく無視されます。
tag	（オプションリスト）カレントエレメントの子として挿入される追加の構造エレメント。表 14.2 に従ったすべてのオプションをサブオプションとして使えます。タグ群は任意のレベルにネストすることができます。
tagname	（名前文字列。PDF_add_table_cell( ) を除き、tag オプションを用いた短縮タグ付けのためにはこのオプションが必須です）表 14.1 に従った標準種別か疑似種別の名前、またはカスタムタグの名前。あるいはこのオプションの値は、メソッド引数 tagname を通じて与えることもできます。 カスタムタグ名を与えるには、オプション customtag=true か引数 tagname=Plib_custom_tag を与えます。この場合には、この tagname オプションはカスタムタグの任意の名前を内容とします。ただしこの名前に対して、標準種別へのマッピングが、PDF_begin_document( ) の rolemap オプションを用いて定義されている必要があります。カスタム種別名は、127 個の winansi キャラクターか、または最長 127 UTF-8 バイトへ展開される Unicode キャラクター列を上限とします。カスタム種別名は、予約済み接頭辞 Plib で始まってはいけません。
title	（ハイパーテキスト文字列。インライン・疑似構造種別では不可。PDF/UA-1 では見出しに対して推奨）構造エレメントのタイトル。このタイトルは、Acrobat で構造ツリーを表示または操作するために有用でしょう。
user-coordinates	（論理値）false にすると、BBox・Width・Height はデフォルト座標で表されていると見なされ、そうでない場合にはユーザー座標が用いられます。デフォルト：グローバル usercoordinates オプションの値

表 14.3 PDF_begin_item( ) の構造種別の、およびさまざまなメソッドの tag オプションを用いた短縮タグ付けのための属性オプション。いくつかのオプションは PDF_begin_mc( ) ・ PDF_mc_point( ) でも使用可能です。

オプション	説明
ActualText	（ハイパーテキスト文字列。PDF 1.5 における ASpan 以外の擬似タグでは不可。PDF 1.4 モードで使うときは、direct オプションを true に設定する必要があります）コンテンツアイテムとその子群に対する等価な代替テキスト。
Alt	（ハイパーテキスト文字列。PDF 1.5 における ASpan 以外の擬似タグでは不可。PDF 1.4 モードで使うときは、direct オプションを true に設定する必要があります）コンテンツアイテムとその子群に対する代替説明としての単語か句。図や画像など、テキストとして認識できないものに対して与える必要があります。
ARIARole	（文字列。PDF 2.0）構造エレメントの役割を ARIA 1.1 に従って指定します。WAI-ARIA に挙げられているキーワード（www.w3.org/TR/dpub-aria-1.0/#roles）のうちの 1 つを内容とする文字列を与える必要があります。
artifact-subtype	（キーワード。tagname=Artifact かつ artifacttype=Pagination か artifacttype=Inline の場合にのみ可。PDF 1.7）ページ装飾の下位種別：Header・Footer・Watermark。PDF 2.0 では以下のキーワードも使えます：Bates・LineNum・PageNum・Redaction
artifacttype	（キーワード。tagname=Artifact でのみ可）コンテンツアイテムのページ装飾種別を指定します： ***Background*** （PDF 1.7）ページか構造エレメントの長さいっぱい、および / または幅いっぱいにわたる画像か色付きアイテム ***Inline*** （PDF 2.0）文書の論理構造の中でコンテキストを有するページ装飾コンテンツ。たいていは下位種別 LineNum か Redaction のページ装飾。 ***Layout*** 脚注罫線や表の背景色といったタイポグラフィーまたはデザイン要素。 ***Page*** トンボやカラーバーといった印刷支援。 ***Pagination*** ランニングヘッダーやページ番号といった付随的なページ機能。
Attached	（キーワードのリスト。tagname=Artifact かつ artifacttype=Pagination か Background で、ページ全体の背景ページ装飾を持つ場合にのみ可）ページ装飾が論理的に添付される、ページの辺を、もしあれば指定します。このリストは、1 ～ 4 個のキーワード Top・Bottom・Left・Right を内容とします。Left と Right の両方、または Top と Bottom の両方を含む場合は、ページ装飾がそれぞれ幅いっぱいか高さいっぱいであることを示します。
BBox	（長方形。tagname=Artifact・Figure・Formula・Form・Table でのみ可。artifacttype=Background の場合には必須、それ以外の場合にはオプショナル）このエレメントの外接枠を、デフォルト座標で（usercoordinates=false の場合）、あるいはユーザー座標で（usercoordinates=true の場合）表したもの。PDFlib は、配置された画像・グラフィック・PDF ページ（tagname=Figure か Artifact を持つ）・フォームフィールド（tagname=Form か Artifact を持つ）と、表エンジンによって作成された表（tagname=Table か Artifact を持つ）に対して、自動的にこの BBox を作成します。
Checked	（キーワード。PDF 1.7。PrintFieldRole=rb か cb の場合にのみ可）ラジオボタンまたはチェックボックスフィールドの状態：on か off か neutral
ColSpan	（整数。tagname=TH・TD でのみ可）テーブルのセルがわたる列の数。デフォルト：1
Continued-From	（文字列。tagname=L でのみ可。continuedlist=false の場合には許されません。PDF 2.0）このリストへ続くリストの識別子。この識別子は、他の 1 つのリストに、id オプションを用いて割り当ててある必要があります。
Continued-List	（論理値。tagname=L でのみ可。continuedfrom を与えている場合には強制的に true になります。PDF 2.0）true ならば、このリストは直前のリストの続きです。直前のリストは、continuedfrom オプションで指定したものか、（このオプションを指定しなかった場合）構造ヒエラルキー内の同一レベルにある前のリストです。デフォルト：false

表 14.3 PDF_begin_item( ) の構造種別の、およびさまざまなメソッドの tag オプションを用いた短縮タグ付けのための属性オプション。いくつかのオプションは PDF_begin_mc( )・PDF_mc_point( ) でも使用可能です。

オプション	説明
Desc	（ハイパーテキスト文字列。PDF 1.7）フィールドの代替名（PDF_create_field( ) の tooltip オプションと同様）
E	（ハイパーテキスト文字列。ASpan 以外の擬似タグでは不可。構造タグに対しては要 PDF 1.5）コンテンツアイテムに対する略語の展開形。略語や頭字語を説明するためには与える必要があります。Acrobat の読み上げ機能は、展開テキストを、たとえ明示的な単語区切りがなくても、独立した単語として扱います。
Headers	（文字列のリスト。tagname=TH・TD でのみ可。PDF 1.5）このリストの各文字列は、このセルに紐付いたテーブルヘッダーセル（TH エレメント）の識別子です。この識別子（群）は、そのターゲットセルに、その id オプションを用いて割り当てられている必要があります。複数のヘッダーセルが参照される場合には、テーブル行ヘッダー群をテーブル列ヘッダー群より前に挙げるべきです。これらのグループ名では、ヘッダー群は、最も個別なものから最も汎用なものへと並べるべきです。
Height	（float。tagname=Figure・Form・Formula・Table・TD・TH でのみ可）このエレメントの高さを、デフォルト座標で（usercoordinates=false の場合）、あるいはユーザー座標で（usercoordinates=true の場合）表したもの。
Lang	（文字列。ASpan 以外の擬似タグでは不可）コンテンツアイテム（および紐付けられたオプション ActualText・Alt・E）の言語識別子を、表 3.1 の lang オプションで説明している形式で指定します。これを使うと、文書のデフォルトの言語を、個々のコンテンツアイテムについてオーバーライドすることが可能です。
List-Numbering	（キーワード。tagname=L でのみ可。PDF/UA-1 では必須。PDF/UA-1 では、LI のいずれの子もエレメントを内容としない箇条書きでは None とする必要があります）番号付き箇条書き内の Lbl エレメント群のコンテンツに対して用いられる付番系列、あるいは番号なし箇条書き内の各項目の頭に付く記号（デフォルト：None）： ***Circle*** 白丸ビュレット ***Decimal*** 10 進アラビア数字（1 ～ 9、10 ～ 99、…） ***Description*** （PDF 2.0）用語とその定義の箇条書き ***isc*** 黒丸ビュレット ***LowerAlpha*** 英小文字（a, b, c, ...） ***LowerRoman*** ローマ数字小文字（i, ii, iii, iv, ...） ***None*** 自動付番なし。Lbl エレメント群（もしあれば）が、付番なしで任意のテキストを内容とする。この場合には、この箇条書きのラベル群を表現するすべてのグラフィックを Artifact としてマークする必要があります。 ***Ordered*** （PDF 2.0）番号付け方式を指定しない番号付き箇条書き ***Square*** 黒四角ビュレット ***Unordered*** （PDF 2.0）ビュレットを指定しない番号なし箇条書き ***UpperAlpha*** 英大文字（A, B, C, ...） ***UpperRoman*** ローマ数字大文字（I, II, III, IV, ...）

表 14.3 PDF_begin_item( ) の構造種別の、およびさまざまなメソッドの tag オプションを用いた短縮タグ付けのための属性オプション。いくつかのオプションは PDF_begin_mc( )・PDF_mc_point( ) でも使用可能です。

オプション	説明
Placement	（キーワード。擬似エレメントとインラインエレメント種別 Span・Quote・Note・Reference・BibEntry・Code では不可）エレメントの、それを囲う参照領域に対する相対的な位置付け。これは、文書が再整形される際や、他の形式へ書き出される際に意味を持ちます。Figure・Formula・Form・Link・Annot エレメントに対しては、それがグループ化エレメントの子として作成されているならば、Placement=Block を推奨します（デフォルト：Inline）： ***Before*** 要素の割り当て四角形の「前」辺が、要素を囲う最も近い参照領域のそれと同じ位置になるよう配置します。 ***Block*** 要素を囲う参照領域または親 BLSE 内のブロック進行向き内にスタックします。 ***End*** 要素の割り当て四角形の「終」辺が、要素を囲う最も近い参照領域のそれと同じ位置になるよう配置します。 ***Inline*** 要素を囲う BLSE 内のインライン進行向き内にパックします。 ***Start*** 要素の割り当て四角形の「始」辺が、要素を囲う最も近い参照領域のそれと同じ位置になるよう配置します。
PrintField-Role	（キーワード。PDF 1.7）印刷される（非インタラクティブ）フォームフィールドの種別： ***rb*** ラジオボタン ***cb*** チェックボックス ***pb*** 押しボタン ***tv*** テキスト値フィールド。このフィールドのテキスト値は、Form 構造エレメントの内容であるべきです。 ***lb*** リストボックス 非インタラクティブフォームフィールドと関連するコンテンツは、Part 構造エレメントで囲う必要があります。
RowSpan	（整数。tagname=TH・TD でのみ可）テーブルのセルがわたるテーブル行の数。デフォルト：1
Scope	（キーワード。tagname=TH でのみ可。PDF 1.5。PDF/UA-1 では推奨）そのテーブルヘッダーセルが、それを含むテーブル行内の他のセル群に対するものなのか、それともそれを含むテーブル列内のそれなのか、あるいはそれを含むテーブル行とテーブル列の両方に対するものなのかを、Row・Column・Both のいずれかのキーワードで指定します。
Short	（文字列。tagname=TH でのみ可。PDF 2.0）テーブルヘッダーセルの内容の短い形式
Summary	（ハイパーテキスト文字列。tagname=Table でのみ可。PDF 1.7）このテーブルの目的と構造の概要
Width	（float。tagname=Figure・Form・Formula・Table・TD・TH でのみ可）このエレメントの幅を、デフォルト座標で（usercoordinates=false の場合）、あるいはユーザー座標で（usercoordinates=true の場合）表したもの。
Writing-Mode	（キーワード）ILSE を並べていき BLSE を積んでいくレイアウト進行方向（デフォルト：LrTb）： ***LrTb*** 西洋筆記モード：左から右へ、上から下へ ***RlTb*** セム語筆記モード：右から左へ、上から下へ ***TbRl*** 日本語・中国語筆記モード：上から下へ、右から左へ ***TbLr・LrBt・RlBt・BtRl・BtLr*** （PDF 2.0）

C++ Java C# ***void end_item(int id)***

Perl PHP ***end_item(int id)***

C ***void PDF_end_item(PDF *p, int id)***

構造エレメントやその他の内容アイテムを閉じます。

id アイテムのハンドル。*PDF_begin_item()* で取得しておく必要があります。

詳細 直接アイテム群を、ページを終える前に閉じる必要があります。標準アイテムは、文書を終える前に閉じる必要があります。ただし、標準アイテムはすべて、完了次第閉じることを強く推奨します。アイテムは、そのすべての子を閉じているときにのみ、閉じることができます。アイテムを閉じた後は、その親がアクティブなアイテムになります。

スコープ **ページ**。グループ化アイテムの場合は**文書**も。対応する *PDF_begin_item()* と必ずペアにして呼び出す必要があります。このメソッドはタグ付き PDF モードでのみ許されます。

C++ Java C# ***void activate_item(int id)***

Perl PHP ***activate_item(int id)***

C ***void PDF_activate_item(PDF *p, int id)***

以前に作成された構造エレメントやその他の内容アイテムをアクティブ化します。

id 標準アイテムのハンドル。*PDF_begin_item()*を用いて取得したものである必要があり、かつ、まだ閉じていてはいけません。

詳細 構造エレメントを一時停止し、のちにアクティブ化すると、より柔軟にタグ付き PDF のページを作成できます。多段組みレイアウトや、本文に割り込むテキストなど、ページ上で複数の構造分岐が並行している時に有用です。

parent・index タグ付けオプション（表 14.2 参照）を使うと、構造ツリー内の特定の位置に構造エレメントを挿入できるわけですが、*PDF_activate_item()* を使って、以前に作成した構造エレメントに内容を追加することもできるのです。

Acrobat での諸問題を回避するために、*PDF_activate_item()* を呼び出した直後に追加するのは内容アイテムではなく他の構造エレメントだけにするべきです。

スコープ **文書・ページ**。このメソッドはタグ付き PDF モードでのみ許されます。

14.2 短縮タグ付け

構造エレメントとページ装飾は、*PDF_begin/end_item()* ペアを用いて作成することができます。あるいはそのかわりに、以下のメソッドの *tag* オプション（表 14.4 参照）を用いて短縮タグ付けを利用することも可能です：

- *PDF_add_table_cell()* と、*PDF_fit_*()* の対応するオプション：*fitgraphics*・*fitimage*・*fitpath*・*fitpdipage*・*fittextline*・*fittextflow*・*fitannotation*・*fitfield*
- *PDF_begin_document()*：短縮タグ付けを用いて、その構造ヒエラルキーのルートエレメントを作成できます
- *PDF_create_annotation()*
- *PDF_create_field()*
- *PDF_draw_path()*
- *PDF_fit_graphics()*
- *PDF_fit_image()*
- *PDF_fit_pdi_page()*・*PDF_info_pdi_page()*
- *PDF_fit_table()*：短縮タグ付けが自動テーブルタグ付けを引き起こします
- *PDF_fit_textflow()*
- *PDF_fit_textline()*
- さまざまなメソッドの *matchbox* オプション

PDF_begin_document() の場合を除き、これらのメソッドは、ページコンテンツアイテム群を生成しますので、グループ化エレメント群を生成するために使用することはできません。ただし、そのグループ化エレメントが、ネストされた *tag* オプションを通じて、別のエレメントを伴っている場合を除きます。*PDF_begin_document()* と *PDF_add_table_cell()* の場合を除き、短縮タグ付けは**ページ**スコープ内でのみ利用可能です。短縮タグ付けに関する詳しい説明は PDFlib チュートリアルにあります。

表 14.4 PDF_add_table_cell( ) と、PDF_add_table_cell( )・PDF_create_annotation( )・PDF_create_field( )・PDF_draw_path( )・PDF_fit_graphics( )・PDF_fit_image( )・PDF_fit_pdi_page( )・PDF_fit_table( )・PDF_fit_textflow( )・PDF_fit_textline( ) の対応する fit* オプションと、さまざまなメソッドの matchbox オプションの、短縮タグ付けのためのオプション

オプション	説明
tag	（オプションリスト）その配置されるコンテンツに対する構造エレメントか Artifact を作成します。表 14.2 に挙げているサブオプション群を使えます。この tag オプションはサブオプションとしても許容されますので、ただ 1 回の呼び出しで、ネストされたタグ構造を生成することもできます。

14.3 マーク付きコンテンツ

マーク付きコンテンツメソッドを使うと、コンテンツにプロパティでマークできます。これにはタグ付き PDF モードは必要ではありません。*PDF_begin_item()* は必要に応じて内部的にマーク付きコンテンツを生成します。

C++ Java C# ***void begin_mc(String tagname, String optlist)***

Perl PHP ***begin_mc(string tagname, string optlist)***

C ***void PDF_begin_mc(PDF *p, const char *tagname, const char *optlist)***

マーク付きコンテンツシーケンスを開始させます。プロパティ群を与えることもできます。

tagname マーク付きコンテンツシーケンスの名前。以下のタグが使えます：

- 表 14.2 のすべての標準・擬似アイテム群。
- ユーザー定義プロパティを持つカスタム項目に対してはタグ名 *Plib_custom* を用いることができます。
- タグ名 *Plib* は予約済みです。

optlist 以下のマーク付きコンテンツシーケンスオプションが使えます：

- マーク付きコンテンツシーケンスの標準プロパティ群のためのオプション。表 14.2 のタグ付けオプション群の、以下の部分集合が使えます：
  ActualText・Alt・artifactsubtype・artifacttype・contents・customtag・E・Lang・ListNumbering・Placement・tagname
- タグ *Plib_custom・Plib* では表 14.5 の *properties* オプションも使えます。
- C の場合と、Perl・PHP・Ruby で *stringformat=legacy* の場合のエンコーディングオプション：*hypertextencoding*（表 2.1 参照）

詳細 指定したタグとプロパティ群を持ったマーク付きコンテンツシーケンスが開始されます。オプションを何も与えなければ、プロパティを何も持たないシーケンスが作成されます。マーク付きコンテンツシーケンスは任意の階層にネスト可能です。*PDF_begin/end_item()・PDF_begin/end_mc()* のネストを適切に行うのはユーザー側の役割です。

スコープ **ページ・パターン・テンプレート・グリフ**。対応する *PDF_end_mc()* と必ずペアにして、同じスコープ内で呼び出す必要があります。

表 14.5 PDF_begin_mc( )・PDF_mc_point( ) によるタグのユーザー定義プロパティのためのオプション

オプション	説明
properties	（オプションリストのリスト。tagname=Plib・tagname=Plib_custom でのみ可）各リストは、ユーザー定義プロパティを指定した 3 個のオプションを内容として持ちます： ***key*** （文字列。必須）プロパティの名前。 ***type*** （キーワード。必須）プロパティ値の型：boolean・name・string のいずれか。 ***value*** （type=string ならハイパーテキスト文字列、そうでないなら文字列。必須）プロパティの値。

C++ Java C# ***void end_mc( )***

Perl PHP ***end_mc( )***

C ***void PDF_end_mc(PDF *p)***

もっとも最近に開かれたマーク付きコンテンツシーケンスを終了させます。

詳細 マーク付きコンテンツシーケンスはすべて、*PDF_end_page_ext( )* を呼び出す前に閉じる必要があります。

スコープ **ページ・パターン・テンプレート・グリフ**。対応する *PDF_begin_mc( )* と必ずペアにして、同じスコープ内で呼び出す必要があります。

C++ Java C# ***void mc_point(String tagname, String optlist)***

Perl PHP ***mc_point(string tagname, string optlist)***

C ***void PDF_mc_point(PDF *p, const char *tagname, const char *optlist)***

マーク付きコンテンツポイントを追加します。プロパティ群を与えることもできます。

tagname マーク付きコンテンツポイントの名前。以下のタグが使えます：

- 表 14.2 のすべての標準・擬似アイテム群。
- カスタム項目に対してはタグ名 *Plib_custom* を用いることができます。
- タグ名 *Plib* は予約済みです。

optlist 以下のオプションが使えます：

- 表 14.5 に従ったマーク付きコンテンツポイントの標準プロパティオプション群。
- タグ *Plib_custom*・*Plib* では表 14.5 のオプションも使えます。

詳細 指定したタグ名とプロパティ群を持ったマーク付きコンテンツポイントが作成されます。オプションを何も与えなければ、プロパティを何も持たないマーク付きコンテンツポイントが作成されます。

スコープ **ページ・パターン・テンプレート・グリフ**

15 文書交換

この章の API メソッド：

- *PDF_set_info()*
- *PDF_add_portfolio_folder()*
- *PDF_add_portfolio_file()*
- *PDF_begin_dpart()*
- *PDF_end_dpart()*

15.1 文書情報フィールド

C++ Java C# ***void set_info(String key, String value)***

Perl PHP ***set_info(string key, string value)***

C ***void PDF_set_info(PDF *p, const char *key, const char *value)***

C ***void PDF_set_info2(PDF *p, const char *key, const char *value, int len)***

文書情報フィールド ***key*** に ***value*** を書き込みます。

key （名前文字列）文書情報フィールドの名前。標準名のいずれか（表 15.1 参照）、または任意のカスタム名を用いることができます。カスタムフィールドの数に制限はありません。カスタム文書情報フィールドの利用と意味づけに関しては、Dublin Core Metadata 要素セットを推奨します。[1]

value （ハイパーテキスト文字列）***key*** 引数に設定したい文字列。Acrobat では、***value*** に最大長 255 バイトという制約が課されています。

len （C 言語バインディングのみ）***value*** の長さをバイト単位で。***len=0*** にすると null 終了文字列を与える必要があります。

詳細 与えたキー / 値のペアが文書情報辞書に書き込まれます。*PDF_begin/end_document()* の ***autoxmp*** オプションが ***true*** であれば、PDFlib は、表 15.1 に従った標準情報キー群を、対応する XMP メタデータプロパティに同期させます。PDF/A モードでない限り、カスタム情報キー群は XMP 内の ***pdfx*** スキーマ（「PDF 拡張スキーマ」の意であり、PDF/X 規格とは無関係）へ同期されます。この方式を使うと、シンプルな XMP プロパティ群を、文書 ***metadata*** オプションを通じて XMP を与えることなく作成できます。とはいえ、XMP はシンプルなキー / 値のペアよりもはるかに強力なものです。

文書情報フィールド群は、*PDF_begin/end_document()* の ***metadata*** オプションに与えられる XMP 文書メタデータ内の対応するプロパティ群をオーバーライドします。

PDF 2.0 では、文書情報辞書は廃止されており、ユーザーが与えたフィールド群は、*PDF_begin/end_document()* で ***emitdocinfo*** オプションが `true` の場合にのみ出力されます。文書情報辞書内のユーザーが与えたフィールド群が出力されない場合であっても、PDFlib は文書情報項目群を XMP へ同期します。

PDF/A カスタム文書情報フィールド群は XMP へ同期されません。なぜならカスタムメタデータプロパティ群は PDF/A では拡張スキーマ記述を必要とするからです（PDFlib チュートリアル参照）。

1. dublincore.org 参照

PDF/UA *key=Title* を持つ情報フィールドには、空でない値を与える必要があります。あるいは、*PDF_begin_document()*の*metadata*オプションで*dc:title* XMPプロパティを与えることもできます。

PDF/X *key=Title* と *key=Creator* を持つ情報フィールドには、空でない値を与える必要があります。あるいは、PDF/X-4・PDF/X-5 では、*PDF_begin_document()* の *metadata* オプションで *dc:title*・*xmp:CreatorTool* XMP プロパティを与えることもできます。

Trapped 情報フィールドでは値 *True* と *False* のみが許容されます。

スコープ 任意。**オブジェクト**スコープで使うと、与える値は次の文書にのみ使われます。

表 15.1 文書情報フィールドのキーと、対応する XMP プロパティ

キー	説明	対応する XMP プロパティ
Author	文書を作成した人の名前	dc:creator[0]
Creator	文書の作成に使用されたソフトウェア（つねに PDFlib である、PDF 出力の Producer とは別です）。	xmp:CreatorTool
Keywords	文書の内容を記述したキーワード	pdf:Keywords
Subject	文書のサブタイトル	dc:description["x-default"]
Title	文書のタイトル	dc:title["x-default"]
Trapped	文書にトラッピングが適用されているかどうかを表します。とりうる値は True・False・Unknown。	pdf:Trapped
その他任意の名前	ユーザー定義の文書情報フィールド。1 つのカスタムフィールド名は一度しか与えてはいけません。 標準識別に用いられるフィールドは許容されません（GTS_PDFXVersion など）。	pdfx:<key>

15.2 XMP メタデータ

XMP メタデータは、文書全体か、個別のページ・フォント・ICC プロファイル・画像・テンプレート・取り込み PDF ページに対して与えることができます。PDFlib は、PDF 2.0 出力の場合にはつねに XMP を生成します。さまざまなメソッドの *metadata* オプションのサブオプションを表 15.2 に示します。

表 15.2 PDF_begin/end_document( )・PDF_begin/end_page_ext( )・PDF_load_font( )・PDF_PDF_load_image( )・PDF_begin_template_ext( )・PDF_open_pdi_page( ) の metadata オプションと PDF_load_graphics( )・PDF_begin_item( ) の templateoptions オプションとさまざまなメソッドの tag オプションのサブオプション

オプション	説明
associatedfiles	（アセットハンドルのリスト。PDF 2.0）メタデータに紐付けるファイル（複数可）のアセットハンドル（複数可）。このファイル（複数可）は、PDF_load_asset( ) で type=attachment を用いて読み込んである必要があります。
compress	（論理値。PDF_begin/end_document( ) では不可）PDF 出力内の XMP メタデータストリームを圧縮します。このオプションを PDF_begin_page_ext( ) でのみ与え、PDF_end_page_ext( ) で与えないと、その値はデフォルトよりも優先されます。デフォルト：false PDF/A-1・PDF/X：compress=true は許されません。
inputencoding	（キーワード）filename で与えたメタデータを解釈するためのエンコーディング。デフォルト：unicode
inputformat	（キーワード）filename で与えたメタデータの形式。デフォルト：utf8、ただし inputencoding が 8 ビットエンコーディングのときは bytes
keepxmp	（論理値。PDF_load_image( )・PDF_load_graphics( ) でのみ可。filename との併用不可）画像またはグラフィックファイル内に存在する XMP メタデータが保持されます。すなわち、PDF 文書内の生成画像に付けられます。XMP メタデータは、TIFF・JPEG・JPEG 2000 画像形式内と SVG グラフィック内で効力を持ちます。画像またはグラフィックファイル内に XMP メタデータが見つからないときは、このオプションは何の効力も持ちません。デフォルト：false
filename	（名前文字列。keepxmp を与えない場合には必須）整形式 XMP メタデータを内容として持つファイルの名前。これは filenamehandling グローバルオプションに従って解釈されます（表 2.1 参照）。
strict	（論理値）false にすると、ある種の XMP 違反はエラーとして報告されず黙って修正されます。true にすると、そのような違反は例外を発生させます。デフォルト：false

PDF/A PDF/A のための XMP 識別プロパティ群は自動的に作成されます。

PDFlib は、ユーザーが与えた XMP ストリームの中の関連項目群を、標準文書情報フィールド群へ同期させます（*autoxmp* モードが文書情報フィールドを XMP へ同期させるのと同様です）。ただし、PDFlib はそれ以外の XMP 項目群をカスタム文書情報フィールド群へ同期させることはしません。XMP 文書メタデータに対するさらなる PDF/A 必要条件群を PDFlib チュートリアルで説明しています。以下の検証が XMP メタデータに対して行われます：

- PDF/A-1：文書レベル XMP が、XMP 2004 に準拠するか、あるいは拡張スキーマ記述を含む必要があります。文書レベル XMP のためのスキーマ記述は、*PDF_begin_document( )* か *PDF_end_document( )* で与えることができます。
- PDF/A-2/3：文書レベルおよびコンポーネントレベル（ページ等）の XMP が、XMP 2005 に準拠するか、あるいは拡張スキーマ記述を含む必要があります。コンポーネントレベル XMP のためのスキーマ記述は、それぞれのコンポーネントレベル XMP を用いて

(*PDF_begin_page_ext()* 内などで)、あるいは、*PDF_begin_document()* の中で文書レベル XMP を用いて与えることができます。

PDF/UA PDF/UA のための XMP 識別プロパティ群は自動的に作成されます。

PDF/VT PDF/VT のための XMP 識別プロパティ群は自動的に作成されます。

PDF/X PDF/X-4/5 のための XMP 識別プロパティ群は自動的に作成されます。

15.3 PDF パッケージとポートフォリオ

ポートフォリオの諸機能は、以下のメソッドとオプションを用いて実装されています：

- ポートフォリオを作成するには、*PDF_end_document()* の *portfolio* オプションを用います。このメソッドについては 43 ページ「3.1 文書関数」で説明しており、この *portfolio* オプションについては表 15.5 で説明します。
- ファイルとフォルダーをポートフォリオに追加するには、*PDF_add_portfolio_folder()* と *PDF_add_portfolio_file()* を用います。これらのメソッドについては後述します。
- ポートフォリオ内をナビゲートするためのアクションを作成するには、*PDF_create_action()* と *type=GoToE* を用います（265 ページ「12.4 アクション」を参照）。

C++ Java C# ***int add_portfolio_folder(int parent, String, foldername, String optlist)***

Perl PHP ***int add_portfolio_folder(int parent, string foldername, string optlist)***

C ***int PDF_add_portfolio_folder(PDF *p, int parent, const char *foldername, int len, const char *optlist)***

新規または既存のポートフォリオにフォルダーを追加します。

parent 親フォルダーを、これ以前に *PDF_add_portfolio_folder()* を呼び出して返されたフォントハンドルで指定するか、またはルートフォルダーの場合は -1（PHP では 0）。

foldername （ハイパーテキスト文字列で 1 ～ 255 キャラクター。キャラクター / ¥ : * ″ < > | は使ってはいけません。最後のキャラクターはピリオド「.」にしてはいけません）フォルダーの名前。同じ親を持つ 2 個のフォルダーが、大文字・小文字を無視したときに同じ名前を持ってはいけません。ルートフォルダーの名前は Acrobat では無視されます。

len （C 言語バインディング）*foldername* の長さ（バイト単位で）。*len=0* とすると、null 終端文字列を与える必要があります。

optlist ポートフォリオのプロパティ群を指定したオプションリスト。以下のオプションが使えます：

- 一般オプション群：*errorpolicy*（表 1.5 参照）
- 表 13.6 に従った、フォルダープロパティ群のためのオプション：***description*・*thumbnail***
- 表 15.3 に従ったメタデータオプション：*fieldlist*
- C の場合と、Perl・PHP・Ruby で *stringformat=legacy* の場合のためのエンコーディングオプション：*hypertextencoding*・*hypertextformat*（表 2.3 参照）

戻り値 *PDF_add_portfolio_folder()* または *PDF_add_portfolio_file()* で使えるハンドル。

詳細 生成されたフォルダー構造は、カレント文書の PDF ポートフォリオを作成するために使われます。このフォルダー構造は *PDF_end_document()* の後に削除されます。このメソッドは、*PDF_begin_document()* に *attachments* オプションを与えているときは使ってはいけません。

スコープ **オブジェクト**以外任意。要 PDF 1.7ext3

表 15.3 PDF_add_portfolio_folder( )・PDF_add_portfolio_file( ) のオプション

オプション	説明
fieldlist	（オプションリストのリスト）ファイルかフォルダーのメタデータフィールド群を指定します。各リストは、PDF_end_document( ) の portfolio オプションの schema サブオプションの中のフィールドを指し示します。使えるサブオプションを表 15.4 に挙げます。

C++ Java C# **int add_portfolio_file(int folder, String filename, String optlist)**

Perl PHP **int add_portfolio_file(int folder, string filename, string optlist)**

C **int PDF_add_portfolio_file(PDF *p, int folder, const char *filename, int len, const char *optlist)**

ポートフォリオのフォルダーまたはパッケージにファイルを追加します。

folder これ以前に *PDF_add_portfolio_folder()* を呼び出して返されたフォントハンドルか、またはルートフォルダーの場合は -1（PHP では 0）。ルートフォルダーでないフォルダーは要 PDF 1.7ext3 です。

filename （名前文字列。*filenamehandling* グローバルオプションに従って解釈されます。表 2.3 参照）PDF ポートフォリオの指定したフォルダーに付けたいディスク上のファイルか仮想ファイルの名前。*PDF_begin_document()* の *createpvf* オプションを使えば、一時ファイルをディスク上に一切作らずに、文書をメモリー内で作成して、それを受け渡して PDF ポートフォリオの中に入れることができます。

Acrobat は、Acrobat 内からファイルを開く際に、どのアプリケーションを起動するかを、そのファイル名の拡張子を用いて決めることに留意してください。適切な拡張子を持ったファイル名を、外部的な制約のために使うことができないときは、かわりに PVF ファイル（任意のファイル名が使えます）を作成することもできます。

len （C 言語バインディング）*filename* の長さ（バイト単位で）。*len=0* とすると、null 終端文字列を与える必要があります。

optlist フォントのプロパティ群を指定したオプションリスト：

- 表 13.6 に従った、ファイルプロパティ群のためのオプション：
  description・filename・mimetype・name・password・relationship・thumbnail
- 表 15.3 に従ったメタデータオプション：*fieldlist*
- C の場合と、Perl・PHP・Ruby で *stringformat=legacy* の場合のためのエンコーディングオプション：*hypertextencoding*（表 2.3 参照）

戻り値 ファイルの追加が成功したときは値 1、メソッド呼び出しが失敗したときはエラーコード -1（PHP では 0）。*errorpolicy=exception* とすると、このメソッドはエラー時に例外を発生させます。PDF 文書群は、変更日と作成日を取得するために開かれます。PDF 文書を開くことができなかったとき（パスワードを与えなかった場合等）も、その文書は PDF ポートフォリオの中へ取り込まれます。

詳細 指定したファイルが、PDF 1.7ext3 のポートフォリオの指定したフォルダーに、または PDF 1.7 のパッケージに付けられます。PDI が利用できる場合は、PDF 文書群は可能ならば開かれ、その作成日と変更日がポートフォリオに書き込まれます。このメソッドは、*PDF_begin_document()* に *attachments* オプションを与えているときは使ってはいけません。

PDF/A PDF/A-1：このメソッドを呼び出してはいけません。
PDF/A-2：*filename* は、PDF/A-1 または PDF/A-2 文書を参照している必要があります。いくつかのオプションは制約されます。表 13.6 を参照してください。
PDF/A-3：任意のファイル種別を追加できます。*relationship* オプションが必須です。パッケージに追加されたファイルは、暗黙に文書全体に関連付けられます。

スコープ **オブジェクト**以外任意。要 PDF 1.7

表 15.4 PDF_add_portfolio_folder( ) と PDF_add_portfolio_file( ) の fieldlist オプションのサブオプション

オプション	説明
key	（文字列。必須）フィールドの名前。これは、PDF_end_document( ) の portfolio オプションの schema サブオプションの中のキーを指し示す必要があります。名前は一意である必要があります。
prefix	（ハイパーテキスト文字列）ユーザーに見せるフィールド値の頭に付ける接頭辞文字列。Acrobat はこの項目を type=text の場合にのみ用います。デフォルト：なし
type	（キーワード）フィールドのデータ種別。使えるキーワード（デフォルト：text）： ***text*** テキストフィールド：フィールド値はハイパーテキスト文字列として格納されます。 ***date*** 日付フィールド：フィールド値は PDF 日付文字列として格納されます。 ***number*** 数値フィールド：フィールド値は PDF 数値として格納されます。
value	（ハイパーテキスト文字列。必須）PDF_end_document( ) の portfolio オプションの schema サブオプションの中のフィールドの値。そのデータ種別を type オプションで指定する必要があり、かつ、これを portfolio オプションの schema サブオプションの対応する type サブオプションに一致させる必要があります。

表 15.5 PDF_end_document( ) の portfolio オプションのサブオプション

オプション	説明
coversheet	（ハイパーテキスト文字列）ユーザーインターフェイスで最初に見せるポートフォリオ内のファイルの名前。デフォルト：ポートフォリオを含む文書
coversheet-folder	（フォルダーハンドル）coversheet オプションで指定したファイルが入っているポートフォリオ内フォルダーの名前。もしもその coversheet で指定した名前を持つファイルが複数のポートフォリオフォルダー内に存在していて coversheetfolder を指定していない場合には、最初に出現したものが使われます。デフォルト：なし
initialview	（キーワード）初期表示。使えるキーワード（デフォルト：detail）： ***detail*** ポートフォリオを詳細モードで表示します。すなわち、schema オプション内のすべての情報を多段組の形で表示します。このモードがユーザーに最も多くの情報を提供します（Acrobat：「上に表示」）。 ***hidden*** ポートフォリオを初め非表示にします。ユーザーが明示的に操作すればファイル一覧を得ることはできます（Acrobat：「最小化表示」）。 ***tile*** ポートフォリオをタイルモードで表示します。すなわち、コレクション内の各ファイルを小さなアイコンで表し、schema オプション内の情報の一部分を表示します。このモードはユーザーにファイル添付に関するトップレベルな情報を提供します（Acrobat：「左に表示」）。

表 15.5 PDF_end_document( ) の portfolio オプションのサブオプション

オプション	説明
schema	（オプションリストのリスト）ポートフォリオのメタデータスキーマ：各オプションリストは、フォルダーまたはファイルの fieldlist 内のキーに、または標準フィールドの名前に対応する一意な名前を持つフィールドを定義します。これらのフィールドは、Acrobat でのポートフォリオの表示動作を定義します（デフォルト：Acrobat は、ファイル名・サイズ・変更日・説明（あれば）を表示します）： ***editable*** （論理値）Acrobat がフィールド値の編集を許すべきかどうかを指定します。デフォルト：false ***key*** （文字列。必須）初期フィールド名。これは一意である必要があります。以下の名前（ユーザー定義フィールドには使えません）を用いると、内蔵フィールドに新たなラベルを割り当てることができます：_creationdate・_description・_filename・_moddate・_size。 ***label*** （ハイパーテキスト文字列。必須）ユーザーに見せるフィールドラベルのテキスト。 ***order*** （整数）ユーザーインターフェイスでのフィールドの相対順序（1,2,3,...） ***type*** （キーワード）フィールドのデータ種別。以下の種別を用いて、fieldlist オプション内のユーザー定義フィールドを指し示すことができます（デフォルト：text）： ***text*** ハイパーテキスト文字列 ***date*** PDF 日付文字列 ***number*** 数値 ***visible*** （論理値）ユーザーインターフェイスでのフィールドの初期の表示・非表示。デフォルト：true。ただし、ユーザー定義フィールドがあるときは、Acrobat は、内蔵フィールドを、それらを表示すると明示的に指定されていない限り、非表示にします。
sort	（オプションリストのリスト。各リストは文字列 1 個と、オプショナルなキーワード 1 個を内容として持ちます）schema オプションで指定したフィールド群をユーザーインターフェイスで並べ替える順序。各サブリストは、フィールド名（必須）とキーワード 1 個（オプショナル）を内容として持ちます。使えるキーワード（デフォルト：ascending）： ***ascending*** フィールド値を昇順で並べ替えます。 ***descending*** フィールド値を降順で並べ替えます。 Acrobat はこのリストを用いて、ポートフォリオ内のリストを並べ替えます。このリストは、追加のフィールド群が並べ替えに寄与できるようにするために用いられ、追加の各フィールドを用いて順序の決定が推進されます：schema オプション内の複数のフィールドがリスト内の 1 番目のフィールドで同じ値を持つときは、リスト内の後続のフィールド群が並べ替えに用いられ、それは順序が一意に定まるか、あるいはフィールド名を使い果たすまで続けられます。デフォルト：並べ替えしない
split	（オプションリスト。PDF 1.7ext3）分割バーの向きと位置。デフォルトは initialview オプションに依存します：値 detail（または値なし）は横向きを暗示し、tile は縦向きを意味します。initalview=hidden にすると分割バーは用いられません： ***direction*** （キーワード）分割バーの向き。使えるキーワード： ***horizontal*** ウインドウを横に分割します。 ***vertical*** ウインドウを縦に分割します。 ***none*** ウインドウを分割しません。ウインドウ全体がファイルナビゲーション表示に用いられます。 ***position*** （パーセント値。direction=none の場合には無視されます）分割バーの初期位置を、得られるウインドウ領域に対するパーセント値として指定します。使える値は範囲 0 ～ 100 です。デフォルト：ビューアー依存

15.4 文書部分ヒエラルキー

C++ Java C# ***void begin_dpart(String optlist)***

Perl PHP ***begin_dpart(string optlist)***

C ***void PDF_begin_dpart(PDF *p, const char *optlist)***

文書部分ヒエラルキー内に新規ノードを開始します(PDF/VTかPDF 2.0を必要とします)。

optlist 表 15.6 に従った文書部分ヒエラルキーオプションを指定したオプションリスト：*associatedfiles*・*dpm*

PDF/VT PDF/VT を生成する際には、このメソッドと、その相手である *PDF_end_dpart()* を、少なくとも 1 回呼び出す必要があります。*PDF_begin_dpart()* への初めての呼び出しは、暗黙的に、文書部分（DPart）ヒエラルキーのルートノードを作成します。最上位レベルで *PDF_begin_dpart()* を複数回呼び出すことはエラーです。

PDF_begin_dpart() を呼び出した後に *PDF_begin_page_ext()* を呼び出すことは、文書部分のページ範囲の開始を定義します。次に *PDF_begin_dpart()* を呼び出すまでの間のすべての後続するページ群は、その同一の文書部分に属します。すべての呼び出しはともに文書部分ヒエラルキーを木構造として作成し、この文書部分ヒエラルキーは 2 種類のノードを内容とすることができます：

- 内部ノードは、他の 1 個ないし複数のノードを子孫として持ちます。この子孫は、内部ノードか葉ノードのいずれかであることができますが、ある 1 つの内部ノードが両方の種類の子孫を内容とすることはできません。
- 葉ノードは、ある範囲内のページ（群）を記述します。葉ノードは子孫ノードを決して持ちません。

PDF_begin_dpart() と *PDF_end_dpart()* への呼び出しは、*PDF_end_document()* が呼び出された際にマッチしている必要があります。文書に対して文書部分ヒエラルキーを作成しようとするときは、このメソッドを、*PDF_begin_page_ext()* を初めて呼び出すよりも前に少なくとも 1 回は呼び出しておく必要があります。*PDF_begin_dpart()* への複数回の呼び出しは、文書部分ヒエラルキーの、より深いレベル群を作成します。文書部分ヒエラルキーのこの生成される深さ（すなわち、*PDF_begin_dpart()* の最大ネスト化レベル）は、***nodenamelist*** 文書オプションを用いて指定されたリストの長さとマッチする必要があります。

スコープ 文書。このメソッドは、マッチする *PDF_end_dpart()* への呼び出しと必ずペアにする必要があります。

表 15.6 PDF_begin/end_dpart( ) のオプション

オプション	説明
associatedfiles	（アセットハンドルのリスト。PDF 2.0 でのみ可）DPart ノードに紐付けるファイル群のアセットハンドル群。このファイル群は、PDF_load_asset( ) と type=attachment を用いて読み込まれている必要があります。
dpm	（POCA コンテナハンドル。PDF_begin_dpart( ) か PDF_end_dpart( ) に与えることができますが、同一文書部分について両方のメソッドに与えることはできません）PDF_poca_new( ) を用いて作成された辞書コンテナに対するハンドル。この辞書コンテナは、その新規ノードに対する文書部分メタデータを内容としています。この辞書は、オプション usage=dpm を用いて作成されている必要があります。

C++ Java C# **void end_dpart(String optlist)**

Perl PHP **end_dpart(string optlist)**

C **void PDF_end_dpart(PDF *p, const char *optlist)**

文書部分ヒエラルキー内でノードを閉じます（PDF/VT か PDF 2.0 を必要とします）。

optlist 表 15.6 に従って文書部分ヒエラルキーオプションを指定したオプションリスト：*associatedfiles*・*dpm*

PDF/VT *PDF_end_page_ext()* の後の初めての *PDF_end_dpart()* への呼び出しは、暗黙的に、文書部分ヒエラルキーの葉に属するページ範囲の終了を定義します。*PDF_begin_dpart()* と *PDF_end_dpart()* への呼び出しは、*PDF_end_document()* が呼び出された際にマッチしている必要があります。

スコープ 文書。このメソッドは、マッチする *PDF_begin_dpart* への呼び出しと必ずペアにする必要があります。

A API メソッド一覧

この付章ではすべての API メソッドを挙げます。メソッド名をクリックするとその説明へ飛べます。

一般

get_errnum
get_errmsg
get_apiname
get_opaque（C/C++のみ）
convert_to_unicode
set_option
get_option
get_string
new（Cのみ）
delete
create_pvf
delete_pvf
info_pvf
download
poca_new
poca_delete
poca_insert
poca_remove

文書とページ

begin_document
begin_document_callback（C/C++のみ）
end_document
get_buffer
begin_dpart
end_dpart
begin_page_ext
end_page_ext
suspend_page
resume_page
define_layer
set_layer_dependency
begin_layer
end_layer

フォント

load_font
close_font
setfont
info_font
begin_font
end_font
begin_glyph_ext
end_glyph
encoding_set_char

単純テキスト出力

set_text_pos
show
show_xy
continue_text
stringwidth

テキスト組版

set_text_option
fit_textline
info_textline
add_textflow
create_textflow
fit_textflow
info_textflow
delete_textflow

表組版

add_table_cell
fit_table
info_table
delete_table

範囲枠

info_matchbox

色

setcolor
load_iccprofile
makespotcolor
create_devicen
shading
shading_pattern
shfill
begin_pattern_ext
end_pattern

画像・テンプレート

load_image
close_image
fit_image
info_image
begin_template_ext
end_template_ext

SVG グラフィック

load_graphics
close_graphics
fit_graphics
info_graphics

グラフィックステート

set_graphics_option
setlinewidth
save
restore
create_gstate
set_gstate

座標変換

translate
scale
rotate
align
skew
concat
setmatrix

パス構築

moveto
lineto
curveto
circle
arc
arcn
circular_arc
ellipse
elliptical_arc
rect
closepath

パス描画・クリッピング

stroke
closepath_stroke
fill
fill_stroke
closepath_fill_stroke
clip
endpath

パスオブジェクト

add_path_point
draw_path
info_path
delete_path

PDI

open_pdi_document
open_pdi_callback（C/C++のみ）
close_pdi_document
open_pdi_page
close_pdi_page
fit_pdi_page
info_pdi_page
process_pdi

pCOS

pcos_get_number
pcos_get_string
pcos_get_stream

ブロック流し込み（PPS）

fill_textblock
fill_imageblock
fill_pdfblock
fill_graphicsblock

インタラクティブ機能

create_action
add_nameddest
create_annotation
create_field
create_fieldgroup
create_bookmark
add_portfolio_folder
add_portfolio_file

マルチメディア

load_3ddata
create_3dview
load_asset

文書交換

set_info
begin_item
end_item
activate_item
begin_mc
end_mc
mc_point

B 全オプション・キーワード一覧

この索引は、すべてのオプションをアルファベット順に並べて、それを使える関数を併記したものです。ページ番号をクリックするとその説明へ飛びます。

B

C

D

E

F

G

H

I

J

K

L

M

N

R

S

T

U

V

W

X

Y

Z

C 改訂履歴

日付	更新点
2022 年 7 月 31 日	▸ PDFlib 10.0.0 に関する更新
2021 年 6 月 19 日	▸ PDFlib 9.3.1 に関する更新
2020 年 8 月 14 日	▸ PDFlib 9.3.0 に関する更新
2019 年 2 月 01 日	▸ PDFlib 9.2.0 に関する更新
2018 年 2 月 01 日	▸ PDFlib 9.1.2 に関する更新
2017 年 7 月 24 日	▸ PDFlib 9.1.1 に関する更新
2017 年 3 月 06 日	▸ PDFlib 9.1.0 に関する更新
2016 年 8 月 16 日	▸ PDFlib 9.0.7 に関する更新
2016 年 1 月 25 日	▸ PDFlib 9.0.6 に関する更新
2015 年 5 月 24 日	▸ PDFlib 9.0.5 に関する更新
2014 年 12 月 16 日	▸ PDFlib 9.0.4 に関する更新
2014 年 5 月 12 日	▸ PDFlib 9.0.3 に関する更新
2013 年 12 月 17 日	▸ PDFlib 9.0.2 に関する更新
2013 年 10 月 17 日	▸ PDFlib 9.0.1 に関する更新
2013 年 5 月 11 日	▸ PDFlib 9.0.0 に関する更新
2011 年 5 月 30 日	▸ PDFlib 8 VT Edition（内部的には 8.1.0）に関する更新
2011 年 5 月 30 日	▸ PDFlib 8.0.3 に関するさまざまな更新・修正
2010 年 12 月 09 日	▸ PDFlib 8.0.2 に関するさまざまな更新・修正
2010 年 9 月 22 日	▸ PDFlib 8.0.1p7 に関するさまざまな更新・修正
2010 年 4 月 13 日	▸ PDFlib 8.0.1 に関するさまざまな更新・修正
2009 年 12 月 04 日	▸ PDFlib 8 に関する更新
2009 年 3 月 13 日	▸ PDFlib 7.0.4 に関するさまざまな更新・修正
2008 年 2 月 13 日	▸ PDFlib 7.0.3 に関するさまざまな更新・修正
2007 年 8 月 08 日	▸ PDFlib 7.0.2 に関するさまざまな更新・修正
2007 年 3 月 09 日	▸ PDFlib 7.0.1 に関するさまざまな更新・修正
2006 年 10 月 03 日	▸ PDFlib 7.0.0 に関する更新と再構成。説明書をチュートリアルと API リファレンスに分割
2006 年 2 月 21 日	▸ PDFlib 6.0.3 に関するさまざまな更新・修正。Ruby の節を追加
2005 年 8 月 09 日	▸ PDFlib 6.0.2 に関するさまざまな更新・修正

日付	更新点
2004 年 11 月 17 日	► PDFlib 6.0.1 に関する小規模な更新・修正 ► 8 章に言語別関数プロトタイプ用新書式導入 ► 3 章にハイパーテキストの例を追加
2004 年 6 月 18 日	► PDFlib 6 に関する大規模な変更
2004 年 1 月 21 日	► PDFlib 5.0.3 に関する小規模な追加・修正
2003 年 9 月 15 日	► PDFlib 5.0.2 に関する小規模な追加・修正。ブロックの仕様を追加
2003 年 5 月 26 日	► PDFlib 5.0.1 に関する小規模な更新・修正
2003 年 3 月 26 日	► PDFlib 5.0.0 に関する全面的変更と書き直し
2002 年 6 月 14 日	► PDFlib 4.0.3 に関する小規模な変更と .NET バインディングに関する追加
2002 年 1 月 26 日	► PDFlib 4.0.2 に関する小規模な変更と IBM eServer エディションに関する追加
2001 年 5 月 17 日	► PDFlib 4.0.1 に関する小規模な変更
2001 年 4 月 1 日	► PDFlib 4.0.0 の PDI・他機能を解説
2001 年 2 月 5 日	► PDFlib 3.5.0 のテンプレート・CMYK 機能を解説
2000 年 12 月 22 日	► ColdFusion 解説と PDFlib 3.03 に関する追加。COM エディションを別マニュアルに
2000 年 8 月 8 日	► Delphi 解説と PDFlib 3.02 に関する小規模な追加
2000 年 7 月 1 日	► PDFlib 3.01 に関する追加・説明の明瞭化
2000 年 2 月 20 日	► PDFlib 3.0 に関する変更
1999 年 8 月 2 日	► PDFlib 2.01 に関する小規模な変更・追加
1999 年 6 月 29 日	► 言語バインディングごとに節を分離 ► PDFlib 2.0 に関する追加
1999 年 2 月 1 日	► PDFlib 1.0 に関する小規模な追加（公開せず）
1998 年 8 月 10 日	► PDFlib 0.7 に関する追加（1 つのお客様用のみ）
1998 年 7 月 8 日	► PDFlib 0.6 の PDFlib スクリプティングサポートを初めて記述
1998 年 2 月 25 日	► PDFlib 0.5 に関して説明を若干追加
1997 年 9 月 22 日	► PDFlib 0.4 と本マニュアルを初めて公開

索引

オプションを別途一覧化したものが 317 ページ「付章 B」にあります。

PDFlib GmbH

PDFlib GmbH
Franziska-Bilek-Weg 9
80339 München, Germany
www.pdflib.com
電話 +49 • 89 • 452 33 84-0

ライセンス発行のお問い合わせ
sales@pdflib.com

サポート
support@pdflib.com（お持ちのライセンス番号をお書きください）