PDFlib GmbH

PDFlib, PDFlib+PDI, PPS

A library for generating PDF on the fly
Version 8.0.3

API リファレンス

Cobol・C・C++・Java・Perl・
PHP・Python・RPG・Ruby・Tcl エディション

PDFlib GmbH
Franziska-Bilek-Weg 9, 80339 München, Germany
www.pdflib.com
電話 *+49 • 89 • 452 33 84-0*
FAX +49 • 89 • 452 33 84-99

疑問点がおありの場合は *tech.groups.yahoo.com/group/pdflib* にある *PDFlib* メーリングリストとアーカイブをご覧ください。

ライセンス取得のための連絡先：*jp.sales@pdflib.com*
商用 *PDFlib* ライセンス保持者向けサポート：*jp.support@pdflib.com*（ライセンス番号をお知らせください）

目次

1 オプションリスト

オプションリストは、さまざまな API 関数呼び出しを制御するための強力な、それでいて簡単な手段です。関数に多数の引数を与える必要なしに、多くの API メソッドで、オプションリストを用いることができます。略して *optlist* ともいいます。これは、オプションを何個でも内容として持つことのできる文字列です。オプションリストには、さまざまなデータ型や、リストなどの複合データを内容として持たせることができます。多くの言語バインディングで optlist は簡単に、必要なキーワードと値を文字列連結していくだけで作成することができます。

バインディング C 言語バインディング：*sprintf( )* 関数を使って optlist を作成するとよいでしょう。

1.1 オプションリストの文法

オプションリストの文法形式の定義 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すべての要素（キーと値）は、右記の区切りキャラクタのうちの 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。
- 最も外側の中括弧対の内側にある区切りキャラクタは、要素を区切らず、要素の一部分となります。ですので、区切りキャラクタのあ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（群）を含むときは、各キャラクタの直前にバックスラッシュキャラクタ 1 個を付けて保護する必要があります。
- 要素が、中括弧キャラクタの直前にバックスラッシュキャラクタを 1 個ないし複数連続して含んでいるときは、そのキャラクタ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を付けて保護する必要があります。
- 要素に、開閉照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をつける必要があります。
- オプションリストにバイナリゼロ値を含めてはいけません。

オプションは、この PDFlib リファレンスでの記述に従って、リスト値を内容として持つことができる場合があります。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、等号は区切りキャラクタ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 / 値対を内容として持ちます。キーと値の間、およびキー / 値対どうしの間は、1 個ないし複数

の空白キャラクタ（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 / 値対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上書きします。下記の例では、最初のオプション割り当てを次のものが上書きして、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 *{}* で囲まれ、リスト内の値の間は空白キャラクタ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入れ子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 *}* キャラクタと *{* キャラクタとの間には区切りキャラクタ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、入れ子に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}}                   （入れ子のリスト1個を内容として持つリスト）
```

入れ子に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、入れ子にされた 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入れ子に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}
```

リストが矩形 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ではありませんので、下記は誤りです：

```
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、下記も区切りキャラクタがありませんので誤りです：

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

中括弧は開閉照応している必要がありますので、下記は誤りです：

```
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：

```
filename={C:\path\name\}                      誤り！
filename={C:\path\name\\}                     正しい！
```

1.2 基本型

文字列　文字列はプレーンな ASCII 文字列であり、通常、ローカライズ対象でないキーワードに用いられます。文字列の中に空白キャラクタか「=」キャラクタがあるときは、*{ }* 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 *{* か *}* を文字列に入れたいときは、直前にキャラクタ \ 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対で作成できます：

{}

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。パラメタ *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。

Unicode 非対応言語バインディング：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。

Unichar　Unichar は 1 個の Unicode 値であり、右記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！）
```

Unicode 範囲　Unicode 範囲は、連続的な範囲の Unicode キャラクタ群を、その範囲の先頭キャラクタと末尾キャラクタ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、固定キーワード群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・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 関数によって以前に返された整数値です。たとえば、フォントハンドルは *PDF_load_font()* 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 関数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フォントハンドルを与えてはいけません。

1.3 文字サイズ・色・アクション

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一覧

オプション	説明
ascender	数値は、アセンダ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とアセンダ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。

色　色は、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 のすべての有効な色名を直接与えて、RGB 色を指定することができます。例：.

```
strokecolor=pink
```

この色名は、大文字/小文字を区別します。有効な色名の一覧は、下記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「#」キャラクタの直後に、3 個の 16 進数対 00 ～ FF の任意の組み合わせを与えて、RGB 色値を指定することができます。例：

```
strokecolor=#FFC0CB
```

第三の形式は、色空間と色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。色空間キーワードは、*PDF_setcolor()*（145 ページ「8.1 色と色空間の設定」参照）に対するものと同じです。具体的な説明と例を表 1.2 に示します。各関数の説明で記しますが、オプションリストによっては、上記のキーワードの一部分しか与えないものもあります。

クックブック　**完全なコードサンプルがクックブックの color/starter_color トピックにあります。**

表 1.2 オプションリスト内の色データ型のキーワード一覧

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	スポットカラーハンドルと、濃度値を指定する float 1 個	{ spot 1 0.8 }
spotname	（最大 63 バイト。Unicode キャラクタ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、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、色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 } }
iccbasedgray	float 値 1 個	{ iccbasedgray 0.5 }
iccbasedrgb	float 値 3 個	{ iccbasedrgb 1 0 0 }
iccbasedcmyk	float 値 4 個	{ iccbasedgray 0 1 0 0 }
pattern	パターンハンドル	{ pattern 1 }
none	色がないことを示します	none

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）1 個と、アクションハンドル群のリスト 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 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 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 }
```

1.4 図形型

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、*n>2* の偶数 *n* 個の float 値を内容として持つリストです。リスト内のそれぞれの対は、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。これは矩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。

矩形　矩形は、float 値 4 個のリストであり、これらは、矩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）。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。座標リストの中で、相対座標は、直前の *x* 座標か *y* 座標に対する値となります。相対座標がリストの先頭にあるときは、原点に対する値となりますので、すなわち絶対座標となります。例：

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

以下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 番目の対は、中心の *x* 座標と *y* 座標を指定し、2 番目の対は、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。よって、曲線リストは、n ≧ n の 6 × n 個の float 値のリストとして指定されます：

```
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。

1.5 制限

PDF リファレンス、Acrobat、およびいくつかの PDF 標準のよって課されている制限条項に準拠した PDF 出力を生成するために、PDFlib ではいくつかの項目に制約を設けています。これらの制約を以下に示します。

以下の制約は、強制的に守らせるよう、値が適切に変更されます。

- PDF における浮動小数点値の最小の絶対値 : 0.000015。これよりも絶対値が小さい数は 0 に置き換えられます。
- (PDF 1.4。それより後の PDF バージョンにはあてはまりません)PDF において浮動小数点数値として表すことのできる最大の絶対値 : 32767.0。これよりも絶対値が大きい数は、いちばん近い整数に置き換えられます。

以下の制限に違反したときは、例外が発生します。

- PDFlib では、PDF 出力ファイルサイズについて、何ら固定的な制限はありませんが、PDF/A-1・PDF/X-4・PDF/X-5 の生成時には、特定の制限を強制する必要があります。詳しくは PDFlib チュートリアルを参照してください。
- PDF で許容される最大の数値 : 2,147,483,647。
- ハイパーテキスト文字列の最大長 : 65535。
- ページ上のテキスト文字列の最大長 : *kerning=false* かつ *wordspacing=0* の場合は 32,763 バイト (すなわち CID フォントなら 16,381 キャラクタ)、それ以外の場合は 4095 キャラクタ
- 以下のオプションのリスト項目は最大 8191 個に制限されます。
  views・namelist・polylinelist・fieldnamelist・itemnamelist・itemtextlist・children・group
- PDF/A-1・PDF/X-4・PDF/X-5 文書 1 個の中の間接オブジェクトの最大数 : 8,388,607

2 一般関数

2.1 関数のスコープ

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 の API は、文書やページのつくりに非常に近い設計になっていますので、文書を「自然な」やり方で生成すれば、正しい PDFlib クライアントプログラムができあがります。PDFlib は、関数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 2.1 に示します。スコープの入れ子を図 2.1 に図示します。関数の解説では、各関数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関数を呼ぶと、例外が発生します。*scope* パラメタを使うと、カレントのスコープを知ることができます。

クックブック **完全なコードサンプルがクックブックの `general/function_scopes` トピックにあります。**

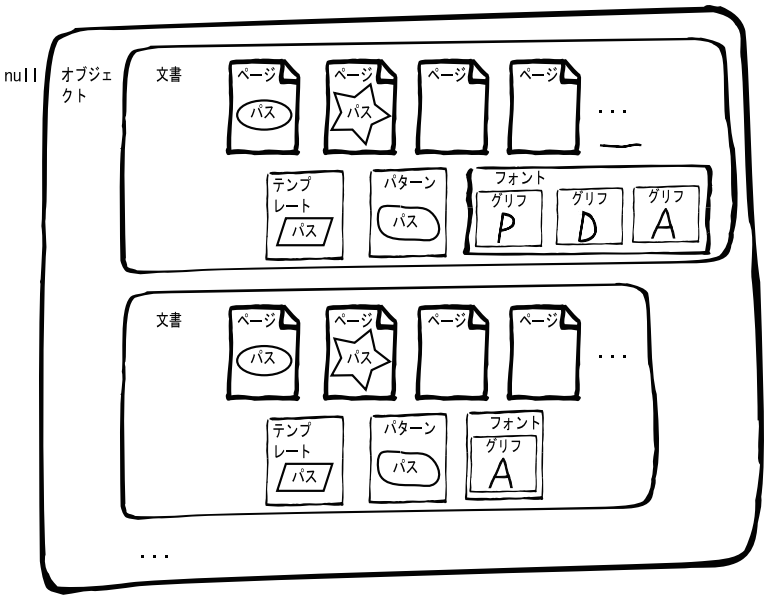


図 2.1
スコープの入れ子

表 2.1 関数のスコープの定義一覧

スコープ名	定義
パス	PDF_moveto( )・PDF_circle( )・PDF_arc( )・PDF_arcn( )・PDF_rect( )・PDF_ellipse( ) のいずれかで開始。137 ページ「7.5 描画とクリッピング」のいずれかの関数で終了
ページ	PDF_begin_page( ) と PDF_end_page( ) 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( ) 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( ) 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・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	オブジェクト指向の言語バインディング：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。 それ以外のバインディングでは PDF_new( ) と PDF_delete( ) の間、ただし文書スコープの外
null	オブジェクトスコープの外
任意	関数の説明で「任意」スコープと言うときは、実際には null を除く任意を意味します。なぜなら null スコープでは、PDFlib オブジェクト自体が存在していないからです。

2.2 パラメタ・オプション処理

PDFlib の操作は、さまざまなグローバルパラメタで制御することができます。文字列と数値のパラメタがあり、それらを使って、PDFlib や、PDF 出力の体裁を制御することができます。数値と文字列のパラメタを設定または取得するために、4 つの関数があります。それぞれの節の最初に、関連するパラメタのキー名と値を記します。使えるパラメタの全一覧を「付章 B 全パラメタ一覧」に示します。

これらのパラメタの設定は、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。ただしパラメタによっては、各ページが始まるごとに明示的に再設定されるものもあります（これは各説明でその旨をそのつど示します）。

C++ Java ***double get_value(String key, double modifier)***
Perl PHP ***float get_value(string key, float modifier)***
C ***double PDF_get_value(PDF *p, const char *key, double modifier)***

数値型の PDFlib パラメタの値を得ます。

key 取得したいパラメタの名前。

modifier もしあれば、パラメタに適用したい修飾子。修飾子が必要かどうか、そして何に関するかは、さまざまなパラメタ一覧表で説明します。修飾子を使わないときは 0 にする必要があります。多くのパラメタでは修飾子として、ハンドルを渡す必要があります。

戻り値 パラメタの数値。

スコープ *key* に依存。

C++ Java ***void set_value(String key, double value)***
Perl PHP ***set_value(string key, float value)***
C ***void PDF_set_value(PDF *p, const char *key, double value)***

数値型の PDFlib パラメタに値を設定します。

key 設定したいパラメタの名前。

value パラメタに設定したい新しい値。

スコープ *key* に依存。

C++ Java ***String get_parameter(String key, double modifier)***
Perl PHP ***string get_parameter(string key, float modifier)***
C ***const char * PDF_get_parameter(PDF *p, const char *key, double modifier)***

文字列型の PDFlib パラメタの内容を得ます。

key 取得したいパラメタの名前。

modifier もしあれば、パラメタに適用したい修飾子。修飾子が必要かどうか、そして何に関するものかは、さまざまなパラメタ一覧表で説明します。修飾子を使わないときは 0 にする必要があります。多くのパラメタでは修飾子として、ハンドルを渡す必要があります。

戻り値 パラメタの文字列値を、ハイパーテキスト文字列として。返された文字列は、カレント**文書**スコープを終えるまで使えます。情報が得られないときは空文字列が返されます。

スコープ *key* に依存。

バインディング C 言語バインディング：返された文字列をクライアント側で解放してはいけません。PDFlib がすべての文字列リソースを内部的に管理しています。

C++ Java ***void set_parameter(String key, String value)***
Perl PHP ***set_parameter(string key, string value)***
C ***void PDF_set_parameter(PDF *p, const char *key, const char *value)***

文字列型の PDFlib パラメタを設定します。

key 設定したいパラメタの名前。

value （名前文字列）パラメタに設定したい新しい値。

スコープ *key* に依存。

C++ Java ***void set_option(String optlist)***
Perl PHP ***set_option(string optlist)***
C ***void PDF_set_option(PDF *p, const char *optlist)***

グローバルなオプション（複数可）を設定します。

optlist 表 2.2 に従ったグローバルなオプション群を指定するオプションリスト。1 つのオプションが複数回与えられたときは、最後に出現したものがそれ以前のすべてを上書きします。1 つのオプションに対して複数の値を与えるためには（例：*searchpath*）、そのオプションに対する 1 つのリスト引数の中ですべての値を与えてください。下記のオプションを用いることができます：

filenamehandling・logging・resourcefile・searchpath・shutdownstrategy

詳細 *searchpath* 以外では、新しい値は古い値を上書きします。*PDF_set_option( )* は、*PDF_set_parameter( )* のパラメタ群のサブセットに対応しています。

スコープ *any*

表 2.2 PDF_set_option( ) のオプション一覧

オプション	定義
filename-handling	（キーワード。Windows では不必要）ファイル名に関する対象のエンコーディング（デフォルト：Mac OS X では unicode、それ以外では legacy）： ***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*** 環境変数 LANG を解釈してファイル名に適用、ただしそれが utf8・UTF-8・cpXXXX・CPXXXX・iso8859-x・ISO-8859-x のいずれかを指定しているときのみ。 ***legacy*** host エンコーディング（すなわちカレントシステムのエンコーディング）を用いてファイル名を解釈し、かつ、honorlang パラメタが設定されていれば LANG 変数を解釈。 ***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** PDFlib によって認識される任意の（内蔵またはユーザー定義の）エンコーディング（表 4.3 参照）、ただし glyphid・builtin を除く
logging	（オプションリスト）表 2.8 に従ったログ記録オプション群
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上書きされます。
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ディレクトリの相対パス名または絶対パス名（複数可）。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）。空の名前文字列（すなわち {{}}）は、既存の検索パス項目群をすべて削除します。Windows は、検索パスはレジストリ項目で設定することもできます。デフォルト：空
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な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 PDF 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れて参照カウンタがゼロに減ったとき、そのリソースは解放されます。 ***1*** リソース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を必要とします。

2.3 セットアップ

PDFlib のセットアップに関連するパラメタと値のキー名を表 2.3・表 2.4 に示します（19 ページ「2.2 パラメタ・オプション処理」参照）。

表 2.3 PDF_get/set_parameter( ) のセットアップ関連のキー一覧

キー	説明
任意のリソースカテゴリ名	任意のリソースカテゴリ内の項目。PDF_get_parameter( ) の場合：修飾子は項目番号をとります（先頭が 1）。項目がもうないときは空文字列が返されます。カテゴリ名の一覧は PDFlib チュートリアルを参照。スコープ：任意
asciifile	(iSeries・zSeries のみ）テキストファイル（PFA・AFM・UPR・エンコーディング）を ASCII エンコーディングと見なします。デフォルト：iSeries では true、zSeries では false。スコープ：任意
filename-handling	表 2.2 に従ったファイル名に対する対象エンコーディング
honorlang	（非推奨。filenamehandling=honorlang を用いてください）true にすると、環境変数 LANG が解釈されて、それが utf8・UTF-8・cp1252・CP1252・iso8859-x・ISO-8859-x のいずれかを指定しているときは、ファイル名に適用されます。デフォルト：false。スコープ：オブジェクト
license[1]	PDFlib か PDFlib+PDI か PPS のライセンスキーを設定します。キーを設定できるのは、初めて PDF_begin_document( ) を呼び出す前です。誤ったプラットフォームに対するライセンスキーは、警告なく無視されます。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、nodemostamp パラメタを用いてください。スコープ：オブジェクト
licensefile	ライセンスキーの入ったファイルの名前を設定します。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。スコープ：オブジェクト
nodemo-stamp	true なら、有効なライセンスキーが見つからなかったときに例外が発生します。false なら、全ページにデモスタンプが生成されます。このオプションは、初めて PDF_begin_document( ) を呼び出す前に設定する必要があります。デフォルト：false。スコープ：オブジェクト
resourcefile	PDFlib の UPR リソースファイルの相対または絶対ファイル名。その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、その値は再設定すれば新しい値で上書きされます。スコープ：任意
scope[2]	カレントスコープの名前を返します（表 2.1 参照）。スコープ：任意
SearchPath	読みたいファイルの入ったディレクトリの相対または絶対パス名。この SearchPath は複数回設定することもでき、その内容は蓄積されて、設定した順序に使われます。空文字列だと SearchPath リストの全文字列を（デフォルト項目群も）削除します。 PDF_get_parameter( )：修飾子は項目番号をとります（先頭が 1）。項目がもうないときは空文字列が返されます。返される文字列のエンコーディングは UTF-8 です。スコープ：任意
string[2]	修飾子で与える文字列番号で指定される文字列を返します。返された文字列は、次に何らかの API 関数を呼び出すまで有効です。スコープ：任意
version[2]	PDFlib の完全なバージョン文字列。その形式は <メジャー>.<マイナー>.<リビジョン> で、さらに末尾に beta・rc 等の限定詞がつくこともあります。スコープ：任意・null[3]

1. PDF_set_parameter( ) のみ
2. PDF_get_parameter( ) のみ
3. 引数 PDF * を NULL か 0 にして呼び出すことができます

表 2.4 PDF_get/set_value( ) のセットアップ関連のキー一覧

キー	説明
compress	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。このパラメタは、パススルーモードで処理される画像データに対しては効力を持ちません。デフォルト：6。スコープ：ページ・文書
major minor revision[1]	それぞれ PDFlib のメジャー・マイナー・リビジョン番号を返します。スコープ：任意・null[2]
maxfile-handles	（非サポート。Windows にのみ実装）同時に開かれている（C ランタイム内で）ファイルの数の新しい最大値。この数は 20 以上、2048 以下である必要があります。この新しい値が C ランタイムによって受け付けられなかったときは、例外が発生します。スコープ：オブジェクト

1. PDF_get_value( ) のみ
2. 引数 PDF * を NULL か 0 にして呼び出すことができます

C ***PDF *PDF_new(void)***

新規 PDFlib オブジェクトを作成します。

詳細 この関数は、PDFlib 内部のデフォルトのエラー処理ルーチンとメモリ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。

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、メモリ不足で成功しなかったときは、NULL を返すか（C の場合）、または例外を発生させます。

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( )* と必ず対にして呼び出す必要があります。

バインディング 不透明な PDFlib オブジェクトハンドルに用いられるデータ型は、言語バインディングによって異なります。このことが PDFlib のクライアントに実際に影響を与えることはありません。なぜならクライアントはただ、PDFlib ハンドルをすべての関数の 1 番目の引数として渡せばよいだけだからです。

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 関数へのポインタを代入された *PDFlib_api* 構造体へのポインタを返します。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。
C++・Java・Perl・PHP：この関数は 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, int errortype, 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。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割り当て関数へのポインタ。

reallocproc ユーザー定義のメモリ再割り当て関数へのポインタ。

freeproc ユーザー定義のメモリ解放関数へのポインタ。

opaque 何らかのユーザーデータへのポインタ。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、メモリ不足で成功しなかったときは、NULL を返すか（C の場合）、また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()* と必ず対にして呼び出す必要があります。この関数を呼び出した後には、同じ PDFlib オブジェクトを持つ他の PDFlib 関数は一切呼び出す必要がありません。

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。与える関数はすべて「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、PDF 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。単発の文書生成ならば必要ないかもしれませんが、すべてのサーバアプリケーションに対しては、PDF 作成が済んだら PDF オブジェクトを削除することを強く推奨します。この関数は、1 つの PDF 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**。この関数は*null*スコープを開始させます。API関数への呼び出しはもう許されません。

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++ : この関数は PDF デストラクタで間接的に得られます。
Java : この関数はラッパ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。
Perl・PHP : この関数は、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。

2.4 PDFlib 仮想ファイルシステム（PVF）

クックブック　完全なコードサンプルがクックブックの general/starter_pvf トピックにあります。

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

Java　***void create_pvf(string filename, const void *data, size_t size, 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としてスラッシュ「/」キャラクタ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* 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。

data　仮想ファイルに入れたいデータへの参照。C・C++ ではこれはメモリ位置へのポインタです。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++ のみ）データの入ったメモリ領域の長さをバイト単位で表したもの。

optlist　表 2.5 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 関数に与えることができます（仮想ファイルを生成PDF出力に使うことはできません。それをするには*PDF_begin_document()* で空のファイル名を使います）。こうした関数のなかには、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。この関数は、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、対になる *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 2.5 PDF_create_pvf( ) のオプション一覧

オプション	説明
copy	（論理値）PDFlib が、与えたデータの内部的なコピーをただちに作成します。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。この copy オプションは、COM・.NET・Java バインディングでは自動的に true に設定されます（その他のバインディングではデフォルト：false）。それ以外の言語バインディングでは、copy オプションを与えなければデータはコピーされません。

C++ Java ***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（ただし内容は解放しません）。

filename （名前文字列。グローバルな*filenamehandling*オプションまたはパラメタに従って解釈されます。表 2.2 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* 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。

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。

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ただちに削除します。*filename* という有効な仮想ファイルがないときは、この関数は警告も出さず何もしません。この関数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、対になる *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

2.5 例外処理

この節に関連するオプションを表 2.6 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。これらのオプションは、*PDF_get/set_parameter( )* に対するパラメタキー名としても用いることができます（19 ページ「2.2 パラメタ・オプション処理」参照）。

表 2.6 PDF_get/set_parameter( ) の例外関連のキー一覧（オプションとしても使用可能）

キー	説明
errorpolicy	（キーワード）エラー発生時のさまざまな関数の動作を制御します。errorpolicy パラメタは、さまざまな関数の errorpolicy オプションによって上書きすることができ、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：legacy。スコープ：任意）： *legacy* （非推奨）関数の動作は PDFlib 6 と同じです。 *return* エラーが起きると関数は返ります。エラーコードを返せる関数（PDF_load_image( ) 等）は -1（PHP では 0）を返します。結果文字列を返す関数（PDF_fit_table( ) 等）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。 *exception* エラーが起きると、関数は例外を発生させます。この例外は、バインディング固有のしくみを用いてクライアントコード内で補足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。

C++ Java ***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、または関数呼び出し失敗の原因の番号を得ます。

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。あるいは関数がエラーコード -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、この関数を呼び出すこともできます。

バインディング C++・Java・PHP 5 では、この関数は *PDFlibException* オブジェクトで *get_errnum( )* としても得られます。

C++ Java ***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、または関数呼び出し失敗の原因のテキストを得ます。

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。あるいは関数がエラーコード -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、この関数を呼び出すこともできます。

バインディング　C++・Java・PHP 5 では、この関数は *PDFlibException* オブジェクトで *get_errmsg( )* としても得られます。

C++ Java ***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 関数の名前を得ます。

戻り値　例外を発生させた API 関数の名前か、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関数の名前。

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。あるいは関数がエラーコード −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、この関数を呼び出すこともできます。

バインディング　C++・Java・PHP 5 では、この関数は *PDFlibException* オブジェクトで *get_apiname(( )* としても得られます。

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。

詳細　PDFlib は、この不透明ポインタを一切さわらずに、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。

2.6 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。ログ記録機能のためのパラメタのキー名を表 2.7 に示します（19 ページ「2.2 パラメタ・オプション処理」参照）。

表 2.7 PDF_set_parameter( ) のログ記録関連のキー一覧

キー	説明
logging	表 2.8 に従ったログ記録オプションによるオプションリスト
logmsg	ログファイルへコピーさせたい文字列

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 *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=debug.log remove")
  ```

- *PDF_set_parameter( )* の *logging* オプションに対するオプションリストとして。例：

  ```
  p.set_parameter("logging", "filename=debug.log 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 関数を呼び出した時からログ出力が始められます。

表 2.8 logging パラメタのオプション一覧

キー	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　MVS の場合 PDFlib.log　Mac・iSeries 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、PDFLIBLOGFILE という環境変数で与えることもできます。
flush	（論理値）true なら、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、出力が確実にフラッシュされます。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 ならログファイルは 1 回だけ開かれます。デフォルト：false
remove	（論理値）true なら、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
stringlimit	（整数）行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 2.8 logging パラメタのオプション一覧

キー	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、その関数パラメタ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、非推奨の関数・オプションはその旨が示されます。api=3 にすると、try/catch 呼び出しがログ記録されます（例外処理が入れ子になっている問題のデバッグに有用）。 ***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。 ***user*** logmsg パラメタで与えている、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( ) で取得させる関数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。

3 文書・ページ関数

3.1 文書関数

C++ Java ***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。グローバルな*filenamehandling*オプションまたはパラメタに従って解釈されます。表 2.2 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( )* 関数を使う必要があります。特殊ファイル名「-」を使うと、PDF を *stdout* チャンネルに生成できます。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* 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。

writeproc （C・C++ 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：

- 一般オプション群：*errorpolicy*（表 2.6 参照）・*hypertextencoding*（表 12.1 参照）
- 表 3.1 に従った文書オプション群。*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：
  attachmentpassword・attachments・autoxmp・compatibility・destination・filemode・flush・groups・inmemory・labels・lang・linearize・masterpassword・metadata・moddate・objectstreams・openmode・optimize・pagelayout・pdfa・pdfx・permissions・recordsize・rolemap・search・tagged・tempdirname・tempfilenames・uri・userpassword・viewerpreferences

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、この関数は必ず成功し、決してエラー値 -1（PHP では 0）を返しません。

詳細 この関数は、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関数を呼び出して PDF 出

力データを受け渡します。*writeproc* で与えるコールバック関数は、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。

スコープ **オブジェクト**。この関数は、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( )* と必ず対にして呼び出す必要があります。

バインディング 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 ***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：

- 一般オプション群：*errorpolicy*（表 2.6 参照）・*hypertextencoding*（表 12.1 参照）
- 表 3.1 に従った文書オプション群。*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・attachmentpassword・attachments・autoxmp・createpvf・destination・destname・labels・metadata・moddate・objectstreams・openmode・pagelayout・portfolio・rolemap・search・uri・viewerpreferences

詳細 この関数は、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。この関数は、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関数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 *PDF_delete( )* でスコープ外に出たときに解放されます。

スコープ **文書**。この関数は**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対にして呼び出す必要があります。

表 3.1 PDF_begin_document( )・PDF_end_document( ) の文書オプション一覧

オプション	説明
action[1]	（アクションリスト。PDF/A では不可）以下のイベント（複数可）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 は open アクションとして使えません。 ***didprint/didsave/willclose/willprint/willsave*** （PDF 1.4）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。

表 3.1 PDF_begin_document( )・PDF_end_document( ) の文書オプション一覧

オプション	説明
attachment-password[2, 3]	（文字列。PDF 1.6。userpassword か masterpassword を設定しているときは無視されます。linearize・optimize オプションとの併用は不可）ファイル添付が、指定文字列をパスワードとして暗号化されます。文書のそれ以外は暗号化されません。
attachments	（オプションリストのリスト）文書レベルのファイル添付を指定します（ページ上の特定箇所に結びつく添付注釈とは異なります）。PDF_begin_document( ) と PDF_end_document( ) の両方でファイル添付を与えてもかまいません。使えるオプション： **description**（ハイパーテキスト文字列。PDF 1.6）ファイルに関する説明テキスト。 ***filename***（ハイパーテキスト文字列。必須）ファイルの名前。Unicode ファイル名は使えますが、ただし Acrobat で正しく表示されるためには PDF 1.7 が必要です。 ***mimetype***（文字列）ファイルの MIME タイプ。Acrobat は、添付が開かれた時に、適切なアプリケーションを起動させるためにこれを用います。 ***name***（ハイパーテキスト文字列）添付の名前。デフォルト：filename からパス要素をすべて除いたもの
autoxmp	（論理値。PDF/X-3/4/5・PDF/A-1 モードでは true を強制されます）true にすると、PDFlib は文書情報フィールドから XMP 文書メタデータを作成します（237 ページ「14.2 XMP メタデータ」参照）。デフォルト：false
compatibility[2]	（キーワード。pdfx・pdfa オプションのいずれかが none 以外の値で用いられているときは無視されます）文書の PDF バージョンを、以下に挙げる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 生成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：1.7）： ***1.3*** PDF 1.3。Acrobat 4 以上が必要です。 ***1.4*** PDF 1.4。Acrobat 5 以上が必要です。 ***1.5*** PDF 1.5。Acrobat 6 以上が必要です。 ***1.6*** PDF 1.6。Acrobat 7 以上が必要です。 ***1.7*** PDF 1.7。ISO 32000-1 で定義されており、Acrobat 8 以上が必要です。 ***1.7ext3*** PDF 1.7 拡張レベル 3。Acrobat 9 以上が必要です。***1.7ext8***PDF 1.7 拡張レベル 8。Acrobat X 以上が必要です。
createpvf[2]	（論理値）true にすると、PDF ファイルをファイル上でなくメモリ内に生成します。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 関数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
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ときは無視されます）文書を開くアクションを表 12.5 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ときは無視されます）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。
filemode[2]	（文字列。MVS のみ）文書ファイルと任意の一時ファイル（例：linearize オプションによる）のファイルモードを設定するパラメタ文字列。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タ群と整合させる必要があります。文字列の例：recfm=fb,lrecl=80,space=(cyl,(1,5),rlse。デフォルト：空、または非ブロック出力の場合は recfm=v。

表 3.1 PDF_begin_document( )・PDF_end_document( ) の文書オプション一覧

オプション	説明
flush[2]	（キーワード。PDF_begin_document_callback( ) のみ）フラッシュ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 回だけフラッシュ ***page*** 各ページが終わるごとにフラッシュ ***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フラッシュ ***heavy*** 内部の 64 KB の文書バッファがいっぱいになるたびにフラッシュ
groups[2]	（文字列のリスト）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。
inmemory[2]	（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作成せず、ファイルをメモリ内で処理します。これはシステムによっては（特に MVS）、驚異的な速度向上につながりますが、文書のサイズの 2 倍のメモリが必要になります。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
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 3.2 に従って指定したオプションリスト（複数可）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：none
lang[2]	（文字列。tagged=true にしているときは推奨）文書の自然言語を、2 字の ISO 639 言語コードで設定します（例：DE・EN・FR・JA）。その後に、ハイフンと 2 字の ISO 3166 国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。 この言語指定は、構造ツリーのあらゆるレベルで個々のアイテムについて上書きできますが、はじめに文書の全体について設定しておく必要があります。
linearize[2]	（論理値。PDF_begin_document_callback( ) では不可）true にすると出力文書は線形化されます。MVS システムでは、このオプションはインコア生成（すなわち filename を空に）とは併用できません。デフォルト：false
master-password[2, 3]	（文字列。permissions を指定しているときは必須。PDF/A・PDF/X では不可）文書のマスターパスワード。空にするとマスターパスワードは適用されません。デフォルト：空
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（237 ページ「14.2 XMP メタデータ」参照）。この XMP は、PDF_set_info( ) で与えている文書情報項目を上書きします。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）。
moddate	（論理値）true にすると、いくつかのプリフライトツールに準拠するために ModDate（更新日時）文書情報キーが作成されます。デフォルト：false

表 3.1 PDF_begin_document( )・PDF_end_document( ) の文書オプション一覧

オプション	説明
objectstreams[2]	（キーワードのリスト。PDF 1.5。optimize または linearize が true のときは false が強制されます）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します。 ***docinfo*** 文書情報フィールド群を圧縮します。 ***fields*** フォームフィールド群を圧縮します。 ***names*** 名前付き参照先を持つオブジェクト群を圧縮します。 ***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、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します。 ***tags*** マークされた内容タグ群を圧縮します。 ***xref*** 圧縮された xref ストリームを生成します。このカテゴリは、他のカテゴリを 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。
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、何度も読み込んだ場合など）を除き、最適化は目立った効果をもたらしません。MVS システムでは、このオプションはインコア生成（すなわち filename を空に）とは併用できません。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*** Acrobat ビューアのデフォルト設定。 ***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。
pdfa[2]	（キーワード）PDF/A 準拠レベルを、PDF/A-1a:2005・PDF/A-1b:2005・none のいずれかに設定します。PDF/A-1a:2005 の場合、タグ付き PDF モードが自動的に有効になります。PDF/A-1 出力は同時に、pdfx オプションの PDF/X-1a:2003・PDF/X-3:2003・PDF/X-4 設定に準拠させることが可能です。デフォルト：none

表 3.1 PDF_begin_document( )・PDF_end_document( ) の文書オプション一覧

オプション	説明
pdfx[2]	（キーワード）PDF/X 準拠レベルを、PDF/X-1a:2001・PDF/X-1a:2003・PDF/X-3:2002・PDF/X-3:2003・PDF/X-4・PDF/X-4p・PDF/X-5g・PDF/X-5pg・none のいずれかに設定します。デフォルト：none
permissions[2]	（キーワードのリスト。PDF/A・PDF/X では不可）出力文書に対する利用権限のリスト。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いようにします。 ***nohiresprint*** （PDF 1.4）Acrobat で高解像度印刷ができないようにします。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・変更をできないようにします。 ***noassemble*** （PDF 1.4。nomodify を含む）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いようにします。 ***noannots*** Acrobat でしおり・フォームフィールドの追加・変更をできないようにします。 ***noforms*** （PDF 1.4。nomodify・noannots を含む）Acrobat でフォームフィールドへの記入をできないようにします。 ***nocopy*** Acrobat でテキスト・グラフィックのコピー・抽出をできないようにします。アクセシビリティインタフェースは noaccessible で制御します。 ***noaccessible*** （PDF 1.4）Acrobat でアクセシビリティ目的（読み上げプログラムなど）でのテキストやグラフィックの抽出ができないようにします。 ***plainmetadata*** （PDF 1.5）暗号化した文書に対しても、XMP 文書メタデータを暗号化しないままにします。
portfolio[1]	（オプションリスト。PDF 1.7）表 12.16 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
recordsize[2]	（整数。MVS のみ）出力ファイル、および linearize・optimize オプションで作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）
rolemap[2]	（対のリスト。各対の 1 番目の項目は名前文字列、2 番目の項目は文字列。タグ付き PDF に対してのみ）カスタム要素種別から標準要素種別へのマッピング。各対は、標準またはカスタムの要素種別の名前と、そのカスタム種別をマップさせたい標準要素種別の名前を内容として持ちます。既存の要素種別に対して別のセマンティクスを割り当てるために、標準要素種別を、別の種別へマップすることができます。タグ付き PDF におけるカスタム要素種別の使用については 239 ページ「14.3 タグ付き PDF」を参照してください。
search	（オプションリスト）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。使えるサブオプション： ***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。このインデックスのファイル名は文書に対する相対指定でもよいですが、正しいインデックスファイル名を与えるのはユーザー側の役割となります。 ***indextype*** （名前文字列）インデックスの種類で、Acrobat では PDX とする必要があります。デフォルト：PDX
tagged[2]	（論理値。PDF 1.4）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（239 ページ「14.3 タグ付き PDF」参照）。pdfa オプションの値が PDF/A-1a:2005 のときは、このオプションは自動的に true を強制されます。デフォルト：false

表 3.1 PDF_begin_document( )・PDF_end_document( ) の文書オプション一覧

オプション	説明
tempdirname[2]	（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作成したいディレクトリの名前。空にすると、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ときはこのオプションは無視されます。デフォルト：空
temp-filenames[2]	（文字列 2 個のリスト。MVS での PDF_begin_document( ) のみ）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：空
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：none
user-password[2, 3]	（文字列。PDF/A・PDF/X では不可）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。デフォルト：空
viewer-preferences	（オプションリスト）さまざまな表示設定を表 3.3 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( ) のみ
2. PDF_begin_document( )・PDF_begin_document_callback( ) のみ
3. パスワードでは、Winansi エンコーディング外のキャラクタは、PDF 1.7 拡張レベル 3 以上が生成されるときにのみ許されます。

表 3.2 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一覧

オプション	説明
group	（文字列。PDF_begin_document( ) のみ。文書がページグループを使っているときは必須、そうでないときは禁止）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を指定します。空文字列にすると unicode を意味します。デフォルト：グローバルな hypertextencoding パラメタの値。
pagenumber	（整数。PDF_end_document( ) のみ。文書がページグループを使っていないときは必須、そうでないときは禁止）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）範囲内のすべての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 3.3 PDF_begin_document( )・PDF_end_document( ) の viewerpreferences オプションのサブオプション一覧

オプション	説明
centerwindow	（論理値）文書のウィンドウを画面の中央に置くかどうかを指定します。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）Acrobat のタイトルバーに Title 文書情報フィールドを表示するか（true）、それともファイル名を表示するか（false）を指定します。デフォルト：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。
fitwindow	（論理値）文書のウィ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[1]	（論理値）Acrobat 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[1]	（論理値）Acrobat 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
hidewindow-ui[1]	（論理値）Acrobat のウィ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
nonfullscreen-pagemode	（キーワード。openmode オプションを fullscreen に設定しているとき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

表 3.3 PDF_begin_document( )・PDF_end_document( ) の viewerpreferences オプションのサブオプション一覧

オプション	説明
numcopies	（整数で範囲 1 ～ 5、PDF 1.7）印刷ダイアログの部数。デフォルト：ビューア依存
picktrayby-pdfsize	（論理値。PDF 1.7。Mac OS 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*** Acrobat で指定されているカレントの印刷の拡縮を使用します。
printpage-range	（整数対のリスト。PDF 1.7）印刷ダイアログのページ番号。各対は、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：ビューア依存
printarea ***printclip*** ***viewarea*** ***viewclip***	（キーワード。PDF/X では media・bleed のみ可）文書を画面表示・印刷する時に表示・断ち切りしたいページ領域を表すページ境界枠の種類。Acrobat はこの設定を無視しますが、他のアプリケーションでは有用かもしれません。使えるキーワード（デフォルト：crop）： ***art*** ArtBox を使用 ***bleed*** BleedBox を使用 ***crop*** CropBox を使用 ***media*** MediaBox を使用 ***trim*** TrimBox を使用

1. Acrobat 8 以上では、hidemenubar と hidetoolbar と hidewindowui の組み合わせ（すなわちすべてのユーザーインタフェース要素を隠す）には対応していません。これら 3 つの要素を hidden に設定してもメニューバーは依然表示されます。

3.2 PDF 文書をメモリから取得

空でない *filename* パラメタが *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* パラメタが空であれば、PDF 文書はメモリ内に生成されます。この場合、PDF 文書データはメモリから *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 サーバから頒布するときに有用です。

C++ ***const char *get_buffer(long *size)***
Java ***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得ます。

size （C・C++ 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位置への C スタイルのポインタ。

戻り値 クライアント側で消費するためのバイナリ PDF データで満たされたバッファ。返されるバイナリデータのデータ型は言語に依存します。返されたバッファは、クライアントで他の何らかの PDFlib 関数を呼び出す前に使う必要があります。他の PDFlib 関数をいろいろ呼び出しながらこのデータも使いたいときは、忘れずにデータをコピーしてください（特に、このデータの入った PVF ファイルを作成するために *PDF_create_pvf( )* を呼び出す前には）。

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。この関数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 をメモリ内に生成している場合は、この関数は *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。この関数を 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 出力にはバイナリキャラクタ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）。この関数は、*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、すなわちこの関数は *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* パラメタは 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* パラメタは与えてはいけません。

3.3 ページ関数

この節に関連するパラメタと値のキー名を表 3.4・表 3.5 に示します（19 ページ「2.2 パラメタ・オプション処理」参照）。

表 3.4 PDF_get/set_parameter( ) のページ関連のキー一覧

キー	説明
topdown	true なら、ページ・パターン・テンプレート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。詳しくは PDFlib チュートリアルを参照してください。スコープ：文書。デフォルト：false

表 3.5 PDF_get/set_value( ) のページ関連のキー一覧

キー	説明
pagewidth *pageheight*	カレントページの大きさ（MediaBox の寸法）を得ます。スコープ：ページ・パス

C++ Java ***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上書きできます（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 3.6 に示します。詳しくは表 3.7 も参照してください（*width・height* オプション）。

表 3.6 代表的な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・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 3.7 に従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。以下のオプションが使えます：*action・artbox・bleedbox・cropbox・defaultcmyk・defaultgray・defaultrgb・duration・group・height・label・mediabox・metadata・pagenumber・rotate・separationinfo・taborder・topdown・transition・transparencygroup・trimbox・userunit・viewports・width*。

詳細 この関数は、新規ページに対して、テキスト・グラフィック・色状態のパラメタを、すべてデフォルトにリセットします。

スコープ **文書**。この関数は**ページ**スコープを開始させます。対応する *PDF_end_page_ext()* と必ず対にして呼び出す必要があります。

C++ Java ***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 3.7 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：
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**。この関数は**ページ**スコープを終了させます。対応する *PDF_begin_page_ext()* と必ず対にして呼び出す必要があります。

表 3.7 PDF_begin_page_ext( )・PDF_end_page_ext( ) のオプション一覧

オプション	説明
action	（アクションリスト）以下のイベント（複数可）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。
artbox ***bleedbox*** ***cropbox***	（矩形）カレントページのページ枠のパラメタを変更します。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
defaultgray ***defaultrgb*** ***defaultcmyk***	（ICC ハンドル）与える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。
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
group[1]	（文字列。文書がページグループを使っているときは必須、そうでないときは禁止）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。

表 3.7 PDF_begin_page_ext( )・PDF_end_page_ext( ) のオプション一覧

オプション	説明
height	（float または キーワード。topdown オプションまたはパラメタが 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、同名のパラメタを上書きします。 以下のシンボリックな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 3.2 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	（矩形。topdown オプションまたはパラメタが 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 オプションおよびパラメタを上書きします。
metadata	（オプションリスト。PDF 1.4）ページにメタデータを与えます（237 ページ「14.2 XMP メタデータ」参照）。
pagenumber[1]	（整数）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）カレントページに対する色分版の詳細を入れたオプションリスト。これは Acrobat では無視されますが、サードパーティソフトウェアでは、分版ページを分版前工程において見分けて正しくプレビューするのに役立つかもしれません。 ***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。
taborder	（キーワード。PDF 1.5）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：none）。 ***column*** 注釈は段組みごとに上から下へ順に選ばれていき、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のと同じになります。 ***none*** タブ順序を指定しません。 ***structure*** 注釈は構造ツリー内での出現順に選ばれていきます。構造ツリーに入っていない注釈については順序は指定されません。 ***row*** 注釈は上の行から下の行へ順に選ばれていき、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のと同じになります。
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

表 3.7 PDF_begin_page_ext( )・PDF_end_page_ext( ) のオプション一覧

オプション	説明
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*** 矩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
transparency group	（オプションリスト。PDF 1.4）ページまたはテンプレートの透過グループ属性を指定します。使用可能なオプション： ***colorspace***（キーワード。必須）透過グループの色空間を、次のキーワードのいずれかで指定します：DeviceGray・DeviceRGB・DeviceCMYK。 ***isolated*** （論理値）透過グループが分離されているかを指定します。デフォルト：false ***knockout*** （論理値）透過グループが抜きグループかどうかを指定します。デフォルト：false デフォルト：複数ビットか PDF_create_gstate( ) の opacityfill/opacitystroke オプションの画像マスクがページ内にあるときは、出力品質の向上のため、次のオプションリストが自動的に作成されます：transparencygroup={colorspace=DeviceRGB}
trimbox	（矩形）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）ページ上の地理参照付き領域（ビューポート）（複数可）を指定します。詳しくは 232 ページ「13.2 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。
width	（float または キーワード。topdown オプションまたはパラメタが true なら禁止）先述の height オプションを参照してください。

1. PDF_begin_page_ext( ) のみ

C++ Java ***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（図形・色・テキストなど）・レイヤー状態が内部的に保存されます。これは後で *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**。この関数は**文書**スコープを開始させます。対応する *PDF_resume_page()* と必ず対にして呼び出す必要があります。この関数は、タグ付き PDF モードでは使ってはいけません。

C++ Java ***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 3.8 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：
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。

スコープ **文書**。この関数は**ページ**スコープを開始させます。対応する *PDF_suspend_page()* と必ず対にして呼び出す必要があります。

表 3.8 PDF_resume_page( ) のオプション一覧

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ときは必須、そうでないときは禁止）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。
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

クックブック　完全なコードサンプルがクックブックの graphics/starter_layer トピックにあります。

C++ Java　***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（要 PDF 1.5）。

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* 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。

optlist　レイヤー設定を表 3.9 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：

- 一般オプション群：*hypertextencoding*・*hypertextformat*（表 12.1 参照）
- レイヤー制御オプション群：
  creatorinfo・*defaultstate*・*initialexportstate*・*initialprintstate*・*initialviewstate*・*intent*・*language*・*onpanel*・*pageelement*・*printsubtype*・*removeunused*・*zoom*

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）、それは通常望むところではないでしょう。

PDF/X：レイヤーは PDF/X-1/2/3 では許されていません。PDF/X-4 または PDF/X-5 でレイヤーを使うには、いくつかのオプションが制限されており、また、少なくとも 2 個のレイヤーが *PDF_set_layer_dependency()* で指定されている必要があります。

スコープ　文書・ページ

表 3.9　PDF_define_layer( ) のオプション一覧

オプション	説明
creatorinfo	（オプションリスト。PDF/X 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
initial-exportstate	（論理値。PDF/X では不可）レイヤーの推奨書き出し状態を指定します。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/X では不可）レイヤーの推奨印刷状態。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

表 3.9 PDF_define_layer( ) のオプション一覧

オプション	説明
initial-viewstate	（論理値。PDF/X では不可）レイヤーの推奨表示状態。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/X では不可）レイヤーの言語を指定します： *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/X 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/X では不可）レイヤーがページネーションアーティファクトを含んでいることを示します。HF（ヘッダ / フッタ）・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/X 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：レイヤーが、listmode=visiblepages を用いて非デフォルトバリアントに含められていないかぎり、false。
zoom	（float かパーセント値のリスト。PDF/X 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 ***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)***

レイヤー間の階層・グループ・ロック条件を定義します（要 PDF 1.5）。

type　表 3.10 に従った、依存または関係の種別。

表 3.10 レイヤーの依存と関係の種別一覧

種別	説明。この種別に関連するオプション
GroupAllOn	（PDF/X では不可）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	（PDF/X では不可）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

表 3.10 レイヤーの依存と関係の種別一覧

種別	説明。この種別に関連するオプション
GroupAllOff	（PDF/X では不可）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	（PDF/X では不可）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、レイヤーの状態を 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	（PDF/X では不可）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。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	（PDF/X では不可）parent オプションで指定するレイヤーハンドルは、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親レイヤーノード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
Variant	文書バリアント、すなわち 1 個ないし複数のレイヤーの組み合わせを指定します。この後に PDF_set_layer_dependency( ) を呼び出せば、この設定に対する依存規則を指定するために variantname オプションをもう一度与えることができます。この種別に関連するオプション：basestate・defaultvariant・includelayers・invisiblelayers・visiblelayers

optlist レイヤー依存を表 3.11 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：*hypertextencoding*（表 12.1 参照）
- レイヤー依存オプション群：
  basestate・*children*・*createorderlist*・*defaultvariant*・*depend*・*includelayers*・*invisiblelayers*・*group*・*visiblelayers*・*listmode*・*parent*・*variantname*。

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。

バリアントは、生産安全性を高めるための、複数レイヤーの固定された組み合わせととらえることができます。個々のレイヤーをユーザーに操作させるのではなく、1 つのバリアントとして、表示・非表示を切り替えさせることができます。文書がバリアントを含むとき、Acrobat 9 は個々のレイヤー名を表示せず、バリアントの名前のみを表示します。レイヤーバリアントは、Acrobat 9 で PDF/X 文書に対してのみ表示されます。Acrobat X はレイヤーバリアントを表示しません。

影響を受けるすべてのレイヤーが同一のバリアントの一部であるわけではないレイヤーバリアントが存在する場合において依存を指定するには、デフォルトバリアントを設定する前にその依存を設定する必要があります。

PDF/X：レイヤーは PDF/X-1/2/3 では許されていません。PDF/X-4 または PDF/X-5 でレイヤーを使うには、少なくとも 2 個のレイヤーバリアント（すなわち *type=Variant*）を指定する必要があります。*type* のいくつかの値は PDF/X では制限されています。

スコープ　**文書・ページ**。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。

表 3.11 PDF_set_layer_dependency( ) のオプション一覧

オプション	説明
basestate	（キーワード。type=Variant のみ。PDF/X では不可）visiblelayers・invisiblelayers オプションで明示的に設定されていないすべてのレイヤーの表示・非表示を指定します。使えるキーワード（デフォルト：on）： ***on***　選択されたバリアントに対してすべてのレイヤーが表示されます。 ***off***　選択されたバリアントに対してすべてのレイヤーが非表示になります。 **unchanged**　選択されたバリアントに対してすべてのレイヤーの状態が変更されず保持されます。
children	（レイヤーハンドルのリスト。type=Parent・Title のみ）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レイヤーハンドル（複数可）。
createorder-list	（論理値。type=Variant かつ defaultvariant=true の場合のみ）ユーザーインタフェースですべてのレイヤーを表すために用いることのできる /Order 配列の中へすべてのレイヤーを含めます。このオプションを true に設定すると下記の影響があります： ▸ Acrobat 9 は、レイヤーバリアント群を、レイヤー名群の中ではなく「レイヤー」パネル内に表示します。Acrobat 9 は、createorderlist による文書に対しては PDF/X-4 検証エラーを出します。なぜならこのオプションは PDF/X-4:2008 では許されていないからです。 ▸ Acrobat X は、レイヤー名群を、レイヤーバリアント群の中ではなく「レイヤー」パネル内に表示します。Acrobat X は、createorderlist による文書の検証を成功させます。なぜならこのオプションは PDF/X-4:2010 では許されているからです。 デフォルト：PDF/X では false、それ以外では true
default-variant	（論理値。type=Variant のみ）true にすると、指定されたバリアントはデフォルトバリアントになります。すなわち、文書が開かれた時に表示されます。ちょうど 1 個のバリアントをデフォルトバリアントに指定する必要があります。デフォルト：false
depend	（レイヤーハンドル。type=GroupAllOn・GroupAnyOn・GroupAllOff・GroupAnyOff のみ）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・Radiobtn のみ）グループを構成させたいレイヤーハンドル（複数可）。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。
includelayers	（レイヤーハンドルのリスト。type=Variant のみ）バリアントに属するレイヤー群を指定します。デフォルト：文書ないでそれまでに定義されたすべてのレイヤー
invisiblelayers	（レイヤーハンドルのリスト。type=Variant のみ）選択されたバリアントに対して初期状態で非表示になるレイヤー群のリストを指定します。1 つのレイヤーをバリアントの visiblelayers リストと invisiblelayers リストに同時にリストしてはいけません。defaultvariant=true のとき、このオプションは PDF_define_layer( ) の defaultstate オプションを上書きします。デフォルト（basestate オプションに依存）：basestate=off ならば includelayers リスト内のすべてのレイヤー、basestate=on ならば空リスト。
listmode	（キーワード。type=Variant のみ）どのレイヤー名群が Acrobat のレイヤーペーンに表示されるかを指定します。使えるキーワード（デフォルト：visiblepages）： ***allpages***　すべてのページのすべてのレイヤーの名前が表示されます。 **visiblepages**　その時表示されているページのすべてのレイヤーの名前が表示されます。これは、そのバリアントに属するすべてのレイヤーに対して removeunused=true を暗黙に前提します。 Acrobat ではこれは、defaultvariant=true のときにのみ効果を持ちます。

表 3.11 PDF_set_layer_dependency( ) のオプション一覧

オプション	説明
parent	（レイヤーハンドル。type=Parent・Title のみ）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。
variantname	（ハイパーテキスト文字列。type=Variant では必須）選択されたバリアントの名前。type=Variant のときは、各バリアントの名前を 1 回だけ指定する必要があります。type が Variant 以外の場合のデフォルト：デフォルトのバリアント
visiblelayers	（レイヤーハンドルのリスト。type=Variant のみ）選択されたバリアントで初期状態で表示されるレイヤー群のリストを指定します。1 つのレイヤーをバリアントの visiblelayers リストと invisiblelayers リストに同時にリストしてはいけません。defaultvariant=true のとき、このオプションは PDF_define_layer( ) の defaultstate オプションを上書きします。デフォルト（basestate オプションに依存）：basestate=on ならば includelayers リスト内のすべてのレイヤー、basestate=off ならば空リスト。

C++ Java ***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（要 PDF 1.5）。

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。

この関数は、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・テンプレート・フォームフィールドのレイヤーは、それぞれの関数で *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

C++ Java ***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（要 PDF 1.5）。

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

4 フォント・テキスト関数

4.1 フォント処理

この節に関連するパラメタと値のキー名を表 4.1 に示します（19 ページ「2.2 パラメタ・オプション処理」参照）。

表 4.1 PDF_get/set_parameter( ) のフォント関連のキー一覧

キー・説明
Encoding・FontAFM・FontPFM・FontOutline・HostFont それぞれ、UPR ファイルで各カテゴリに書くのと同じリソースファイル行。複数回呼び出すと、内部のリストに新しい項目が追加されます。表 2.3 の resourcefile も参照。スコープ：任意

C++ Java ***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、このパラメタを上書きする *fontname* オプションで与えることもできます。詳しくは表 4.3 の *fontname* オプションを参照。

len （C 言語バインディングのみ）*filename* が UTF-16 文字列のときの長さをバイト単位で指定します。*len=o* 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、このパラメタを上書きする *encoding* オプションで与えることもできます。詳しくは表 4.3 の *encoding* オプションを参照。エンコーディング関連では以下のような問題がよく起こりますので留意してください：

- 8 ビットエンコーディングを与えたが、フォントがこのエンコーディングに対するグリフを一切持っていない、またはフォントが標準日中韓フォントである。
- *builtin* エンコーディングを与えたが、フォントが内蔵エンコーディングを一切持っていない。これは TrueType フォントについてのみ起こりえます。
- 定義済み CMap を与えたが、フォントに一致しない。

optlist 下記のオプション群によるオプションリスト：

- 一般オプション：*errorpolicy*（27 ページ「2.5 例外処理」参照）
- 表 4.3 に従ったフォント読み込みオプション群：
  ascender・autocidfont・autosubsetting・capheight・descender・dropcorewidths・embedding・encoding・fallbackfonts・fontname・fontstyle・initialsubset・keepfont・keepnative・linegap・metadata・monospace・optimizeinvisible・readfeatures・readkerning・readshaping・replacementchar・skipposttable・subsetlimit・subsetminsize・subsetting・unicodemap・vertical・xheight

戻り値 フォントハンドル。以後の *PDF_info_font()* とテキスト出力関数と、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（たとえばフォント・メトリック・エンコーディングファイルが見つからないか、あるいは組み合わせの誤りが検知された）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* パラメタまたはオプションで変えることができます。

関数が -1（PHP では 0）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、この関数によって返された値を、他のフォント関連の関数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()* で閉じると、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、ただし対応する *PDF_load_font()* への呼び出しで *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 この関数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：この関数を、同じフォント名・同じエンコーディング・同じオプション群で再度呼び出すと、最初の呼び出しと同じフォントハンドルが返されます。例外：下記のオプションのいずれかを最初の呼び出しで指定し、その後の呼び出しでは指定しなかった場合も、2 度目のフォントハンドルは最初のフォントハンドルと同一になります：*embedding*・*readkerning*・*replacementchar*・*fallbackfonts*・*metadata*。同様に、*initialsubset* オプションはフォントの比較にあたっては無視されます。たとえば、フォントを *initialsubset* なしで読み込んだ後、*initialsubset* をつけて再度読み込んだ場合、最初のフォントへのハンドルが返され、*initialsubset* は何ら効力を持ちません。

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。この状況はたいてい、アプリケーション内の問題を指し示します。

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 関数のなかには、オプションリストでフォント名とエンコーディング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。

テキスト出力機能のなかには、特定のエンコーディングでは利用できないものがあります（表 4.2 参照）。

表 4.2 各種エンコーディングでの PDFlib 諸機能の利用可能性一覧

機能	unicode・Unicode CMap	8 ビットエンコーディング	レガシ CMap・cp936 等	glyphid
テキストフロー	可	可	可[1]	可
グリフ置き換え	可[2]	可	可[1]	—
予備フォント	可[2]	可	可[1]	—
シェーピング	可[2]	—	可[1]	可
OpenType レイアウト機能	可	—	可[1]	可

1. この機能は、日中韓フォントに対しては、keepnative=true では利用できません。

2. この機能は、Unicode CMap または keepnative=true による標準日中韓フォントに対しては利用できません。

スコープ　任意

パラメタ　表 4.1 参照。

表 4.3 PDF_load_font( ) と暗黙的フォント読み込みのフォント読み込みオプション一覧

オプション	説明
ascender	(−2048 から 2048 までの整数) アセンダ。同名のタイポグラフィ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上書きされますが、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)
autocidfont	(論理値) true にすると、winansi・macroman・builtin 以外の 8 ビットエンコーディングによる TrueType フォントと、グリフ名のない OpenType フォントは、自動的に CID フォントとして格納されます。これは、winansi エンコーディング外のグリフが得られない問題を回避します。デフォルト：true
auto-subsetting	(論理値) フォントをサブセット化するかどうかを、subsetlimit・subsetminsize オプションと、グリフの実際の使われ方に従って動的に決定します。subsetting オプションを与えているときはこのオプションは無視されます。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。
descender	(−2048 から 2048 までの整数) ディセンダ。上記 ascender 参照。
dropcore-widths	(論理値。非サポート。PDF/A・PDF/X モードでは false を強制されます) 埋め込んでいないコアフォントの幅を、生成する PDF に埋め込みません。出力ファイルサイズが少し小さくなりますが、キャラクタによっては不正確なテキスト印字が生成されるおそれがあります。このオプションはデフォルト値のままにしておくことを強く推奨します。デフォルト：false
embedding	(論理値。PDF/A・PDF/X では true にする必要があります。つねに埋め込まれる SING フォント・Type 3 フォントに対しては無視されます) フォントを埋め込むかどうかを制御します。フォントを埋め込むためには、そのフォントのアウトラインファイルが得られる必要があり、さらにそのメトリック情報も必要で (TrueType・OpenType フォントを除き)、それらが得られれば、実際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、そのフォントの一般的情報だけが PDF 出力へ書き込まれます。 デフォルト：通常は false ですが、CID フォントへの変換を引き起こすエンコーディングを持つ TrueType・OpenType フォントの関与する特定の状況においては true。PDFlib はそのようなフォントを自動的に埋め込みますが、embedding を false に設定すると、フォントを埋め込ませないことができます。この場合は、PDF 文書を表示・印刷するシステムに、そのフォントがインストールされている必要があります。 オプション embedding=false は、それ以前に同じフォントをすでに embedding=true で読み込んいるときは無視されます。不可視テキストでのフォントに対する埋め込み動作は、embedding=true の場合であっても、optimizeinvisible オプションで変更することが可能です。

表 4.3 PDF_load_font( ) と暗黙的フォント読み込みのフォント読み込みオプション一覧

オプション	説明
encoding	（文字列。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 が指定されていないときは、PDF_fill_textblock( ) の場合以外には必須） このフォントに対して入ってくるテキストに期待する円コーディング（大文字・小文字を区別します）： ワイドキャラクタエンコーディング： ▸ unicode および Unicode CMap の名前 ▸ 非 Unicode（レガシ） CMap の名前、あるいは CID 指定の Identity-H か Identity-V ▸ glyphid：フォント内のすべてのグリフを、そのフォント独自の ID で指定できます バイトエンコーディング・マルチバイトエンコーディング： ▸ 定義済み 8 ビットエンコーディング winansi・macroman・macroman_apple・ebcdic・ebcdic_37・pdfdoc・iso8859-X・cpXXXX のうちの 1 つ、および非 Unicode CMap ▸ 日中韓コードページの cp932・cp936・cp949・cp950 のいずれか（Windows の場合、システムコードページが用いられます。それ以外のすべてのシステムでは、対応する CMap が利用可能である必要があります） ▸ 自動的に選択されるエンコーディングの host または auto ▸ ファイルから読み込まれる、または PDF_encoding_set_char( ) で定義されるユーザー定義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（主に記号フォントで使用） ▸ オペレーティングシステムが知っているエンコーディング名（すべてのプラットフォームで利用できるわけではありません） 非 Unicode 言語バインディングでは、textformat=auto は以下のように動作します（どちらの場合にもすべての UTF 形式が許されることに留意してください）： ▸ ワイドキャラクタエンコーディング：読み込むフォントのテキストはテキスト形式 utf16 で扱われます（encoding=glyphid の場合、サロゲートは解釈されません） ▸ バイトエンコーディング・マルチバイトエンコーディング：読み込むフォントのテキストはテキスト形式 bytes で扱われます。 PDF_load_font( )：このオプションはあるいは、関数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。
fallbackfonts	（表 4.4 に従ったオプションリスト群のリスト） 読み込むフォントに対する予備フォント（複数可）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。予備フォントは、フォントの種別とエンコーディングの組み合わせによっては対応していません（表 4.2 参照）。 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。

表 4.3 PDF_load_font( ) と暗黙的フォント読み込みのフォント読み込みオプション一覧

オプション	説明
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 が指定されていないときは必須）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検索されます。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「@」キャラクタの場合、そのフォントには縦書きが適用されます。 PDF_load_font( ) : あるいは関数の引数として与えることもできます。
fontstyle	（キーワード）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。埋め込まれていない TrueType（TTC でない）・OpenType フォントの場合は、擬似フォントスタイルは Acrobat によって作成され、それ以外の場合は PDFlib によって作成されます（fakebold パラメタまたはオプションと同じ太字化方式を使って）。後者の場合は、傾きの角度を italicangle パラメタまたはオプションで制御できます。このオプションをコアフォントのいずれかに適用すると、擬似フォントスタイルは作成されず、適切なボールド・イタリック・ボールドイタリックのフォントバリエーションが選ばれます。そのようなフォントが得られないときは（Times-Bold に bold を適用した等）、このオプションは無視されます。デフォルト : normal
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、PDF 内のフォント容量を削減しつつ、同等のサブセットを生成させることができ、ひいては複数の文書を連結する際の最適化も容易になります。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。Unichar・Unicode 範囲はエンコーディング名に優先します。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。サブセットの内容は文書内のテキストによって決定されます。 **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で見つかったすべての Unicode 値が初期サブセットへ含まれます。追加のキャラクタに対するグリフも、文書内のテキストによって、または features・shaping テキストオプションによって必要となれば、自動的にサブセットへ追加されます。
keepfont	（論理値。Type 3 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
keepnative	（論理値。非 Unicode CMap による日中韓フォントでのみ意味を持ち、それ以外のフォントでは無視されます。embedding=true にしているときは false を強制されます）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）。これは、選んだ CMap（Shift-JIS 等）にまだ一致していなければならない API 関数に与えるテキストに対しては効力を持ちません。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関数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）。Unicode CMap によるフォントは、グリフの置き換えと予備フォントが利用できない点以外は、encoding=unicode と同様に動作します。 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関数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 フォントまたは embedding=true の場合は false、それ以外の場合は true。
kerning	非推奨。フォント内のカーニングデータの解析を制御するには readkerning オプションを用いてください。
linegap	（-2048 から 2048 までの整数）行間。上記 ascender 参照。

表 4.3 PDF_load_font( ) と暗黙的フォント読み込みのフォント読み込みオプション一覧

オプション	説明
metadata	（オプションリスト。PDF 1.4）フォントのためのメタデータを与えます（237 ページ「14.2 XMP メタデータ」参照）
monospace	（1 から 2048 までの整数。PDF/A では不可）フォント内のすべてのグリフを強制的に指定幅（フォント座標系で表したもの。1000 単位が文字サイズに等しい）にします。Type 3 フォントの場合は、0 でないグリフ幅がすべて変更されます。このオプションは、標準日中韓フォントに対してのみ推奨され、また、コアフォントでは使えません。フォントを埋め込むと、このオプションは無視されます。デフォルト：値なし（フォント内のメトリックが使われます）
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。フォントが埋め込まれなくても、フォントファイルは通常どおり設定する必要があります。なぜなら PDFlib が埋め込み省略を決定するのは文書の最後になってからだからです。デフォルト：false
readfeatures	（論理値。TrueType・OpenType フォントに対して、かつ encoding=unicode・glyphid・Unicode CMap のいずれかのときのみ意味を持ちます）TrueType または OpenType フォントの機能テーブルをフォントから読み込むかどうかを指定します。OpenType 機能のテキストへの実際の適用はテキストオプション features によって制御されます（表 5.3 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 5.3 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：true
readshaping	（論理値。TrueType・OpenType フォントに対して、かつ unicode・glyphid エンコーディングに対してのみ意味を持ちます）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テキストオプション shaping によって制御されます（表 5.3 参照）。シェーピ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、メモリを節約することができます。デフォルト：true
replace-mentchar	（Unichar。glyphcheck=replace にするときのみ意味を持ちます。非 Unicode CMap または glyphid エンコーディングで読み込むフォントに対しては無視されます）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的に類似のキャラクタ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。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が、ただしこれは PDF/A-1・PDF/X-4・PDF/X-5 標準では許されていません。encoding=builtin で読み込んだ記号フォントに対しては、Unicode 値でなくコードを与える必要があります。 デフォルト：フォントにもしあれば U+00A0（NO-BREAK SPACE）、なければ、フォントにもしあれば U+0020（SPACE）、なければ、U+0000（ただし PDF/A-1・PDF/X-4・PDF/X-5 の場合を除く）。これらの値は、指定した replacementchar がフォントの中にないときにも用いられます。
skippost-table	（論理値。非サポート。TrueType・OpenType フォントに対してのみ意味を持ちます）TrueType・OpenType フォントの post テーブルを解析してグリフ名を決定するかどうかを指定します。このオプションを true 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が、非標準的な名前を持つグリフへのグリフ名参照はそのフォントに対しては働かなくなります（これは主に記号フォントについて起こりうることで、テキストフォントでは通常起こりません）。デフォルト：false
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が無効になります。デフォルト：100%

表 4.3 PDF_load_font( ) と暗黙的フォント読み込みのフォント読み込みオプション一覧

オプション	説明
subsetminsize	(float。Type 3 フォントでは無視されます) 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、自動フォントサブセット化が無効になります。デフォルト : 50
subsetting	(論理値) フォントをサブセット化するかどうかを制御します。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 (PDFlib チュートリアル参照)、PDF_load_font( ) への 1 回目の呼び出しで subsetting オプションを与える必要があります。デフォルト : false
unicodemap	(論理値。pdfa=PDF/A-1a:2005 なら false に設定する必要があります) 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。定義済み CMap を指定しているときは無視されます。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ます。
xheight	(−2048 から 2048 までの整数) x ハイト。上記 ascender 参照。

表 4.4 PDF_load_font( ) の fallbackfonts オプションのサブオプション一覧

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 オプションでなく fontname・encoding オプションで) 指定したときは、表 4.3 に従った fallbackfonts 以外の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。非 Unicode CMap で読み込んだフォントは、予備フォントとして用いることはできません。
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)。フォントは、embedding=false かつ keepnative=true による標準日中韓フォントであってはいけません。
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) キャラクタ群を指定します。予備フォントは、指定したキャラクタに対するグリフを含んでいるか (個々のキャラクタを指定した場合)、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。
textrise	(float またはパーセント値) テキストライズ値 (表 5.2 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 ***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 この関数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 ***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 4.6 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*・*metricsfile*・*outlineformat*・*singfont*・*standardfont*・*supplement*
- フォントの技術的情報：*feature*・*featurelist*・*fontstyle*・*hostfont*・*kerningpairs*・*monospace*・*numglyphs*・*shapingsupport*・*vertical*
- フォントとエンコーディングの関係：*codepage*・*codepagelist*・*encoding*・*fallbackfont*・*keepnative*・*maxcode*・*numcids*・*numusableglyphs*・*predefcmap*・*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*

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 4.6 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 4.5 に従ったマッピングオプション：*cid*・*code*・*glyphid*・*glyphname*・*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 4.5 PDF_info_font( ) でグリフを指定するオプション一覧

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
unicode	（Unichar）Unicode キャラクタ

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 が返されます。いくつかのキーワードは文字列を、その文字列番号を返すことによって間接的に返します。その対応する文字列は、*PDF_get_parameter( )* で *string* パラメタを使って取得できます（表 2.3 参照）。

詳細 この関数は、以下の異なる情報源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 オブジェクト以外の任意

表 4.6 PDF_info_font( ) のキーワード・オプション一覧

キーワード	説明・オプション
ascender	アセンダのメトリック値。使えるオプション（デフォルト：fontsize=1000）： *faked*　（論理値）値をフォントまたはメトリック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
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
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

表 4.6 PDF_info_font( ) のキーワード・オプション一覧

キーワード	説明・オプション
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 の形で指定します。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。
descender	ディセンダのメトリック値。ascender 参照。
encoding	フォントのエンコーディングまたは CMap 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	フォントファミリの名前の文字列番号、または得られないなら –1
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*** （文字列。script を与えているときのみ）言語の名前を指定します。 ***name*** （文字列。必須）OpenType 機能テーブルの 4 文字の名前を指定します。例：liga（標準合字）・ital（日中韓フォントの斜体）・vert（縦書き）。機能 kern については取得できません。 ***script*** （文字列）用字系の名前を指定します。 以下の戻り値で、指定された OpenType 機能テーブル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全く得られません。 ***0*** その機能テーブルが得られないか、または PDFlib がそれに対応していません。 ***1*** 指定された用字系と言語に対してその機能テーブルが得られ、かつ PDFlib がそれに対応しています。
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）： ***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 ***acrobat*** （論理値）true にすると、Acrobat で表示されるのと同じフォント名

表 4.6 PDF_info_font( ) のキーワード・オプション一覧

キーワード	説明・オプション
fontstyle	fontstyle オプションの値（normal・bold・italic・bolditalic）の文字列番号。使えるオプション： ***faked*** フォントスタイルが PDFlib によって実現されるときは 1、フォントスタイルが Acrobat によって実現されるときは 0
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 Multiple Master・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。 ***code*** （範囲 0 ～ 255 の数値。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のみ）エンコーディングスロット ***unicode*** （Unichar。encoding= unicode によるフォントのみ）指定する Unicode キャラクタの入ったスロット
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	keepnative オプションの出力値。
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 mainGID）。
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 に対しては 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。
metricsfile	フォントのメトリックファイル（AFM または PFM）のパス名の文字列番号か、または得られないときは –1
monospace	monospace オプションの値、またはそれが与えられていないときは 0。
numcids	フォントが標準 CMap を使っている場合は CID の数、そうでないなら –1
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・LWFN・TTF・OTF・TTC のいずれか
predefcmap	フォントに対するエンコーディングとして指定された定義済み CMap の名前の文字列番号、または得られないなら –1。

表 4.6 PDF_info_font( ) のキーワード・オプション一覧

キーワード	説明・オプション
replacement char	replacementchar オプションで指定されたキャラクタの 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。
shaping-support	フォントがシェーピングに対応していて、かつ PDF_load_font( ) で readshaping オプションが与えられていたなら 1、そうでないなら 0
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 コアフォントか標準日中韓フォントのときは 1、そうでないときは 0
supplement	標準日中韓 CMap によるフォントのときはキャラクタ集合の補足数、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 cid・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
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。非 Unicode CMap を持つ日中韓フォントで keepnative=true のときは 0 を返します。
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
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（embedding オプションか、または強制フォント埋め込みで）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（autosubsetting=true なら、サブセット化が有効になるためには、subsetlimit を超えないことが必要です）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 Type 3 フォントの定義

クックブック　完全なコードサンプルがクックブックの fonts/starter_type3font トピックにあります。

C++ Java　***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のときのフォント定義の 2 度目の呼び出しでは無視されます）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。6 個の値は、PostScript と 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* は *b* × *c* と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　（サブセットフォントのときの 2 度目の呼び出しでは無視されます）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・familyname・stretch・weight・widthsonly*

詳細　この関数は、テキスト・グラフィック・色状態のパラメタを、すべてデフォルトにリセットします。フォントには任意の数のグリフを入れられます。フォントは、カレントの**文書**スコープが終わるまで使えます。

スコープ　**文書・ページ**。この関数は**フォント**スコープを開始させます。対応する *PDF_end_font()* と必ず対にして呼び出す必要があります。サブセット化されるフォントの 2 度目の呼び出しについては、**文書**スコープのみ許されます。

表 4.7　PDF_begin_font( ) のオプション一覧

オプション	説明
colorized	（論理値）true にすると、フォントはキャラクタごとにその色を明示的に指定することができます。false にすると、キャラクタはすべてカレント色（フォントを定義した時点ではなく使う時点の）で描かれるので、グリフ定義に色オペレータまたはマスク以外の画像を入れてはいけません。デフォルト : false
familyname[1]	（文字列。PDF 1.5）フォントファミリの名前
stretch[1]	（キーワード。PDF 1.5）フォントの幅の値。キーワード : ultracondensed・extracondensed・condensed・semicondensed・normal・semiexpanded・expanded・extraexpanded・ultraexpanded。デフォルト : normal
weight[1]	（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

表 4.7 PDF_begin_font( ) のオプション一覧

オプション	説明
widthsonly	（論理値）true にすると（Type 3 フォントのサブセット化の 1 回目の呼び出し）、フォントとグリフのメトリックだけが定義されます。PDF_begin_glyph( ) と PDF_end_glyph( ) の間では他の API 関数を一切呼び出すべきではありません。他の関数をたとえ呼び出しても、PDF 出力にはいかなる効果も与えず、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 2 回目の呼び出し）、実際のグリフのアウトラインを定義できます。このように 2 回の呼び出しによる定義により、PDFlib で Type 3 フォントのサブセット化を実行することが可能になります。デフォルト：false

1. これらのオプションは、タグ付き PDF を作成しているときには強く推奨しますが、それ以外のときは無視されます。

C++ Java ***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**。この関数は**フォント**スコープを終了させます。対応する *PDF_begin_font( )* と必ず対にして呼び出す必要があります。

C++ Java ***void begin_glyph(String glyphname, double wx, double llx, double lly, double urx, double ury)***
Perl PHP ***begin_glyph(string glyphname, float wx, float llx, float lly, float urx, float ury)***
C ***void PDF_begin_glyph(PDF *p,***
const char *glyphname, double wx, double llx, double lly, double urx, double ury)

Type 3 フォントのためのグリフ定義を開始させます。

glyphname グリフの名前。この名前は、フォントとともに使われるあらゆるエンコーディングで使われる必要があります。Adobe グリフリスト（AGL）に従ったグリフ名を使うことを強く推奨します。グリフ名はフォント内で一意である必要があります。

wx （Type 3 フォントのサブセットのときのフォント定義の 2 度目の呼び出しでは無視されます）グリフの幅を、フォントの行列で指定したグリフ座標系で指定します。

llx・lly・urx・ury （Type 3 フォントのサブセットのときのフォント定義の 2 度目の呼び出しでは無視されます）フォントの *colorized* オプションを *false* にしているときは（デフォルトではそうなっています）、グリフの外接枠の左下隅と右上隅の座標。この外接枠の値は、PostScript 印刷で問題が起きないよう、正しくする必要があります。フォントの *colorized* オプションを *true* にしているときは、4 つの値すべてを 0 にする必要があります。

詳細 フォント内のグリフは、テキスト・グラフィック・画像関数を使って定義することができます。ただし画像は、フォントの *colorized* オプションを *true* にしているか、または画像を *mask* オプションを指定して開いているときにしか使えません。Type 3 フォントの中でビットマップを定義したいときは、インライン画像機能を使うことを強く推奨します。

colorized オプションを *true* にしているときは、今いるページのグラフィック状態がまるごとグリフ定義へ継承されますので、グリフ定義に関係する種類のグラフィック状態はすべて（*linewidth* 等）、グリフ定義の中で明示的に設定する必要があります。

PDFlib は内蔵のリストの中でグリフ名を検索して各グリフの Unicode 値を決定します。グリフ名が見つからなかったときは、PUA Unicode 値がそのグリフ名に割り当てられます（U+E000 から始まる）。この値は *PDF_info_font()* で取得できます。

スコープ **ページ・フォント**。この関数は**グリフ**スコープを開始させます。対応する *PDF_end_glyph()* と必ず対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()* と *PDF_end_glyph()* の間の API 関数の呼び出しはすべて無視されます。

C++ Java ***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**。この関数は**グリフ**スコープを終了させます。対応する *PDF_begin_glyph()* と必ず対にして呼び出す必要があります。

4.3 エンコーディングの定義

C++ Java ***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。このエンコーディング名は、あらゆる組み込みエンコーディングと、以前に使ったあらゆるエンコーディングと異なっている必要があります。

slot 定義したいキャラクタの位置を *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 キャラクタの名前。

uv キャラクタの Unicode 値。

詳細 この関数は、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、キャラクタを追加していくことができますが、そうでなければ例外が発生します。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。

グリフ名と Unicode 値との組み合わせは 3 通りが可能です。

- ***glyphname* を与え、*uv=0*** : これは、Unicode 値を持たないエンコーディングファイルと同等です。
- ***uv* を与えるが、*glyphname* を与えない** : **これは、コードページファイルと同等です。**
- ***glyphname* と *uv* を与える** : これは、Unicode 値を持つエンコーディングファイルと同等です。

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）。

エンコーディングを Type 3 フォントで使うことを意図している場合は、エンコーディングスロット群をグリフ名だけで指定することを推奨します。

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

4.4 単純テキスト出力

注 この節の関数に与えるテキストはすべて、*PDF_load_font()* で選んでいるエンコーディングと、指定している *textformat* に合っていなければなりません。Acrobat での制約により、テキスト文字列の長さは 32 KB を超えてはいけません。

この節に関連するパラメタと値を表 4.8・表 4.9 に示します（19 ページ「2.2 パラメタ・オプション処理」参照）。

表 4.8 PDF_get/set_parameter( ) のテキスト関連のキー一覧

キー	説明
autospace	true なら、カレントフォントが U+0020 に対するグリフを含んでいるときは、PDFlib は show 操作で生成する各テキスト出力の後に、スペースキャラクタが自動的に追加されます。これはタグ付き PDF の生成の際に便利でしょう。ただしスペースを追加すると、show 操作後のカレントテキスト位置が変わります。デフォルト：false。スコープ：任意
charref	表 5.1 参照。同名のオプションと異なり、このパラメタは、ハイパーテキスト文字列と名前文字列に対しても効力を持ちます。デフォルト：false。スコープ：任意
decoration-above	表 5.2 参照。デフォルト：false。スコープ：任意
escape-sequence	表 5.1 参照。同名のオプションと異なり、このパラメタは、ハイパーテキスト文字列と名前文字列に対しても効力を持ちます。たとえば Windows UNC ファイル名は、escapesequence=true のときは 4 個のバックスラッシュキャラクタで始まる必要があります。注意：このパラメタは環境変数（例：PDFLIBLICENSEFILE）に対しても効力を持ちます。Windows の環境変数の中のパス名は通常バックスラッシュキャラクタを含みますので、このパラメタは Windows システムでは使用を避け、同名のオプションを都度関数に与えるほうがよいでしょう。デフォルト：false。スコープ：任意
fakebold	表 5.2 参照。デフォルト：false。スコープ：ページ・パターン・テンプレート・グリフ・文書
glyphcheck	表 5.1 参照。デフォルト：replace。スコープ：任意
kerning	表 5.2 参照。デフォルト：true。スコープ：任意
textformat	（Unicode 非対応言語バインディングのみ）テキスト出力関数がクライアントの与える文字列に前提する形式。使えるキーワードは bytes・utf8・ebcdicutf8（iSeries・zSeries のみ）・utf16・utf16le・utf16be・auto。デフォルト：auto。スコープ：任意
underline overline strikeout	表 5.2 参照。デフォルト：false。スコープ：ページ・パターン・テンプレート・グリフ・文書

表 4.9 PDF_get/set_value( ) のテキスト関連のキー一覧

キー	説明
charspacing	表 5.2 参照。float 値にのみ対応しています。スコープ：ページ・パターン・テンプレート・グリフ・文書
font[1]	PDF_setfont( ) で設定しているカレントフォントの識別子か、またはフォントを設定していないなら −1（PHP では 0）。スコープ：ページ・パターン・テンプレート・グリフ
fontsize[1]	カレントフォントのサイズ。PDF_setfont( ) であらかじめ設定しておく必要があります。スコープ：ページ・パターン・テンプレート・グリフ
horizscaling	表 5.2 参照。float 値にのみ対応しています。スコープ：ページ・パターン・テンプレート・グリフ・文書

表 4.9 PDF_get/set_value( ) のテキスト関連のキー一覧

キー	説明
italicangle	表 5.2 参照。スコープ：ページ・パターン・テンプレート・グリフ・文書
leading	行送り、すなわちテキストの隣りあう行のベースラインの間隔。行送りは、PDF_continue_text( ) に使われます。PDF_setfont( ) で新しいフォントを選んだ時は、文字サイズと同じ値に設定されます。行送りを文字サイズに等しく設定すると、行間が密になります（leading=0 にすると、行が重なります）。しかし、隣りあう行のアセンダとディセンダはふつう重なりあいません。スコープ：ページ・パターン・テンプレート・グリフ
strokewidth	表 5.2 参照。float 値にのみ対応しています。値 0 で内蔵のデフォルトを指定します。デフォルト：0
textrendering	表 5.2 参照。スコープ：ページ・パターン・テンプレート・グリフ・文書
textrise	表 5.2 参照。float 値にのみ対応しています。スコープ：ページ・パターン・テンプレート・グリフ
textx[1] *texty*[1]	カレントテキスト位置の x・y 座標。デフォルト：0。スコープ：ページ・パターン・テンプレート・グリフ
underline-position	表 5.2 参照。文字サイズの分数を指定する float 値にのみ対応しています。値 1000000 を指定すると、フォントのメトリックまたはアウトラインファイルから取得されるフォント固有の値を設定することができます。デフォルト：1000000
underline-width	表 5.2 参照。float 値にのみ対応しています。値 0 を指定すると、フォントのメトリックまたはアウトラインファイルが得られるときはその中のフォント固有の値を、得られないなら文字サイズの 5% を用います。デフォルト：0
wordspacing	表 5.2 参照。float 値にのみ対応しています。スコープ：ページ・パターン・テンプレート・グリフ・文書

1. PDF_get_value( ) のみ

C++ Java ***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 この関数は、*PDF_show( )* 等の低レベルテキスト出力関数で用いられるフォントを設定します。フォントは、ページごとに、いかなる単純テキスト出力関数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。ページの中でカレントフォントは何度でも変更可能です。

スコープ ページ・パターン・テンプレート・グリフ

パラメタ この関数は自動的に *leading* パラメタを *fontsize* と同じ値に設定します。

C++ Java ***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 設定したいカレントテキスト位置。

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**

パラメタ 表 4.8・表 4.9 参照。

C++ Java ***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 キャラクタを含んでいてはいけません。null キャラクタ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 （*PDF_show2( )* 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。

詳細 フォントはあらかじめ、*PDF_setfont( )* 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。

スコープ **ページ・パターン・テンプレート・グリフ**

パラメタ 表 4.8・表 4.9 参照。

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 ***void xshow(String text, const double *xadvancelist)***
C ***void PDF_xshow(PDF *p, const char *text, int len, const double *xadvancelist)***

テキストをカレントのフォント・サイズで個々の横位置に印字します。

text （内容文字列）印字したいテキスト。

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。

xadvancelist テキストの各グリフに対する *x* 増分値の配列。それぞれの値は、各グリフを配置した後の相対横変位を指定します（ユーザー座標で）。配列の要素数は、テキストのグリフ数と同じにする必要があります（*len* とは同じとは限りません。*len* はバイト数だからです！）。

詳細 フォントはあらかじめ、*PDF_setfont()* で設定しておく必要があります。

スコープ **ページ・パターン・テンプレート・グリフ**

パラメタ 表 4.8・表 4.9 参照。

バインディング C・C++ 言語バインディングでのみ得られます。他のバインディングでは、*PDF_fit_textline()* で *xadvancelist* オプションを使えば、同じ動作をさせることができます。

C++ Java ***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 キャラクタを含んでいてはいけません。null キャラクタ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 （*PDF_show_xy2()* 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。

詳細 フォントはあらかじめ、*PDF_setfont()* 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。

スコープ **ページ・パターン・テンプレート・グリフ**

パラメタ 表 4.8・表 4.9 参照。

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 ***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 キャラクタを含んでいてはいけません。null キャラクタ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 （*PDF_continue_text2()* 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、*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* パラメタと、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。この関数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。**この関数は縦書きでは使ってはいけません。

パラメタ 表 4.8・表 4.9 参照。

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 ***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 キャラクタを含んでいてはいけません。null キャラクタ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 （*PDF_stringwidth2( )* 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。

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（*PDF_setfont( )* 参照）。

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、次のテキストパラメタのカレント値が考慮されます：*horizscaling*・*kerning*・*charspacing*・*wordspacing*。

スコープ **フォント・ページ・パターン・テンプレート・パス・グリフ・文書**

パラメタ 表 4.8・表 4.9 参照。

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。

4.5 Unicode 変換関数

これらの関数は、Unicode 文字列の変換に便利でしょう。これらは、このような変換機能を提供してい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。

バインディング　Unicode 変換関数は、Unicode 対応言語バインディングでは利用できません（C++ を除き）。

C バインディング：この節の関数によって返される文字列は、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関数への呼び出しを 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。

C++　***string utf16_to_utf8(string utf16string)***
Perl PHP　***string utf16_to_utf8(string utf16string)***
C　***const char *PDF_utf16_to_utf8(PDF *p, const char *utf16string, int len, int *size)***

文字列を UTF-16 形式から UTF-8 へ変換します。

utf16string　変換したい文字列。文字列内の Byte Order Mark（BOM）は解釈されます。BOM がないときは、プラットフォームのネイティブのバイト順序と見なされます。

len　（C 言語バインディングのみ）*utf16string* の長さをバイト単位で指定します。

size　（C 言語バインディングのみ）返される文字列の長さ（バイト単位）を格納させたいメモリ位置への C スタイルのポインタ。ポインタを NULL にすると、それは無視されます。

戻り値　変換後の UTF-8 文字列。この生成される UTF-8 文字列は、それが範囲 < U+007F 内のキャラクタのみを含んでいる場合を除いて、UTF-8 の BOM（*\xEF\xBB\xBF*）が先頭についています。EBCDIC プラットフォームでは、この BOM を含む変換結果は、最終的に EBCDIC へ変換されます。

スコープ　任意

バインディング　この関数は、Unicode 対応の言語バインディングでは得られません。

C++　***string utf8_to_utf16(string utf8string, string ordering)***
Perl PHP　***string utf8_to_utf16(string utf8string, string ordering)***
C　***const char *PDF_utf8_to_utf16(PDF *p, const char *utf8string, const char *ordering, int *size)***

文字列を UTF-8 形式から UTF-16 へ変換します。

utf8string　変換したい文字列。有効な UTF-8 シーケンスを入れる必要があります（EBCDIC プラットフォームでは、それを EBCDIC でエンコードする必要があります）。Byte Order Mark（BOM）があるときは、それは除去されます。

ordering　結果文字列のバイト順序を以下のように指定します。

- *utf16* または空文字列：変換後の文字列は BOM を持たず、プラットフォームのネイティブのバイト順序で格納されます。

- *utf16le*：変換後の文字列はリトルエンディアン形式になり、先頭にリトルエンディアン BOM（*\xFF\xFE*）がつきます。
- *utf16be*：変換後の文字列はビッグエンディアン形式になり、先頭にビッグエンディアン BOM（*\xFE\xFF*）がつきます。

size （C 言語バインディングのみ）返される文字列の長さ（バイト単位）を格納させたいメモリ位置への C スタイルのポインタ。

戻り値 変換後の UTF-16 文字列。

スコープ 任意

バインディング この関数は、Unicode 対応の言語バインディングでは得られません。

C++ ***string utf32_to_utf16(string utf32string, string ordering)***
Perl PHP ***string utf32_to_utf16(string utf32string, string ordering)***
C ***const char *PDF_utf32_to_utf16(PDF *p, const char *utf32string, const char *ordering, int *size)***

文字列を UTF-32 形式から UTF-16 へ変換します。

utf32string 変換したい文字列。有効な UTF-32 シーケンスを入れる必要があります。Byte Order Mark（BOM）があるときは、それは解釈されます。

len （C 言語バインディングのみ）*utf32string* の長さをバイト単位で指定します。

ordering 結果文字列のバイト順序を以下のように指定します。
- *utf16* または空文字列：変換後の文字列は BOM を持たず、プラットフォームのネイティブのバイト順序で格納されます。
- *utf16le*：変換後の文字列はリトルエンディアン形式になり、先頭にリトルエンディアン BOM（*\xFF\xFE*）がつきます。
- *utf16be*：変換後の文字列はビッグエンディアン形式になり、先頭にビッグエンディアン BOM（*\xFE\xFF*）がつきます。

size （C 言語バインディングのみ）返される文字列の長さ（バイト単位）を格納させたいメモリ位置への C スタイルのポインタ。

戻り値 変換後の UTF-16 文字列。

スコープ 任意

バインディング この関数は、Unicode 対応の言語バインディングでは得られません。

C++ ***string utf32_to_utf8(string utf32string)***
Perl PHP ***string utf32_to_utf8(string utf32string)***
C ***const char *PDF_utf32_to_utf8(PDF *p, const char *utf32string, int len, int *size)***

文字列を UTF-32 形式から UTF-8 へ変換します。

utf32string 変換したい文字列。有効な UTF-32 シーケンスを入れる必要があります。Byte Order Mark（BOM）があるときは、それは解釈されます。

len （C 言語バインディングのみ）*utf32string* の長さをバイト単位で指定します。

size （C 言語バインディングのみ）返される文字列の長さ（バイト単位）を格納させたいメモリ位置への C スタイルのポインタ。

戻り値 変換後の UTF-8 文字列。この生成される UTF-8 文字列は、それが範囲 < U+007F 内のキャラクタのみを含んでいる場合を除いて、UTF-8 の BOM（*\xEF\xBB\xBF*）が先頭についています。EBCDIC プラットフォームでは、この BOM を含む変換結果は、最終的に EBCDIC へ変換されます。

スコープ 任意

バインディング この関数は、Unicode 対応の言語バインディングでは得られません。

C++ ***string utf16_to_utf32(string utf16string, string ordering)***

Perl PHP ***string utf16_to_utf32(string utf16string, string ordering)***

C ***const char *PDF_utf16_to_utf32(PDF *p, const char *utf16string, int len, const char *ordering,int *size)***

文字列を UTF-16 形式から UTF-32 へ変換します。

utf16string 変換したい文字列。文字列内の Byte Order Mark（BOM）は解釈されます。BOM がないときは、プラットフォームのネイティブのバイト順序と見なされます。

len （C 言語バインディングのみ）*utf16string* の長さをバイト単位で指定します。

ordering 予約済み。空にする必要があります。

size （C 言語バインディングのみ）返される文字列の長さ（バイト単位）を格納させたいメモリ位置への C スタイルのポインタ。ポインタを NULL にすると、それは無視されます。

戻り値 変換後の、プラットフォームのネイティブなバイト順序の UTF-32 文字列。

スコープ 任意

バインディング この関数は、Unicode 対応の言語バインディングでは得られません。

5 テキストと表の組版

5.1 テキストオプション

この項では、*PDF_fit_textline()*・*PDF_info_textline()*・*PDF_add/create_textflow()* のテキストオプション群を挙げます。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：

- 表 5.1 に従った入力フィルタオプション群。
- 表 5.2 に従ったテキスト書式オプション群。
- 表 5.3 に従ったシェーピング・タイポグラフィオプション群。

表 5.1 入力フィルタオプション一覧

オプション	説明
charref	（論理値）true にすると、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。デフォルト：グローバルな charref パラメタ
escape-sequence	（論理値）true にすると、内容文字列・ハイパーテキスト文字列・名前文字列の中のエスケープシーケンスの置き換えが有効になります。PDF_create_textflow( ) に与えるテキストの中にインラインオプションリストがある場合、インラインオプションリストの先頭キャラクタはエスケープシーケンスで表すことも可能です。しかし、インラインオプションリスト内では、末尾キャラクタも含め、エスケープシーケンスは働きません。デフォルト：グローバルな escapesequence パラメタ
glyphcheck	（キーワード）グリフ検査ポリシ。すなわち、選択したフォント内のグリフへマップできないコードがテキスト内にあるときにどのように動作するか（デフォルト[1]：replace）： ***none*** 検査しません ***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。 ***replace*** PDFlib は、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。
textformat	（キーワード。Unicode 非互換の言語バインディングのみ）与えるテキストの解釈に使わせたい形式。使えるキーワード：bytes・utf8・ebcdicutf8（iSeries ・zSeries のみ）・utf16・utf16le・utf16be・auto。デフォルト：グローバルな textformat パラメタ（表 4.8 参照）。

1. PDF_fit_textline( )・PDF_info_textline( ) で inittextstate=false の場合のデフォルト：オプションと同名のパラメタ（表 4.8・表 4.9 参照）

表 5.2 テキスト書式オプション一覧

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[1]：0
dasharray	（float 2 個のリスト）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。デフォルト：{0 0}（すなわち実線）

表 5.2 テキスト書式オプション一覧

オプション	説明
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。デフォルト [1]：false
fakebold	（論理値）true にすると、3 回の重ね印字で太字フォントに見せかけます。強調にはボールドフォントバリエーションを使うことを強く推奨します。このパラメタが作成するテキスト出力は、本当のボールドのテキストより劣り、テキスト抽出も妨げます。デフォルト [1]：false
fillcolor	（色）テキストの塗り色。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状態の中の対応するパラメタ。デフォルト：{gray 0}（PDF/A モードでは {lab 0 0 0}）
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）。この font オプションを使うと、fontname/encoding オプションによる暗黙的なフォント読み込みよりもパフォーマンスが向上します。 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、そうでないならカレントテキスト状態の中のフォント（テキスト行で inittextstate=false の場合のみ）。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。必須）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。テキストフローの場合、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：カレント文字サイズ
gstate	（グラフィック状態ハンドル）PDF_create_gstate( ) で取得されたグラフィック状態のハンドル。そのグラフィック状態が、この関数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：グラフィック状態なし（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（0 以外にする必要があります）。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 [1]：100
italicangle	（float）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、レギュラー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 [1]：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を有効にします。そうでないなら無効にします。デフォルト：グローバルな kerning パラメタ
overline	（論理値）上線モード。デフォルト [1]：false
strikeout	（論理値）取り消し線モード。デフォルト [1]：false
strokecolor	（色。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）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 をテキストを袋文字に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

表 5.2 テキスト書式オプション一覧

オプション	説明
textrendering	（整数）テキスト表現モード。Type 3 フォントの場合は、値 3 だけが効力を持ちます（デフォルト[1]：0）： *0* テキストを塗る *1* テキストを描線（袋文字） *2* テキストを塗って描線 *3* 不可視テキスト *4* テキストを塗り、クリッピングパスに追加 *5* テキストを描線し、クリッピングパスに追加 *6* テキストを塗って描線し、クリッピングパスに追加 *7* テキストをクリッピングパスに追加 （PDF_fit_textline( ) または PDF_fit_textflow( ) では何ら効力を持ちません）
textrise	（float またはパーセント値）テキストライズパラメタ。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[1]：0
underline	（論理値）下線モード。デフォルト[1]：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、フォントのメトリックまたはアウトラインファイル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[1]：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、フォントのメトリックまたはアウトラインファイル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[1]：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（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[1]：0

1. PDF_fit_textline( )・PDF_info_textline( ) で inittextstate=false の場合のデフォルト：オプションと同名のパラメタ（表 4.8・表 4.9 参照）

表 5.3 シェーピング・タイポグラフィオプション一覧

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に対するキーワードは、警告を出さずに無視されます。以下のキーワードを与えることが可能です： ***_none*** フォント内の機能を一切適用しません。ただし例外として、vert 機能は novert キーワードで明示的に無効化する必要があります。 ***<name>*** 機能を有効にするために、その 4 文字の OpenType タグ名を与えます。よく使われる機能名は 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は PDFlib チュートリアルにあります。 ***no<name>*** 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。 デフォルト：横書きでは _none、縦書きでは vert。
language	（キーワード。script を与えているときのみ意味を持ちます）指定した言語に従ってテキストが処理されます。これは 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 なら必須）指定した用字系に従ってテキストが処理されます。これは features・shaping・advancedlinebreak オプションに対して意味を持ちます。用字系としてもっともよく使われるキーワードは次のとおりです：_none（用字系未定義）・latn・grek・cyrl・armn・hebr・arab・deva・beng・guru・gujr・orya・taml・thai・laoo・tibt・hang・kana・han。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。キーワード _auto を指定すると、テキスト内のキャラクタの多数が属する用字系が選択されます。その際、latn と _none は無視されます。_auto は、shaping に対してのみ意味を持ち、features と advancedlinebreak に対しては無視されます。デフォルト：_none
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、同じフォント内のキャラクタ群に対してのみ行われます。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

表 5.4 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一覧

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・encoding オプションを用いて）指定するときは、表 4.3 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、数値またはキーワード fontsize・ascender・xheight・baseline・descender・textrise のいずれかで指定します。デフォルト：baseline

5.2 テキスト行による一行テキスト

クックブック　完全なコードサンプルがクックブックの text_output/starter_textline トピックにあります。

C++ Java　***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 111 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*（27 ページ「2.5 例外処理」参照）
- 表 4.3 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群：
  ascender・autocidfont・autosubsetting・capheight・descender・embedding・encoding・fallbackfonts・fontname・fontstyle・keepnative・linegap・metadata・monospace・readfeatures・replacementchar・subsetlimit・subsetminsize・subsetting・unicodemap・vertical・xheight
- 表 5.1 に従った入力フィルタオプション群：
  charref・escapesequence・glyphcheck・textformat
- 表 5.2 に従ったテキスト書式オプション群：
  charspacing・dasharray・decorationabove・fakebold・fillcolor・font・fontsize・gstate・horizscaling・inittextstate・italicangle・kerning・overline・strikeout・strokecolor・strokewidth・textrendering・textrise・underline・underlineposition・underlinewidth・wordspacing
- 表 5.3 に従ったシェーピング・タイポグラフィオプション群：
  features・language・script・shaping
- 表 6.1 に従ったはめ込みオプション群：
  alignchar・blind・boxsize・fitmethod・margin・matchbox・orientate・position・rotate・stamp・showborder・shrinklimit
- 表 5.5 に従ったテキスト行組版のためのオプション群：
  inittextstate・leader・shadow・textpath・xadvancelist

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・グラフィック状態のパラメタ群が、オプションで明示的に上書きしない限り、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・グラフィック状態のパラメタ群のデフォルト値が、オプションで明示的に上書きしない限り、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に対しては効力を持ちません。

カレントのテキスト・グラフィック状態は、この関数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* パラメタは、生成したテキスト出力の末尾位置へ変更されます。

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**。この関数は縦書きでは使ってはいけません。

パラメタ　表 4.8・表 4.9 参照。

表 5.5 PDF_fit_textline( ) の追加オプション一覧

オプション	説明
inittextstate	（論理値）true の場合、すべてのテキスト書式オプションがそれぞれのデフォルト値へ初期化されます。false の場合、カレントテキスト状態値が用いられます。デフォルト：false
leader	（オプションリスト。boxsize を指定していないとき、または枠の幅が 0 のときは無視されます）隙間埋めテキスト（点リーダ等）と組版オプション群を指定します。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 5.4 を参照。デフォルト：リーダなし
shadow	（オプションリスト）テキストに影を付けます（デフォルト：影なし）： ***offset***　（float 2 個のリスト）テキスト行の参照点からの影の変位を、ユーザー座標系か、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：{5% -5%} ***fillcolor***　（色）影の色。デフォルト：{gray 0.8} ***gstate***　（グラフィック状態ハンドル）影に適用させたい、PDF_create_gstate( ) で取得したグラフィック状態。デフォルト：なし
stamp	（キーワード。boxsize を指定していないときは無視されます）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スタンプを生成することができます。スタンプのテキストは可能な限り大きくされます。スタンプのテキストを枠内に配置する際、position・fitmethod・orientate（north・south のみ）オプションは効力を持ちます。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 5.5 PDF_fit_textline( ) の追加オプション一覧

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。使えるサブオプションの一覧は表 5.6 を参照。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。 PDF_fit_textline( ) の以下のオプション群は、パス上の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*** 各グリフの回転角を指定します。 以下のはめ込み枠関連のオプション群は無視されます： boxsize・margin・fitmethod・orientate・alignchar・showborder・stamp・leader カーニングと、日中韓レガシエンコーディングによるテキストは、パス上のテキストでは対応していません。
xadvancelist	（float のリスト）テキスト内のすべてのグリフの変位幅を指定します。リストの要素数は、テキストのグリフ数以下にする必要があります。標準のグリフ幅のかわりに、この xadvance 値がそれぞれ使われます。それ以外のカーニングや字間などの書式に対しては効力を持ちません。

表 5.6 PDF_fit_textline( ) の textpath オプションのサブオプション一覧

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、テキストはパスの右端に配置され、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。
rotate	（float）参照点を中心とし、与えた値を度単位で表した回転角として、パスを回転させます。デフォルト：0
scale	（float 1 個か 2 個のリスト）参照点を中心とし、与えた値を横・縦の拡縮倍率として、パスを拡縮小します。値を 1 個だけ与えると、それが両方向に対して用いられます。デフォルト：{1 1}
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%
subpaths	（整数のリストか、またはキーワード 1 個）描画させたいサブパスの番号のリスト。キーワード all を指定するとすべてのサブパスになります。デフォルト：all
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：パスが構築された時に指定された値、あるいは値が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：パスが構築された時に指定された値、あるいは値が指定されなかったなら 0

C++ Java ***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。

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 終了文字列に対しては 0。

keyword ほしい情報を表 5.7 に従って指定したキーワード。

optlist *PDF_fit_textline( )* のオプション群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 この関数は、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、この関数はエラー時に 0 を返します。*errorpolicy=exception* の場合、この関数はば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 5.7 PDF_info_textline( ) のキーワード一覧

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・キャップハイト・ディセンダ。それぞれ、同名のタイポグラフィ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系で表したもの
height	範囲枠の boxheight 指定に従ったテキスト文字列の高さ
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、タイポグラフィ的に類似のキャラクタ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群に基づいて、かつ、テキスト内に方向マーカが存在していれば（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。
scalex・scaley	横・縦伸縮倍率。これが 1 でないなら、テキストは枠に収めるために伸縮される必要があります。

表 5.7 PDF_info_textline( ) のキーワード一覧

キーワード	説明
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
unknownchars	glyphcheck=none の場合：スキップされたキャラクタの数。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（replacementchar オプション）で置き換えられたキャラクタの数。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と、タイポグラフィ的に類似のキャラクタに置き換えることができなかったキャラクタ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の数。すなわち、replacedchars と unknownchars の合計。
wellformed	それぞれ対応するオプションで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。
width	テキスト文字列の幅（横書きの場合）か、またはもっとも幅の広いグリフの幅（縦書きの場合）。字間は、末尾グリフの後には適用されません。
writingdirx ***writingdiry***	優勢な筆記方向（行内のテキスト進行方向）の x・y 座標、すなわち、(startx, starty) から (endx, endy) への単位ベクトル。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。同名のタイポグラフィ特性をユーザー座標で表したもの

5.3 テキストフローによる複数行テキスト

クックブック　完全なコードサンプルがクックブックの text_output/starter_textflow トピックにあります。

C++ Java　***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。テキストには、任意のインラインテキストを入れることもできます。

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 終了文字列に対しては 0。

optlist　テキストフローオプションを表 5.5・表 5.8 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*（27 ページ「2.5 例外処理」参照）
- 表 4.3 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群：
  ascender・autocidfont・autosubsetting・capheight・descender・embedding・encoding・fallbackfonts・fontname・fontstyle・keepnative・linegap・metadata・monospace・readfeatures・replacementchar・subsetlimit・subsetminsize・subsetting・unicodemap・vertical・xheight
- 表 5.1 に従った入力フィルタオプション群：
  charref・escapesequence・glyphcheck・textformat
- 表 5.2 に従ったテキスト書式オプション群：
  charspacing・dasharray・decorationabove・fakebold・fillcolor・font・fontsize・gstate・horizscaling・inittextstate・italicangle・kerning・overline・strikeout・strokecolor・strokewidth・textrendering・textrise・underline・underlineposition・underlinewidth・wordspacing
- 表 5.3 に従ったシェーピング・タイポグラフィオプション群：
  features・language・script・shaping
- 表 5.8 に従ったテキストフロー組版のためのオプション群：
  alignment・avoidemptybegin・fixedleading・hortabmethod・hortabsize・lastalignment・leader・leading・leftindent・minlinecount・parindent・rightindent・ruler・tabalignment
- 表 5.9 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群：
  adjustmethod・advancedlinebreak・avoidbreak・locale・maxspacing・minspacing・nofitlimit・shrinklimit・spreadlimit

- 表 5.10 に従ったコマンドオプション群：
  comment・*mark*・*matchbox*・*nextline*・*nextparagraph*・*resetfont*・*return*・*space*
- 表 5.11 に従ったテキスト意味付けオプション群：
  charclass・*charmapping*・*hyphenchar*・*tabalignchar*

戻り値 テキストフローハンドル。テキストフロー関連の関数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 パラメタを -1 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* パラメタで与えたハンドルが返されます。デフォルトではこの関数は、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* パラメタまたは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* パラメタで与えたハンドルは、以後の関数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 この関数は、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、この関数はそれとは異なり、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。*text* と *optlist* の両方を空にすることもできます。

textflow=-1 にすると、この関数はほとんど *PDF_create_textflow( )* と同等になります。ただし *PDF_create_textflow( )* とは違って、この関数は *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）。

この関数は、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 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（関数 *PDF_fit_textflow( )* は終了して文字列 *_nextpage* を返します）。

水平タブキャラクタ（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（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 5.8 PDF_add/create_textflow( ) と、PDF_create_textflow( ) のインラインの、追加の組版オプション一覧

オプション	説明
alignment	（キーワード）段落内の行の整列を指定します。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
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）を指定します。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 5.4 を参照。デフォルト：リーダなし
leading	（float またはパーセント値）隣りあうテキストのベースラインの間隔[2]。実際の値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があるときは、行送りは最後の関連オプション（font・fontsize・leading 等）によって決定されます。その同じ行にさらにオプションリストがあるときは、行送りに関連するオプションはすべて、fixedleading=false のときにのみ考慮されます。行の中にオプションリストが全くないときは、前の行送り値が考慮されます。デフォルト：100%
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

表 5.8 PDF_add/create_textflow( ) と、PDF_create_textflow( ) のインラインの、追加の組版オプション一覧

オプション	説明
ruler	（float かパーセント値のリスト） hortabmethod=ruler のとき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
tabalignment	（キーワードのリスト。hortabmethod=ruler の場合のみ） タブ位置の整列。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
2. 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一覧

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 5.9 PDF_add/create_textflow( ) と、PDF_create_textflow( ) のインラインの、改行アルゴリズムを制御するための追加のオプション一覧

オプション	説明
locale	（キーワード）advancedlinebreak=true のとき、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、以下の構成要素（複数可）から成り、オプショナルな構成要素は下線キャラクタ「_」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（www.loc.gov/standards/iso639-2 参照）。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 15924 に従った、4 文字の用字系コード（www.unicode.org/iso15924/iso15924-codes.html 参照）。例：Hira（ひらがな）・Hebr（ヘブライ文字）・Arab（アラビア文字）・Thai（タイ文字）。 ▸（オプション）ISO 3166 に従った、大文字 2 文字の国コード（www.iso.org/iso/country_codes/iso_3166_code_lists 参照）。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、いくつかの用字系に対しては、高度な改行のために必要です。デフォルト：_none 例：Thai・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 U+0020 を少なくとも 1 個含み、かつ alignment=justify のとき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 のとき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 にしているときの、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 方式にしているときの、字間の上限[2]。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
2. 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 5.10 PDF_add/create_textflow( ) と、PDF_create_textflow( ) のインラインの、追加のコマンドオプション一覧

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
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 6.3 に従った、範囲枠を作成するためのオプションリスト
nextline ***nextparagraph***	（論理値）改行または改段落を強制します。

表 5.10 PDF_add/create_textflow( ) と、PDF_create_textflow( ) のインラインの、追加のコマンドオプション一覧

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の特性について最初の設定と違う設定を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。
return	（文字列。先頭にアンダースコアキャラクタ _ 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。
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 5.11 PDF_add/create_textflow( ) と、PDF_create_textflow( ) のインラインの、追加のテキスト意味付けオプション一覧

オプション	説明
charclass	（対のリスト。それぞれの対の 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。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*** 文字（a B 等）のように動作します ***punct*** 句読点（+ / ; : 等）のように動作します ***open*** 開きカッコ（[等）のように動作します ***close*** 閉じカッコ（] 等）のように動作します ***default*** すべてのキャラクタ分類を、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	（対のリスト。それぞれの対は 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）個々のキャラクタを、別のキャラクタ（複数可）で置き換えます。オプションリストには Unichar の対を入れます（複数可）、。それぞれの対の 1 番目のキャラクタが、2 番目のキャラクタに置き換わります。一対一対応でなく、それぞれの対の 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をその個数並べます。 **個数＜ *o*** キャラクタ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*** キャラクタを削除します。 例： charmapping={ hortab space CRLF space LF space CR space } charmapping={ shy {shy 0} } charmapping={ hortab {space 4} }
hyphenchar	（Unichar 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。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。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）小数点タブを整列させたいキャラクタ。デフォルト：U+002E「.」

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 5.12 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。下記に、2 個のマクロのマクロ定義の例を示します。

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

これらのマクロは、オプションリストの中で下記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、任意の深さの入れ子にすることができます(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、関数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 5.12 PDF_add/create_textflow( )・PDF_fit_textflow( ) 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	(対のリスト)それぞれの対は、マクロの名前と定義を以下のように記述します。 ***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 & 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をマクロ名の中で使ってはいけません。 入れ子のマクロは、入れ子の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 ***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。さまざまなエンコーディングのテキストと、マクロと（90 ページ「テキストフローオプションのマクロ」参照）、表 5.8・表 5.13 に従ったインラインオプションリスト（92 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 終了文字列に対しては 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：

- 一般オプション：*errorpolicy*（27 ページ「2.5 例外処理」参照）
- *PDF_add_textflow( )* のすべてのオプション（*PDF_add_textflow( )* のオプション一覧を参照）
- インラインオプションリストの処理を表 5.13 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、この関数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* パラメタまたはオプションで変えることもできます。

詳細 この関数は、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（92 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）。

この関数は、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**

テキストフローのインラインオプションリスト *PDF_create_textflow( )*（*PDF_add_textflow( )* では不可）の *text* 引数で与える内容の中には、テキストフローオプションを表 5.8 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_*

表 5.13 PDF_create_textflow( ) 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一覧

オプション	説明
begoptlistchar	（Unichar 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。デフォルトのキャラクタ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に替えると便利かもしれません（92 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。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。デフォルト：U+003F（>）
fixedtextformat	（論理値。Unicode 対応の言語バインディングでは無視されます。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、テキスト本体で使われている形式にかかわらず、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。Unicode 非対応言語で fixedtextformat=false かつ textformat=utf16xx の部分テキストに対しては必須）次のインラインオプションリストの前のバイトか（Unicode 対応言語の場合）、またはキャラクタの数（92 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）。キャラクタは、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。

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（デフォルトでは < と >）で囲む必要があります。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<キャラクタで開始。

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

  ただしこの方式は、*encoding=builtin* によるフォントに対しては動作しません。
- *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を設定し（*$* 等）、そのキャラクタ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 encoding=winansi>
```

- 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 encoding=winansi>
```

注 **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 ***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 対象にしたい矩形（**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。矩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 処理オプション群を表 5.14 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：
blind・createfittext・createlastindent・exchangefillcolors・exchangestrokecolors・firstlinedist・fitmethod・fontscale・lastlinedist・linespreadlimit・maxlines・minfontsize・orientate・returnatmark・rewind・rotate・showborder・showtabs・stamp・verticalalign・wrap

戻り値 関数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 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、次にこの関数を呼び出すまで有効です。

詳細　カレントテキスト・グラフィック状態は、この関数が作成するテキスト出力に対して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）。この関数から返った後で、テキスト状態は変更されていません。ただしカレントテキスト位置は、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 5.14 PDF_fit_textflow( ) のオプション一覧

オプション	説明
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。使えるサブオプション： **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 6.3 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対で、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。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対で、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。

表 5.14 PDF_fit_textflow( ) のオプション一覧

オプション	説明
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。ヒ等の、読み分け記号つきの代表的なキャラクタ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値。d や h 等の、大きなアセンダを持つ代表的なキャラクタ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。
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を指定します。デフォルト：clip ***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はみ出させることもあります。
fontscale	（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	（グラフィック状態ハンドル）PDF_create_gstate( ) で取得したグラフィック状態のハンドル。このグラフィック状態が、この関数で配置されるすべてのテキストに対して効力を持ちます。別のグラフィック状態をすでに PDF_add/create_textflow( ) に与えていたときは、両方のグラフィック状態がマージされます。デフォルト：グラフィック状態なし（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 にしているときは無視されます）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 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値。g や j 等の、ディセンダを持つ代表的なキャラクタ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 にしているときのみ）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

表 5.14 PDF_fit_textflow( ) のオプション一覧

オプション	説明
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%
orientate	(キーワード) テキストを配置する時に向けたい向きを指定します。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) 与えるテキストフローの状態を、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にリセットします。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( ) を最後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状態を使って)。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状態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*** 線をそれが有効な垂直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。

表 5.14 PDF_fit_textflow( ) のオプション一覧

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 5.15 に従ったオプションリスト）テキストが、表 5.15 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 引数で与える矩形にかぎらず）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 5.15 PDF_fit_textflow( ) の wrap オプションのサブオプション一覧

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 2 個以上。
boxes	（矩形のリスト）回り込み領域に追加したい矩形（複数可）。
circles	（円のリスト）回り込み領域に追加したい円（複数可）。

表 5.15 PDF_fit_textflow( ) の wrap オプションのサブオプション一覧

オプション	説明
creatematch-boxes	（オプションリストのリスト）boxes オプション内の矩形 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 6.3 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を指定します（デフォルト：evenodd）。詳しくは表 7.2 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、テキスト行の垂直範囲を定義します。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、示したい矩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矩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。それぞれの矩形の外接枠が、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パス（複数可）。使えるサブオプション： ***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( ) の下記のオプション（表 6.1・表 7.8 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折れ線（複数可。閉じていなくてもかまいません）。

C++ Java ***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状態を取得します。

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 5.16 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。この関数は、ブラインドモードでも正しい位置情報を返します（*textx/texty* パラメタとは異なり）。

スコープ **オブジェクト**以外の任意

表 5.16 PDF_info_textflow( ) のキーワード一覧

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）
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応するテキスト文字列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。文字列は下記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 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
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

表 5.16 PDF_info_textflow( ) のキーワード一覧

キーワード	説明
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を取得するために PDF_get_parameter( ) の string パラメタ（表 2.3 参照）で使える文字列番号（表 2.3 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（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

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 ***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 この関数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

5.4 表の組版

クックブック　完全なコードサンプルがクックブックの tables/starter_table トピックにあります。

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* 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。

optlist　表セルの組版の詳細を表 5.17 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*（27 ページ「2.5 例外処理」参照）
- 列・行定義：*colwidth*・*colscalegroup*・*minrowheight*・*return*・*rowheight*・*rowjoingroup*・*rowscalegroup*
- セル特性：*checkwordsplitting*・*colspan*・*margin*・*marginleft*・*marginbottom*・*marginright*・*margintop*・*matchbox*・*rowspan*
- セル内容：*annotationtype*・*continuetextflow*・*fieldname*・*fieldtype*・*fitannotation*・*fitfield*・*fitimage*・*fitpdipage*・*fittextline*・*fittextflow*・*fitpath*・*image*・*path*・*pdipage*・*repeatcontent*・*textflow*

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・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 回の関数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。

スコープ　**オブジェクト**以外の任意

表 5.17 PDF_add_table_cell( ) のオプション一覧

オプション	説明
annotation-type	（文字列）表セルに挿入したい注釈の種類を表 12.6 に従って指定します。
checkword-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、テキストを表セルにはめ込む際にテキストフローが少なくとも 1 箇所で単語の分割を強制する必要があるかどうかを検査します。もしそうなら、セルの幅を拡げて、単語の分割を回避しようとします。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・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、checkwordsplitting 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。
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）表セルに挿入したいフォームフィールドの種類を表 12.9 に従って指定します。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 12.7 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 12.10 に従ったフォームフィールドオプション群。
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、与えている画像またはテンプレートをセルの中に配置するために適用されます。セル内枠の左下隅が参照点として使われます。 デフォルト：boxsize={<幅> <高さ>} fitmethod=meet position=center、ここで <幅> と <高さ> は、算出されたセル内枠の幅と高さです。この算出されたオプションリストは、ユーザーが与えるオプションリストの頭につけられます。[3]
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。path オプションで指定し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、このオプションリストが適用されます。セル内枠の左下隅が参照点として用いられます。 デフォルト：boxsize={<幅> <高さ>} fitmethod=meet position=center、ここで <幅> と <高さ> は、算出されたセル内枠の幅と高さです。この算出されたオプションリストは、ユーザーが与えるオプションリストの頭につけられます。[3]

表 5.17 PDF_add_table_cell( ) のオプション一覧

オプション	説明
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 が利用可能な場合のみ）PDF_fit_pdi_page( ) に対するオプションリスト。与えたページをセル内に配置するために、このオプションリストが適用されます。セル内枠の左下隅が参照点として用いられます。 デフォルト：boxsize={<幅> <高さ>} fitmethod=meet position=center、ここで <幅> と <高さ> は、算出されたセル内枠の幅と高さです。この算出されたオプションリストは、ユーザーが与えるオプションリストの頭につけられます。[3]
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。textflow オプションで与えたテキストフローをセル内に配置するために、このオプションリストが適用されます。セル内枠がはめ込み枠として使われます。 デフォルト：verticalalign=center lastlinedist=descender。このオプションリストは、ユーザーが与えるオプションリストの頭につけられます。
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。与えたテキストをセル内に配置するために、このオプションリストが適用されます。セル内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。カレントテキスト状態は考慮されません。 デフォルト：boxsize={<幅> <高さ>} fitmethod=nofit position=center、ここで <幅> と <高さ> は、算出されたセル内枠の幅と高さです。この算出されたオプションリストは、ユーザーが与えるオプションリストの頭につけられます。[3]
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。
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
matchbox	（オプションリスト）範囲枠の詳細を表 6.3 に従って入れたオプションリスト
minrow-height[1]	（float またはパーセント値）ある表行を 1 つの表インスタンスの中で配置しきれないときに、このオプションは、行を分割していいか、そして各部分はどれだけ小さくできるかを指定します。部分の最小の高さを、ユーザー座標系か、または表行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。デフォルト：100%、すなわち分割なし
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。デフォルト：なし
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し印字する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。

表 5.17 PDF_add_table_cell( ) のオプション一覧

オプション	説明
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「_」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・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
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、セル内枠内に配置されます。continue 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。デフォルト：テキストフローなし

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。
3. 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。

C++ Java ***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矩形（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 5.18 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*（27 ページ「2.5 例外処理」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：*header*・*footer*
- 表装飾：*fill*・*firstdraw*・*gstate*・*stroke*
- 開発・デバッグの視覚支援：*debugshow*・*showcells*・*showgrid*

戻り値　関数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* パラメタまたは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。以後この関数を呼び出して、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 1 個のセルの領域。
- 内容：指定したセル内容は、次の順序で配置されます：画像・取り込み PDF ページ・テキストフロー・テキスト行・パスオブジェクト・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 1 個のセルの領域。
- 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、表関数以外の、注釈・フォームフィールド・画像などをつけることができます。

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を含んでいる場合には、各スコープが勝ります。たとえば、フォームフィールドか注釈を含んでいる表は、テンプレート上に配置することができません。

表 5.18 PDF_fit_table( ) のオプション一覧

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
colwidth-default	（float またはキーワード。1 個の表についての最初の PDF_fit_table( ) への呼び出しでのみ意味を持ちます）PDF_add_table_cell( ) の 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（デフォルト：auto）： *auto*　幅が指定されなかったすべての列に対して、はめ込み枠の幅が均等に分配されます。結果として、表ははめ込み枠の幅いっぱいに広がります。 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）。

表 5.18 PDF_fit_table( ) のオプション一覧

オプション	説明
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、デバッグの助けとして作成されます。デフォルト：false
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。118 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*** ヘッダグループの表行すべて ***footer*** フッタ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 7.2 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*** 例： 表の表行すべてを赤く塗る：fill = { {area=table fillcolor={rgb 1 0 0}} } 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={rgb 0 1 0}} {area=roweven fillcolor={rgb 1 0 0}} }
firstdraw	（キーワード）横線と縦線を生成する順序を指定します（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）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（フッタ行なし）
gstate	（グラフィック状態ハンドル）PDF_create_gstate( ) で取得したグラフィック状態のハンドル。すべての表装飾が、与えるグラフィック状態に従います。セル内容に対しては効力を持ちません。デフォルト：グラフィック状態なし（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）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）表の状態を、PDF_fit_table( ) を呼び出したいずれかの時の前の状態にリセットします。目下、以下の値が使えます。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( ) を最後に呼び出した時の前（今回の呼び出しの前）の状態へ巻き戻し。

表 5.18 PDF_fit_table( ) のオプション一覧

オプション	説明
rowheight-default	(float またはキーワード。1 個の表についての最初の 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 (デフォルト : auto) : ***auto*** 高さが指定されなかったすべての表行に対して、はめ込み枠の高さが均等に分配されます。結果として、表ははめ込み枠の高さいっぱいに広がります。 最小限の高さを持つ表行を生成するには、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状態を使って描線されます。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 7.2 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*** 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 ***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 ほしい情報を表 5.19 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表特性の値。この関数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。

スコープ **オブジェクト**以外の任意

表 5.19 PDF_info_table( ) のキーワード一覧

キーワード	説明
boundingbox	表インスタンス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）
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
height	表インスタンスの高さ
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	返った原因の文字列番号
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・フッタを含む）
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
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。
width	表インスタンスの幅
x1・y1・…x4・y4	表枠の隅の座標を、左下隅から反時計回りに、ユーザー座標で表したもの
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 ***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 5.20 に従って指定したオプションリスト。

詳細 この関数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 5.20 PDF_delete_table( ) のオプション一覧

オプション	説明
keephandles	（論理値） false にすると、PDF_add_table_cell( ) の textflow・image・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

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、矩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、あるいは矩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関数に実装されています：

- *PDF_fit_textline()*・*PDF_info_textline()*
- *PDF_fit_image()*・*PDF_info_image()*
- *PDF_fit_pdi_page()*・*PDF_info_pdi_page()*
- *PDF_draw_path()*・*PDF_info_path()*
- *PDF_add_table_cell()*（*fittextline*・*fitimage*・*fitpdipage*・*fitpath* 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、85 ページ「5.3 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、これらのはめ込み関数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関数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関数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　すべての場合において、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矩形を算出します。この矩形を**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）：オブジェクト枠のデフォルト幅はテキストの幅であり、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。
- 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矩形がオブジェクト枠として用いられます。
- 取り込み PDF ページ（*PDF_fit/info_pdi_page()*・PDF 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。
- 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矩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。
- 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矩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：関数の引数 *x*・*y*。
- *PDF_info_*()* の場合：点 *(o, o)*。*PDF_info_path()* ではこのほかに、*refpoint* オプションを用いて参照点を指定することもできます。
- *PDF_add_table_cell()*・*PDF_fit_table()*・*PDF_fill_*block()* の場合：表セル・表インスタンス・PDFlib ブロックの左下隅。*PDF_fill_*block()* ではこのほかに、*refpoint* オプションを用いて参照点を指定することもできます。

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矩形を、**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** オプション群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/ury/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・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。

表 6.1 さまざまな関数のはめ込みオプション一覧

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します。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。座標は両方とも 0 であってはいけません。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 またはキーワード。テキスト行のみ）指定したキャラクタ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。パスオブジェクトのみ）取付点の名前：fitmethod=nofit にすると、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。デフォルト：パスオブジェクトの原点
blind	（論理値）true の場合、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関数 PDF_info_*( ) でチェックできます。デフォルト：false
boxsize	（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。 ブロックの場合のデフォルト：ブロックの Rect プロパティの幅と高さ それ以外のすべてのはめ込み関数でのデフォルト：{0 0}

表 6.1 さまざまな関数のはめ込みオプション一覧

オプション	説明
dpi	（float のリスト。画像のみ）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。 このオプションは、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：internal
fitmethod	（キーワード）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 参照。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：テキストフローの場合は clip、表・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。テキスト行のみ）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）表 6.3 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。テキストフローのみ）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）オブジェクトに対して求める、カレント座標系に対する相対的な向きを指定します。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。すべての関数で、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）

表 6.1 さまざまな関数のはめ込みオプション一覧

オプション	説明
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 : 通常は {0 0}、ただし表の場合は {0 100} 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 (ラテン文字等) には left を意味します。
refpoint	(float のリスト。PDF_fill_*block( )・PDF_info_path( ) のみ) 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指定します。 PDF_fill_*block( ) の場合のデフォルト : ブロックの Rect プロパティによって定義される矩形の左下隅 PDF_info_path( ) の場合のデフォルト : {0 0}
rotate	(float。表・表セル・パスオブジェクトでは不可) 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
scale	(float のリスト。画像・テンプレート・取り込み PDF ページ・パスオブジェクトのみ) 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。このオプションは、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}
stamp	(キーワード。テキスト行のみ。boxsize が指定されていないときは無視されます) 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 6.1 さまざまな関数のはめ込みオプション一覧

オプション	説明
showborder	（論理値）true にすると、はめ込み枠の境界がカレントグラフィック状態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。テキスト行のみ）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

表 6.2 さまざまな関数の fitmethod オプションのキーワード一覧。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	（テキスト行のみ。それ以外の種類のオブジェクト：meet と同じ動作）この方式は、テキスト枠をはめ込み枠内に自動的に収めようと試みます。 具体的には：テキストがはめ込み枠内に収まっているときは nofit と同じです。そうでないときは、テキストを横に縮めて（つぶして）はめ込み枠に収まるよう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 minfontsize の値に達するまで文字サイズが縮小されます。 Kraxi Systems Kraxi Systems Kraxi Systems
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。 Kraxi Sys
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）。 Kraxi Systems
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）。 Kraxi Systems
nofit	オブジェクトを置くだけです。scale・dpi が画像に適用されます。 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。 Kraxi Systems
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。 Kraxi S

6.2 範囲枠

範囲枠は、それ専用の関数で定義するのではなく、実際の要素を作成する関数を呼び出すに、*matchbox* オプションで定義します。

- テキスト行：*PDF_fit_textline()*・*PDF_fill_textblock()* で *textflow=false*
- テキストフロー：*PDF_add/create_textflow()*・*PDF_fill_textblock()* で *textflow=true*
- 取り込み PDF ページ：*PDF_fit_pdi_page()*・*PDF_fill_pdf_block()*
- 画像・テンプレート：*PDF_fit_image()*・*PDF_fill_image_block()*
- 表セル：*PDF_add_table_cell()*

範囲枠は、これらの関数の *matchbox* オプションで定義されます。このオプションには、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 7.2 に従ったグラフィック書式オプション群：
  borderwidth・*dasharray*・*dashphase*・*fillcolor*・*gstate*・*linecap*・*linejoin*・*shading*・*strokecolor*
- 表 6.3 に従った範囲枠制御オプション群

範囲枠に対応する矩形の詳細は、*PDF_info_matchbox()* で取得できます。

表 6.3 さまざまな関数の matchbox オプションのサブオプション一覧

オプション	説明
boxheight	（要素 2 個のリスト。各要素は正の float かキーワード。テキスト行・テキストフローのみ）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。テキストフローのみ）範囲枠の幅を、ユーザー座標で、または枠の高さ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。デフォルト：0
clipping	（矩形、またはパーセント値 4 個。画像・取り込み PDF ページのみ。innerbox オプションを指定しているときは無視されます）画像または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矩形の左下隅と右上隅の座標。画像の場合、この切り抜き矩形は、ピクセル単位で、または幅・高さに対するパーセント値で指定できます。PDF ページの場合、この切り抜き矩形は、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で指定できます。デフォルト：{0% 0% 100% 100%}
create-wrapbox	（論理値。テキストフローのみ）true にすると、範囲枠を構成する 1 個ないし複数の矩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、その矩形を回り込みます。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、内側の範囲枠矩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、線は範囲枠矩形の外側に描かれ、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）

表 6.3 さまざまな関数の matchbox オプションのサブオプション一覧

オプション	説明
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、矩形のそれぞれの境界が描かれます。デフォルト：true
end	（論理値。テキストフローのみ）範囲枠の終了を指定します。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入れ子にできません。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。テキストフローのみ）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 画像のみ）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）範囲枠の矩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）、または矩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、同じ名前の矩形がもう 1 個作られます。ですので、1 個の範囲枠が複数の矩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。さまざまな関数で、usematchbox オプションを使って、範囲枠の矩形（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。デフォルト：名前なし
offsetleft ***offsetbottom*** ***offsetright*** ***offsettop***	（float またはパーセント値）算出された矩形から、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、または矩形の幅（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・表セルのみ）テキストフロー：true にすると、範囲枠矩形が次行へ分割されるとき、前の矩形の右辺と、後の矩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）範囲枠矩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）

C++ Java ***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）範囲枠の名前。名前は、範囲枠を定義した際に *matchbox* オプションの *name* サブオプションで定義してある必要があります。
　あるいは、名前「*」（アスタリスクキャラクタ 1 個）を使うと、このページ上のすべての範囲枠を評価することができます。

len （C 言語バインディングのみ）*boxname* 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。

num 求める範囲枠矩形の番号（1 から数える）。*num=0* の特殊な場合については表 6.4 を参照。

keyword 求める情報を表 6.4 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範囲枠特性の値。指定した名前の範囲枠が、または指定した番号の範囲枠矩形がカレントページ上にないときは、あらゆるキーワードが値 0 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。

スコープ ページ・パターン・テンプレート・グリフ・パス・フォント

表 6.4 PDF_info_matchbox( ) のキーワード一覧

キーワード	説明
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：このページ上のこの範囲枠の矩形の数 boxname=* のとき：このページ上に少なくとも 1 個の矩形を持つ範囲枠の数
exists	boxname が範囲枠の名前を内容として持つとき：矩形が存在しているなら 1、そうでないなら 0。 boxname=* の場合は、このキーワードを使って、このページ上のすべての範囲枠を評価することができます： num=0 のとき：任意の範囲枠が存在するなら 1、ないなら 0 それ以外のとき：番号 num を持つ範囲枠が存在するなら 1
height[1]	矩形の高さを、ユーザー座標で表したもの
name	番号 num を持つ範囲枠の名前の文字列番号。対応する文字列は、PDF_get_parameter( ) と string パラメタで取得できます（表 2.3 参照）。
rectangle	num 番目の矩形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）
width[1]	矩形の幅を、ユーザー座標で表したもの
x1・y1・… x4・y4[1]	矩形の 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 グラフィック関数

クックブック 完全なコードサンプルがクックブックの graphics/starter_graphics トピックにあります。

7.1 グラフィック書式パラメタ・オプション

表 7.1 に、この章に関連するパラメタキー名を挙げます（19 ページ「2.2 パラメタ・オプション処理」参照）。

表 7.1 PDF_get/set_parameter( ) のパス関連キー一覧

キー	説明
fillrule	カレント塗り規則を winding か evenodd に設定します（表 7.2 参照）。スコープ：ページ・パターン・テンプレート・グリフ

表 7.2 に、*PDF_create_gstate()*・*PDF_draw_path()*・*PDF_shading_pattern()* のグラフィック書式オプションと、*PDF_fit_table()* の *fill*・*stroke* オプションと、さまざまな関数の *matchbox* オプションを挙げます。

表 7.2 グラフィック書式オプション一覧

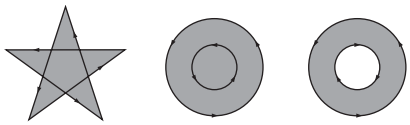
オプション	説明・とりうる値
borderwidth	（float。範囲枠のみ）矩形の境界の線幅。borderwidth を 0 より大きい値に設定すると、すべての矩形の境界が描線されます。上・下・左・右の境界が描線されないようにするには、それぞれ drawtop・drawbottom・drawleft・drawright オプションを false に設定します。デフォルト：0
dasharray	（float のリスト）描線されるパスの短線と間隙の長さ（ユーザー座標系で測ったもの）を交互に指定した 2 ～ 8 個の値のリスト。値はすべてゼロより大きくする必要があります。これらは、パス全体が描線されるまで循環的に再利用されます。
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	（Color）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}）、ただし表・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、単純な形状に対しては 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に対しては異なる結果を生じます。 
flatness	（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	（グラフィック状態ハンドル）PDF_create_gstate( ) で取得したグラフィック状態のハンドル。デフォルト：グラフィック状態なし（すなわち、カレント設定が用いられます）

表 7.2 グラフィック書式オプション一覧

オプション	説明・とりうる値
linecap	（整数またはキーワード）パスの終端の形状（デフォルト：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 オプションの説明を参照）、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）。 マイター長 線幅
shading	（表 7.3 に従ったオプションリスト。範囲枠・表のみ）範囲枠の矩形（群）または表領域のシェーディングを指定します。fillcolor オプションの値が（指定していれば）、またはカレント塗り色が開始色として用いられます。
strokecolor	（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}）、ただし表・範囲枠の場合は none

表 7.3 グラフィック書式オプション shading のサブオプション一覧

オプション	説明
antialias	（論理値）シェーディングの際に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
domain	（float 2 個のリスト）媒介変数 t の限界値を指定した 2 個の数値。この変数は、色勾配が軸の始点と終点の間を変動するにつれて、この 2 値の間を線形的に変動すると考えられます。デフォルト：{0 1}
end	（float 2 個かパーセント値 2 個のリスト）シェーディング軸に対する終点の（type=axial）、または半径を算出するための円上の点の（type=radial）x・y 座標を、矩形の幅・高さに対するパーセント値で、またはユーザー座標で指定します。デフォルト：{100% 100%}
endcolor	（色。必須）終点の色。
N	（float）色遷移関数に対する累乗指数。> 0 にする必要があります。デフォルト：1
start	（float 2 個かパーセント値 2 個のリスト）シェーディング軸に対する始点の（type=axial）、またはシェーディング円の中心の（type=radial）x・y 座標を、矩形の幅・高さに対するパーセント値で、またはユーザー座標で指定します。デフォルト：{0% 0%}
type	（キーワード）シェーディングの種類：axial（線形シェーディング）か radial（放射シェーディング）。デフォルト：axial

7.2 グラフィック状態

グラフィック状態パラメタはすべて、ページの始まりごとにデフォルト値に復帰します。デフォルト値は、それぞれの関数の説明に記します。テキスト状態に関連する関数は、51 ページ「4 章 フォント・テキスト関数」に示しています。

注 **パススコープでは、どのグラフィック状態関数も使ってはいけません。**

C++ Java ***void setdash(double b, double w)***
Perl PHP ***setdash(float b, float w)***
C ***void PDF_setdash(PDF *p, double b, double w)***

カレント破線パターンを設定します。

b・w 交互に描かせたい黒・白部分の数。*b* と *w* は、非負の数値にする必要があります。

詳細 実線をひくには *b* = *w* = *o* に設定します。破線パターンを示す *dasharray・dashphase* パラメタは、ページの始まりごとに実線に設定されます。

スコープ **ページ・パターン・テンプレート・グリフ**

C++ Java ***void setdashpattern(String optlist)***
Perl PHP ***setdashpattern(string optlist)***
C ***void PDF_setdashpattern(PDF *p, const char *optlist)***

破線パターンを、オプションリストで定義して設定します。

optlist 表 7.2 に従ったグラフィック書式オプション群（空のリストにすると、実線が生成されます）：*dasharray・dashphase*

詳細 破線パターンを示す *dasharray・dashphase* パラメタは、ページの始まりごとに実線に設定されます。

スコープ **ページ・パターン・テンプレート・グリフ**

C++ Java ***void setflat(double flatness)***
Perl PHP ***setflat(float flatness)***
C ***void PDF_setflat(PDF *p, double flatness)***

平坦度許容量を設定します。

flatness 平坦度許容量。表 7.2 参照。

詳細 平坦度許容量を示す *flatness* パラメタは、ページの始まりごとにデフォルト値 1 に設定されます。

スコープ **ページ・パターン・テンプレート・グリフ**

C++ Java ***void setlinejoin(int linejoin)***
Perl PHP ***setlinejoin(int linejoin)***
C ***void PDF_setlinejoin(PDF *p, int linejoin)***

線結合スタイルを設定します。

linejoin 描線されるパスの角の形を指定します。表 7.2 参照。この *linejoin* スタイルは、数値 0・1・2 のいずれかで指定する必要があります。

詳細 線結合スタイルを示す *linejoin* パラメタは、ページの始まりごとにデフォルト値 0 に設定されます。

スコープ **ページ・パターン・テンプレート・グリフ**

C++ Java ***void setlinecap(int linecap)***
Perl PHP ***setlinecap(int linecap)***
C ***void PDF_setlinecap(PDF *p, int linecap)***

線端スタイルを設定します。

linecap パスを描線する際の終端の形状を制御します。表 7.2 参照。この *linecap* 引数は、数値 0・1・2 のいずれかで指定する必要があります。

詳細 線端スタイルを示す *linejoin* パラメタは、ページの始まりごとにデフォルト値 0 に設定されます。

スコープ **ページ・パターン・テンプレート・グリフ**

C++ Java ***void setmiterlimit(double miter)***
Perl PHP ***setmiterlimit(float miter)***
C ***void PDF_setmiterlimit(PDF *p, double miter)***

マイターリミットを設定します。

miter マイター結合によって生じるくさび型を制御する 1 以上の値。表 7.2 参照。

詳細 マイターリミットを示す *miterlimit* パラメタは、ページの始まりごとにデフォルト値 10 に設定されます。これは角度にしておよそ 11.5 度にあたります。

スコープ **ページ・パターン・テンプレート・グリフ**

C++ Java ***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。

詳細 線幅を示す *linewidth* パラメタは、ページの始まりごとにデフォルト値 1 に設定されます。

スコープ **ページ・パターン・テンプレート・グリフ**

C++ Java ***void initgraphics( )***
Perl PHP ***initgraphics( )***
C ***void PDF_initgraphics(PDF *p)***

すべての色・グラフィック状態パラメタを、それぞれのデフォルト値にリセットします。

詳細 塗り色・描線色・線幅・線端スタイル・線結合スタイル・マイターリミット・破線パターン・平坦度許容量の各設定と、座標系が、それぞれデフォルト値にリセットされます（テキスト状態パラメタはリセットされません）。カレントクリッピングパスは影響を受けません。

プログラムの流れ的に *PDF_save( )*・*PDF_restore( )* が安易に利用できないような状況でこの関数は有用でしょう。

スコープ **ページ・パターン・テンプレート・グリフ**

C++ Java ***void save( )***
Perl PHP ***save( )***
C ***void PDF_save(PDF *p)***

カレントグラフィック状態をスタックに保存します。

詳細 グラフィック状態は、あらゆる種類のグラフィックオブジェクトを制御するパラメタ群を持っています。グラフィック状態の保存は、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、パラメタ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状態（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。

- それぞれ対応する関数で設定しているグラフィックパラメタ：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。
- 色パラメタ：塗り色・描線色。
- *PDF_set_gstate( )* で明示グラフィック状態によって設定しているグラフィックパラメタ。
- テキスト位置と、次のテキスト関連パラメタ：*charspacing*・*decorationabove*・*fakebold*・*font*・*fontsize*・*horizscaling*・*italicangle*・*leading*・*strokewidth*・*textrendering*・*textrise*・*underlineposition*・*underlinewidth*・*wordspacing*。

PDF_save( ) と *PDF_restore( )* の対は、入れ子にすることもできます。PDF 規格では、保存・復帰の対の入れ子階層の深さに制限はありませんが、さまざまな PDF ビューアが生成する PostScript 出力の持つ制約が引き起こす印刷上の問題を避けるためにも、また、PDFlib が内部的に必要とする保存階層を確保するためにも、アプリケーションでは入れ子階層の深さを 26 未満に抑えておく必要があります。

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_restore( )* と必ず対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。

パラメタ 先述のとおり、テキスト関連パラメタの多くは、保存・復帰の対象となります。次のパラメタは、保存・復帰の対象になりません：*fillrule*・*kerning*・*underline*・*overline*・*strikeout*。

C++ Java ***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状態に復帰します。

詳細 そのグラフィック状態は、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( )* と必ず対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 ***int create_gstate(String optlist)***
Perl PHP ***int create_gstate(string optlist)***
C ***int PDF_create_gstate(PDF *p, const char *optlist)***

グラフィック状態オブジェクトを、さまざまなオプションに従って作成します。

optlist グラフィック状態オプション群を指定したオプションリスト：

- 表 7.2 に従ったグラフィック書式オプション群：
  flatness・linecap・linejoin・linewidth・miterlimit
- 表 7.4 に従ったグラフィック状態オプション群：
  alphaisshape・blendmode・opacitystroke・overprintfill・overprintmode・overprintstroke・renderingintent・smoothness・softmask・strokeadjust・textknockout

戻り値 グラフィック状態ハンドル。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状態パラメタを内容として持つことができます。すべてのパラメタがすべての PDF バージョンで使えるわけではありません。表の中で、最小必須の PDF バージョンを示しています。

スコープ **文書・ページ・パターン・テンプレート・グリフ**

表 7.4 PDF_create_gstate( ) のオプション一覧

キー	説明・とりうる値
alphaisshape	（論理値。PDF 1.4）アルファの供給元の扱いを、輪郭（true）か不透過（false）にします。デフォルト：false
blendmode	（キーワードのリスト。PDF 1.4。PDF/A モードで使うときは、値 Normal にする必要があります）ブレンドモードの名前。複数のブレンドモードを指定することもできます。とりうる値：Color・ColorDodge・ColorBurn・Darken・Difference・Exclusion・HardLight・Hue・Lighten・Luminosity・Multiply・None・Normal・Overlay・Saturation・Screen・SoftLight。デフォルト：None
opacityfill	（float。PDF 1.4。PDF/A モードで使うときは、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。

表 7.4 PDF_create_gstate( ) のオプション一覧

キー	説明・とりうる値
opacitystroke	(float。PDF 1.4。PDF/A モードで使うときは、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。
overprintfill	(論理値) 描線以外の操作でのオーバープリント。デフォルト : false
overprintmode	(整数) オーバープリントモード。0 にすると、それぞれの色要素は、それ以前に配置されたマークを置き換えます。1 にすると (Acrobat では「オーバープリンティングのデフォルトは非ゼロのオーバープリンティング」といいます)、0 の色要素は、そこの要素を変更しないままにします。デフォルト : 0
overprintstroke	(論理値) 描線操作でのオーバープリント。デフォルト : false
renderingintent	(キーワード) 色域圧縮のために用いたいカラーレンダリングインテント。とりうるキーワード : Auto・AbsoluteColorimetric・RelativeColorimetric・Saturation・Perceptual
smoothness	(float) シェーディングにおける線形内挿の最大誤差。≧ 0 かつ≦ 1 にする必要があります
softmask	(オプションリストまたはキーワード) 透過画像処理のためのマスク画像または不透明度値を持ったカレントソフトマスク。キーワード none はソフトマスクなしを指定し、これは、直前に設定されたグラフィック状態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。使えるオプション (デフォルト : none) : ***backdropcolor*** (float 1 個か 3 個か 4 個のリスト。type=luminosity の場合のみ意味を持ちます) 透過グループテンプレートを構築する際の背景として用いたい色。float 値の数は、透過グループテンプレートを作成した際に用いた transparencygroup オプションの colorspace サブオプションに依存します (DeviceGray なら 1 個、DeviceRGB なら 3 個、DeviceCMYK なら 4 個)。デフォルト : それぞれのカラースペースにおける黒 ***template*** (テンプレートハンドル。必須) PDF_begin_template_ext( ) で作成された透過グループテンプレート。 ***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*** 透過グループのアルファ値を用い、色を無視します。 ***luminosity*** 透過グループの色を、1 要素の輝度値へ変換します。この場合、テンプレートは transparencygroup オプションで作成している必要があります。
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。PDF 1.4) 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

C++ Java ***void set_gstate(int gstate)***
Perl PHP ***set_gstate(int gstate)***
C ***void PDF_set_gstate(PDF *p, int gstate)***

グラフィック状態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状態オブジェクトのハンドル。

詳細 グラフィック状態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。この関数を複数回呼び出すと、グラフィック状態のオプションは蓄積されていきます。グラフィック状

態オブジェクトの中で明示的に設定されていないオプションについては、そのカレントの値のままになります。グラフィック状態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

7.3 座標系の変換

変換関数（*PDF_translate()*・*PDF_scale()*・*PDF_rotate()*・*PDF_align()*・*PDF_skew()*・*PDF_concat()*・*PDF_setmatrix()*・*PDF_initgraphics()*）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に対しては効力を持ちません。

C++ Java ***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 ***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 この関数は、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 ***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 ***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° / pi × atan2(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 ***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、−360° < *alpha, beta* < 360° の範囲にする必要があり、かつ、−270° ・−90° ・90° ・270° 以外にしなければなりません。

スコープ ページ・パターン・テンプレート・グリフ

C++ Java ***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、PostScript・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 この関数は、座標系に行列を適用します。これを使うと、もっとも一般的な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、この関数でなく、*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	***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 この関数は *PDF_concat( )* に似ています。ただし、カレント変換行列を捨てて、まったく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 パスの作成

この節に関連する値のキー名を表 7.5 に示します（19 ページ「2.2 パラメタ・オプション処理」参照）。これらのキー名は *PDF_set_value( )* では使えません。これらの値を設定するための API 関数が別にあるためです。

表 7.5 PDF_get_value( ) のキー一覧

キー	説明
currentx *currenty*	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表したもの）。スコープ：ページ・パターン・テンプレート・パス
ctm_a *ctm_b* *ctm_c* *ctm_d* *ctm_e* *ctm_f*	ベクトルグラフィックに対するカレント変換行列（CTM）の各要素。スコープ：ページ・パターン・テンプレート・パス

注 この節の関数を使った後は、必ず 137 ページ「7.5 描画とクリッピング」の関数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。

C++ Java ***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・パス・グリフ**。この関数は**パス**スコープを開始させます。

パラメタ *currentx・currenty*

C++ Java ***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 この関数は、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、この関数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* パラメタ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* パラメタの値によって決定されます。

スコープ **パス**

パラメタ *currentx・currenty*

C++ Java ***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、この関数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

パラメタ *currentx・currenty*

C++ Java ***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 この関数は、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。

スコープ ページ・パターン・テンプレート・パス・グリフ。この関数は**パス**スコープを開始させます。

パラメタ *currentx・currenty*

C++ Java ***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 この関数は、*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・パス・グリフ**。この関数は**パス**スコープを開始させます。

パラメタ *currentx・currenty*

C++ Java ***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、この関数は *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 ***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 この関数は、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。この関数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**

パラメタ *currentx・currenty*

C++ Java ***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 この関数は、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。

スコープ **ページ・パターン・テンプレート・パス・グリフ**。この関数は**パス**スコープを開始させます。

パラメタ *currentx・currenty*

C++ Java ***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)***

矩形を描きます。

x・y 矩形の左下隅の座標。

width・height 矩形の寸法。

詳細 この関数は、矩形を完全なサブパスとして、カレントパスに追加します。この関数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* パラメタの値によって決定される）の半分ずつが描かれます。

スコープ **ページ・パターン・テンプレート・パス・グリフ**。この関数は**パス**スコープを開始させます。

パラメタ *currentx・currenty*

C++ Java ***void closepath( )***
Perl PHP ***closepath( )***
C ***void PDF_closepath(PDF *p)***

カレントパスを閉じます。

詳細 この関数は、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**

パラメタ *currentx・currenty*

7.5 描画とクリッピング

注 この節の関数の多くは、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、これらの関数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 ***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**。この関数は**パス**スコープを終了させます。

C++ Java ***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 この関数は、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**。この関数は**パス**スコープを終了させます。

C++ Java ***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 この関数は、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* パラメタ参照）。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**。この関数は**パス**スコープを終了させます。

パラメタ *fillrule*

C++ Java ***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**。この関数は**パス**スコープを終了させます。

パラメタ *fillrule*

C++ Java ***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 この関数は、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。**この関数は**パス**スコープを終了させます。

パラメタ *fillrule*

C++ Java ***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 この関数は、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。

スコープ **パス。**この関数は**パス**スコープを終了させます。

C++ Java ***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 この関数は、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。**この関数は**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 ***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=move・line・curve・circular のときは、この点が新たにカレント点となります。

type 点の種別を、表 7.6 に従って指定します。

表 7.6 PDF_add_path_point( ) の点の種別一覧

キーワード	説明
circular	カレント点から新規点へ円弧を追加します。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。
control	ベジエ曲線の、または円弧の制御点。
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します。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。
line	カレント点から新規点へ線分を追加します。
move	新規サブパスまたは（書式が変わるとき、または異なるパス操作が用いられるとき）新規パスを開始します。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。
origin	絶対座標の新規原点。relative=true にすると、座標は、もっとも最近の座標に対して相対的になります。原点は任意の頻度で設定することができ、パスオブジェクト内には挿入されません。デフォルト：(0, 0)

optlist パス構築オプション群を指定したオプションリスト：

- 点 1 個に対する、表 7.7 に従ったパス計算・命名オプション群：*name・polar・radians・relative*
- サブパス 1 個に属性群を割り当てるための（*type=move* の場合のみ）、表 7.7 に従ったパス構築オプション群：*close・fill・round・stroke*
- 表 7.2 に従ったグラフィック書式オプション群（*type=move* 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。作成されたパスは、その後、*PDF_draw_path( )* などの関数で用いることができます。

1 個の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 : 書式が変わるとき（新しい色など）、または異なるパス操作オプションが用いられるとき（描線から塗へ）には必ず、新規パスが自動的に開始されます。

さらにパスはそれぞれ、1 個ないし複数のサブパスを含むことができます。1 個のサブパスは、*type=move* による点で開始し、次の *type=move* による点の前で、または点列の末尾の前で終了します。

すべてのサブパスは、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。すべてのパスは個別に塗られます。

スコープ　任意

表 7.7 PDF_add_path_point( ) のオプション一覧

オプション	説明
close	（論理値。type=move のみ）true にすると、サブパスは直線で閉じられます。デフォルト : 脚注参照[1]
fill	（論理値。type=move のみ）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 : 脚注参照[1]
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。
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
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 : 脚注参照[1]
round	（float。type=move のみ）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。デフォルト : 脚注参照[1]
stroke	（論理値。type=move のみ）true にすると、サブパスは描線されます。デフォルト : 脚注参照[1]

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。

C++ Java ***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関数（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。

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。
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群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群：
  align・attachmentpoint・boxsize・fitmethod・orientate・position・scale
- 表 7.8 に従ったパス操作・サブパス選択オプション群：
  clip・close・fill・round・stroke・subpaths
- 表 7.2 に従ったグラフィック書式オプション群。これらは、*fill・stroke* オプションに対してのみ意味を持ちます：
  dasharray・dashphase・fillcolor・fillrule・flatness・gstate・linecap・linejoin・linewidth・miterlimit・strokecolor

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。この関数は、*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状態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上書きしますが、*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、それを上書きすることは一切ありません。

スコープ ページ・パターン・テンプレート・グリフ

表 7.8 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一覧

オプション	説明
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。グラフィック書式オプションはすべて無視されます。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）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）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。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

C++ Java ***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関数（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 求める情報を表 7.9 に従って指定したキーワード。

表 7.9 PDF_info_path( ) のキーワード一覧

キーワード	説明
bounding-box	外接枠をユーザー座標で（参照点に対して相対的に）表したものを内容として持つパスのハンドル
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（ユーザー座標系における）。subpaths オプションは無視されます。
width・height	パスの外接枠の幅・高さをユーザー座標で表したもの。線幅とマイターリミットは無視されます。
x1・y1・x2・y2・x3・y3・x4・y4	外接枠矩形の i 番目の隅の位置（i=1, 2, 3, 4）を、参照点に対して相対的にユーザー座標で表したもの

optlist パス描画オプション群を指定したオプションリスト：
- 表 7.8 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 7.10 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 この関数は、*PDF_add_path_point( )* 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 7.10 PDF_info_path( ) のオプション一覧

オプション	説明
name	px または py 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関数（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。パスオブジェクトは、*PDF_end_document( )* で自動的に削除されるわけではないことに留意してください。

スコープ 任意

8 色関数

8.1 色と色空間の設定

クックブック 完全なコードサンプルがクックブックの color/starter_color トピックにあります。

この節に関連するパラメタのキー名を表 8.1 に示します（19 ページ「2.2 パラメタ・オプション処理」参照）。

表 8.1 PDF_get/set_parameter( ) の色関連キー一覧

キー	説明
preserveold-pantonenames	false なら、旧形式の Pantone スポットカラー名は、対応する新しい色名に変換されます。そうでないなら、そのままになります。デフォルト：false。スコープ：任意
spotcolorlookup	false なら、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。スコープ：任意

色空間　PDFlib のクライアントでは、パスとテキストの文字の内部を塗ったり描線したりするために使う色を指定することができます。色は、いくつかの色空間のいずれかで指定することができます（一覧の各項目の頭に、*PDF_setcolor( )* と色オプションに対するその色空間キーワードを示します）。

- *gray*：0 ＝黒から 1 ＝白までのグレー値。
- *rgb*：RGB の三つ組。すなわち、赤・緑・青の比率を指定する 0 から 1 までの値 3 個。*(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 で必要な範囲 0 ～ 1 へ縮小させるために、255 で割る必要があります。
  RGB 値を数値で指定せずに、RGB カラーを、その HTML 名または 16 進値で指定することもできます（12 ページ「色」参照）。

クックブック RGB カラー値の使用に関する完全なコードサンプルがクックブックの color/web_colornames トピックにあります。

- *cmyk*：シアン・マゼンタ・イエロー・黒を表す、0 ＝無色から 1 ＝ベタ色までの 4 個の CMYK 値。*(0, 0, 0, 0)* ＝白、*(0, 0, 0, 1)* ＝黒。これは RGB の指定と異なることに注意してください。
- *iccbasedgray/rgb/cmyk*：ICC ベースの色は、ICC プロファイルの助けを借りて指定します。
- *spot*：スポットカラー（分版色空間）：定義済みか、または任意に命名したカスタム色に、他の上記の色空間のいずれか 1 つで表した代替表現を与えたもの。これは一般に、オフセット印刷機でカスタム色（複数可）を使って印刷することを想定している文書を作成するために使います。濃度値（比率）は、0 ＝無色から 1 ＝そのスポットカラーの最大濃度までの範囲をとります。
- *lab*：D50 標準光源による CIE $L^*a^*b^*$ 色空間のデバイス独立色をとります。色は、範囲 0 ～ 100 の輝度値 1 個と、範囲 -128 ～ 127 の 2 個の色値 *a* と *b* で指定します。成分 *a* は

緑から赤／マゼンタまでの範囲をとり（負の値は緑を示し、正の値はマゼンタを示す）、成分 *b* は青から黄色までの範囲をとります（負の値は青を示し、正の値は黄色を示す）。

- *pattern*：任意のテキスト・ベクトル・画像グラフィックから成るオブジェクトによるタイリングパターン。
- シェーディング（スムーズブレンド）は、2 つの色の間の連続的遷移を与えるもので、他の色空間にもとづいています。シェーディングは、*PDF_shading( )* を使って作成することができます。
- インデックスカラーは、それ自体は本当の色空間ではなく、他の色空間の効率的なコード化です。これは、インデックス化された（パレットベースの）画像を取り込むと自動的に生成されますが、直接指定することはできません。

描線と塗りの操作のデフォルト色は黒です。

オプションリストでの色指定　12 ページ「色」に、オプションリストでの色データ型の説明と例を示しています。

C++ Java ***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)

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　指定する色値群に対して用いたい色空間を指定するか、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*・*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の後に 6 桁の 16 進数（例：*#FFC0CB*）。詳しくは 12 ページ「色」を参照。

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。

- *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。
- *lab*：*c1*・*c2*・*c3* で、D50 光源による CIE $L^*a^*b^*$ 色空間内のカラー値を指定します。*c1* で L^*（輝度）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( )* か *PDF_shading_pattern( )* によって返されたパターンハンドルを指定します。このパターンを使って塗りや描線を行うときは、カレント

の塗り色や描線色が適用されます。カレント色空間は他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色値と、*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。

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、*setcolor:iccprofilegray/rgb/cmyk* パラメタを使う前に、適切な ICC プロファイルを設定しておく必要があります（表 8.3 参照）。

PDF/X-1a：*colorspace=rgb・iccbasedgray/rgb/cmyk・lab* は許されません。

PDF/X-3：*iccbasedgray/rgb/cmyk・lab* カラーを使うなら、出力インテントの中に ICC プロファイルが必要です（この場合、標準名では不十分です）。

PDF/X-3・4・5：*colorspace=gray* にするなら、PDF/X 出力インテントがグレースケールか CMYK のデバイスでないときは、*PDF_begin_page_ext()* で *defaultgray* オプションを設定しておく必要があります。

colorspace=rgb にするなら、PDF/X 出力インテントが RGB デバイスでないときは、*PDF_begin_page_ext()* で *defaultrgb* オプションを設定しておく必要があります。

colorspace=cmyk にするなら、PDF/X 出力インテントが CMYK デバイスでないときは、*PDF_begin_page_ext()* で *defaultcmyk* オプションを設定しておく必要があります。

PDF/A：*colorspace=gray* にするなら、出力インテント（あらゆる種類の）が指定されていないときは、*PDF_begin_page_ext()* で *defaultgray* オプションを設定しておく必要があります。

colorspace=rgb にするなら、出力インテントが RGB デバイスでないときは、*PDF_begin_page_ext()* で *defaultrgb* オプションを設定しておく必要があります。

colorspace=cmyk にするなら、出力インテントが CMYK デバイスでないときは、*PDF_begin_page_ext()* で *defaultcmyk* オプションを設定しておく必要があります。

スコープ **ページ・パターン**（パターンの *painttype* が 1 の場合のみ）・**テンプレート・グリフ**（Type 3 フォントの *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。

パラメタ *setcolor:iccprofilegray/rgb/cmyk*

C++ Java ***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。この名前は最長 126 バイトに制限されています。スポットカラー名では、8 ビットキャラクタしか使えません。Unicode や null キャラクタを含んだ文字列は使えません。PDF/X-1a モードでは、PANTONE® カラーは使えません。

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、トンボに色をつける際に便利です。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。

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 カラーハンドル。以後の *PDF_setcolor( )* への呼び出しで、文書を終えるまで使えます。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。1 文書内のスポットカラーの数に制限はありません。

詳細 *spotname* が内蔵の色テーブルで既知のときは、*spotcolorlookup* パラメタが *true*（デフォルト）なら、与えたスポットカラー名が使われます。それ以外なら、カレント塗り色の色値（CMYK ないしそれ以外の）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。この代替色は、画面プレビューと低品位印刷でのみ使われます。色分版製版の際は、この代替値ではなく、与えたスポットカラー名が使われます。

spotname が、以前に *PDF_makespotcolor( )* を呼び出した時にもう使っていたものならば、戻り値は前回の呼び出しと同じになり、カレント色の反映もされません。

スコープ **ページ・パターン・テンプレート・グリフ**（Type 3 フォントの *colorized* オプションが *true* の場合のみ）・*document*。カスタム色を定義したいときは、カレント塗り色はスポットカラーかパターンであってはいけません。

パラメタ *spotcolorlookup*・*preserveoldpantonenames*

8.2 ICC プロファイル

C++ Java ***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。PDF/X-1 か PDF/X-3 が生成される場合、*usage=outputintent* ならば、表 8.5 に挙げる標準出力条件名の 1 つ（または *StandardOutputIntent* リソースカテゴリで定義した名前）を用いることもでき、その場合、対応する ICC プロファイルを埋め込む必要はありません。標準出力インテントの 1 つを PDF/X-4 か PDF/X-5pg で用いたいときは、対応する ICC プロファイルを、表 8.5 の参照名をリソース名として用いて、ICCProfile リソースと同様に設定する必要があります。

len （C 言語バインディングのみ）*profilename* 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。

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*（27 ページ「2.5 例外処理」参照）
- 表 8.2 に従ったプロファイル処理オプション群：*description*・*embedprofile*・*metadata*・*urls*・*usage*

表 8.2 PDF_load_iccprofile( ) のオプション

キー	説明・とりうる値
description	（文字列。usage=outputintent かつ非標準出力条件の場合のみ）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	（論理値。PDF/X-1 または PDF/X-3 かつ usage=outputintent の場合のみ。PDF/X-4 と PDF/X-5g では true を強制されます）PDF/X で標準出力インテントを profilename として与えているときにも、ICC プロファイルの埋め込みが強制されます。デフォルト：false
metadata	（オプションリスト。標準出力インテント・参照出力インテント、すなわち PDF/X-4p・PDF/X-5pg モードにおける usage= outputintent の場合には無視されます。PDF 1.4）プロファイルに対するメタデータを与えます（237 ページ「14.2 XMP メタデータ」参照）
urls	（文字列 1 個ないし複数のリスト。PDF/X-4p と PDF/X-5pg, の場合のみ。かつその場合には必須）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 群のリスト。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 8.2 PDF_load_iccprofile( ) のオプション

キー	説明・とりうる値
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 ベースの色空間を定義するために用いられるか、または画像に適用されます。入力・表示・出力デバイス（スキャナ・モニタ・プリンタ）のプロファイルのほか、色空間変換プロファイルを用いることもできます。 ***outputintent*** ICC プロファイルは、PDF/X か PDF/A で出力インテントを定義するために用いられます。PDF/X-4p・PDF/X-5pg モードでは、指定した出力インテント ICC プロファイルは埋め込まれず、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 を生成する際にはローカルで利用可能である必要があり、その PDF の利用者に対しては、文書を表示または印刷する際には利用可能となる必要があります。 PDF/X モードでは、usage=outputintent に対しては、出力デバイス（プリンタ）プロファイルのみ用いることができます。

戻り値 プロファイルハンドル。以後の *PDF_load_image( )* への呼び出しか、またはプロファイル関連のパラメタの設定で使えます。*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。返されたプロファイルハンドルは、複数の PDF 文書にわたって再利用することはできません。また、*usage=outputintent* にしているときは、返されるハンドルを画像に適用することはできません。1 文書内の ICC プロファイルの数に制限はありません。関数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 *usage=iccbased* にすると、プロファイル検索戦略に従って、名前付きプロファイルが検索されます。プロファイルが見つかったときは、それが適切かどうか検査されます（色要素の数等）。*sRGB* プロファイルは、つねに内部的に得られるので、定義してはいけません。読み込む ICC プロファイルは、PDF 1.4 の場合は 2.x 以下の ICC バージョンに、PDF 1.5 以上の場合は 4.x 以下の ICC バージョンに準拠している必要があります。プロファイルは、グレー・RGB・CMYK・Lab のいずれかの色空間で指定可能です。

PDF/X：出力インテントは、この関数を使うか、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-1・PDF/X-3：*usage=outputintent* にすると、名前付きプロファイルは、まず標準出力インテントの内部リスト内で、次いで、ユーザー設定出力インテント群のリスト内で検索されます。与えられた *profilename* が標準出力インテントであることがわかったときは、ICC プロファイルは必要とされず、その名前だけが出力インテントとして PDF 出力に書き込まれます。その名前が標準出力インテント識別子でないことがわかったときは、それはプロファイル名として扱われ、対応する ICC プロファイル（*ICCProfile* リソースカテゴリでマップすることも可能）が出力インテントとして PDF に埋め込まれます。

PDF/A：出力インテントは、この関数を使うか、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。ただし、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。

スコープ **文書**。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* オプションを *iccbased* にしているときは、次のスコープも許されます：**ページ・パターン・テンプレート・グリフ**。

パラメタ 表 8.3・表 8.4 参照。

表 8.3 PDF_get/set_parameter( ) のキー一覧（19 ページ「2.2 パラメタ・オプション処理」参照）

キー	説明
ICCProfile StandardOutputIntent	UPR ファイルの各カテゴリと同じ内容の、同名のリソースファイル行。複数回呼び出すと、内部リストに新たな項目が追加されます（表 2.3 の resourcefile も参照）。スコープ：任意

表 8.4 PDF_get/set_value( ) のキー一覧（19 ページ「2.2 パラメタ・オプション処理」参照）

キー	説明
icccomponents	修飾子で与えるハンドルで参照する ICC プロファイルの色要素の数
setcolor:icc-profilegray	PDF_setcolor( ) で使うためのグレー色空間を指定する ICC プロファイル。スコープ：文書・ページ・パターン・テンプレート・グリフ
setcolor:icc-profilergb	PDF_setcolor( ) で使うための RGB 色空間を指定する ICC プロファイル。スコープ：文書・ページ・パターン・テンプレート・グリフ
setcolor:icc-profilecmyk	PDF_setcolor( ) で使うための CMYK 色空間を指定する ICC プロファイル。スコープ：文書・ページ・パターン・テンプレート・グリフ

表 8.5 PDF/X の標準出力インテント一覧（詳しくは www.color.org を参照）

名前	出力処理
米国のための *CGATS* 規格群（*www.npes.org/standards/tools.html*）	
CGATS TR 001	米国における SWOP（出版）印刷：ANSI CGATS.6
CGATS TR 002	米国における SNAP 印刷
CGATS TR 003	米国 グレード 3 コート出版用紙上への SWOP 校正および印刷
CGATS TR 005	米国グレード 5 コート出版用紙上への SWOP 校正および印刷
CGATS TR 006	米国グレード 1 コート紙上への GRACoL 校正および印刷
米国のための *GRACoL・SWOP* 規格群（*www.gracol.org*）（*CGATS* グループでも入手可）	
GRACoL2006_Coated1	GRACoL グレード 1 コート紙、ISO 12647-2 用紙タイプ 1。ISO 12647-2 特性群に準拠し、スムーズ化され、G7 NPDC 2006 に同調（K50 が CMY50,40,40 より若干明るい）されたデータ
SWOP2006_Coated3	SWOP グレード 3 コート紙、ISO 2846-1 インキ群。SWOP に準拠し、G7 ニュートラル印刷密度曲線に同調された特性
SWOP2006_Coated5	SWOP グレード 5 コート紙、ISO 2846-1 インキ群。SWOP に準拠し、G7 ニュートラル印刷密度曲線に同調された特性
***FOGRA* 規格群（*www.fogra.org*）**	
FOGRA27	ISO/DIS 12647-2:2004、ISO 12647-2 に従ったオフセット商業および特殊印刷、ポジ版、用紙タイプ 1 または 2（グロス紙またはマットコートオフセット紙、115 g/m^2）、スクリーン線数 60/cm。
FOGRA28	ISO/DIS 12647-2:2004、ISO 12647-2 に従ったオフセット商業および特殊印刷、ポジ版、用紙タイプ 3（コートウェブ紙、60 g/m^2）、スクリーン線数 60/cm。
FOGRA29	ISO/DIS 12647-2:2004、ISO 12647-2 に従ったオフセット商業および特殊印刷、ポジ版、用紙タイプ 4（上質白色オフセット紙、120 g/m^2）、スクリーン線数 60/cm。
FOGRA30	ISO/DIS 12647-2:2004、ISO 12647-2 に従ったオフセット商業および特殊印刷、ポジ版、用紙タイプ 5（上質、やや黄味、オフセット紙、115 g/m^2）、スクリーン線数 60/cm。
FOGRA31	ISO/DIS 12647-2:2003、ISO 12647-2 に従った連続紙印刷、ポジ版、用紙タイプ 2（マットコートオフセット紙、115 g/m^2）、スクリーン線数 60/cm。

表 8.5 PDF/X の標準出力インテントー覧（詳しくは www.color.org を参照）

名前	出力処理
FOGRA32	ISO/DIS 12647-2:2004、ISO 12647-2 に従った連続紙印刷、ポジ版、用紙タイプ 4（白色上質オフセット紙、80 g/m^2）、スクリーン線数 60/cm。
FOGRA33	ISO/DIS 12647-2:2004、ISO 12647-2 に従った連続紙印刷、ポジ版、用紙タイプ 2（マットコートオフセット紙、115 g/m^2）、スクリーン線数 54/cm。
FOGRA34	ISO/DIS 12647-2:2004、ISO 12647-2 に従った連続紙印刷、ポジ版、用紙タイプ 4（白色上質オフセット紙、120 g/m^2）、スクリーン線数 60/cm。
FOGRA35	ISO/DIS 12647-2:2004、ISO 12647-2 に従った連続紙印刷、ネガ版、用紙タイプ 2（マットコートオフセット紙、115 g/m^2）、スクリーン線数 54/cm。
FOGRA36	ISO/DIS 12647-2:2004、ISO 12647-2 に従った連続紙印刷、ネガ版、用紙タイプ 4（白色上質オフセット紙、120 g/m^2）、スクリーン線数 54/cm。
FOGRA37	ISO/DIS 12647-2:2004、ISO 12647-2 に従った連続紙印刷、ネガ版、用紙タイプ 2（マットコートオフセット紙、115 g/m^2）、スクリーン線数 60/cm。
FOGRA38	ISO/DIS 12647-2:2004、ISO 12647-2 に従った連続紙印刷、ネガ版、用紙タイプ 4（白色上質オフセット紙、120 g/m^2）、スクリーン線数 60/cm。
FOGRA39	ISO 12647-2:2004/Amd 1、ISO 12647-2 に従ったオフセット商業および特殊印刷、用紙タイプ 1 または 2（グロス紙またはマットコートオフセット紙、115 g/m^2）、スクリーン線数 60/cm。
FOGRA40	ISO 12647-2:2004、ISO 12647-2 に従ったオフセット商業および特殊印刷、強光沢（SC）紙、60 g/m^2、スクリーン線数 60/cm。
FOGRA41	ISO 12647-2:2004、ISO 12647-2 に従ったオフセット印刷、用紙タイプ：MFC（機械仕上げコート紙）、濃淡値強調曲線 B（CMY）および C（K）。
FOGRA42	ISO 12647-2:2004、ISO 12647-2 に従ったオフセット印刷、用紙タイプ：SNP（標準新聞用紙）、濃淡値強調曲線 C（CMY）および D（K）。
FOGRA43	ISO 12647-2:2004/Amd 1、ISO 12647-2 に従ったオフセット印刷、用紙タイプ 1 または 2（コートアート紙）115 g/m^2、非周期スクリーン処理、濃淡値強調曲線 F（CMYK）.。
FOGRA44	ISO 12647-2:2004/Amd 1、ISO 12647-2 に従ったオフセット印刷、用紙タイプ 4（上質オフセット紙）115 g/m^2、非周期スクリーン処理、濃淡値強調曲線 F（CMYK）。
新聞用紙のための *IFRA* 規格群（*www.ifra.com*）	
IFRA26	ISO/DIS 12647-3:2004、コールドセットオフセット印刷、接点露出ネガ圧版またはコンピュータトゥプレート（調子値増大 26%）、新聞用紙、スクリーン線数 40/cm。
IFRA30	ISO/DIS 12647-3:2004、コールドセットオフセット印刷、接点露出ネガ圧版またはコンピュータトゥプレート（調子値増大 30%）、新聞用紙、スクリーン線数 40/cm。（主として米国に適応）
Eurostandard System Brunner（www.systembrunner.com）	
EUROSB104	Eurostandard System Brunner 仕様に従ったオフセット印刷、ISO 12647-2:2004 許容範囲内、用紙タイプ：コート紙／セミマット紙、115 g/m^2、TVI 15%,、スクリーン線数 60/cm、詳しくは仕様書を参照。
EUROSB204	Eurostandard System Brunner 仕様に従ったオフセット印刷、ISO 12647-2:2004 許容範囲内、軽量コート木材パルプ紙、80 g/m^2、TVI 15%,、スクリーン線数 60/cm、詳しくは仕様書を参照。

表 8.5 PDF/X の標準出力インテント一覧（詳しくは www.color.org を参照）

名前	出力処理
日本の規格群	
JC200103	Japan Color 2001 コート紙：ISO 12647-2:2004、枚葉オフセット印刷、ポジ版、用紙タイプ 3（コート紙、105 g/m^2）、スクリーン線数 69/cm。
JC200104	Japan Color 2001 上質紙：ISO 12647-2:2004、枚葉オフセット印刷、ポジ版、用紙タイプ 4（上質紙、105 g/m^2）、スクリーン線数 69/cm。
JCN2002	Japan Color 2002 新聞印刷用：ISO/DIS 12647-3:2004、コールドセットオフセット印刷、ネガ版、新聞用紙、スクリーン線数 39/cm。
JCW2003	Japan Color 2003 商業オフ輪用：ISO 12647-2:2004、ヒートセットオフセット印刷、ポジ版、用紙タイプ 3（コート紙、70 g/m^2）、スクリーン線数 69/cm。

8.3 パターンとシェーディング

C++ Java ***int begin_pattern(double width, double height, double xstep, double ystep, int paintype)***
Perl PHP ***int begin_pattern(float width, float height, float xstep, float ystep, int paintype)***
C ***int PDF_begin_pattern(PDF *p,***
double width, double height, double xstep, double ystep, int paintype)

パターン定義を開始します。

width・height パターンの外接枠の寸法を、ポイント単位で指定します。

xstep・ystep 何らかのオブジェクトを描線したり塗ったりする際に、パターンを繰り返し配置する時の変位。これは、多くの用途で、パターンのそれぞれ *width・height* と同じ値に設定します。

paintype このパラメタは、パターンが自分自身で色指定を含んでいるか、それともパターンが塗りや描線に使われる際にカレントの塗り色や描線色で着色されるステンシルとして使われるかを指定します。

- パターンが、*PDF_setcolor()* への 1 度ないし複数回の呼び出しで着色されている場合や、画像または取り込み PDF ページを配置している場合は、*paintype=1* を用いる必要があります。
- パターンがまったく色指定を含まない場合は、*paintype=2* を用いる必要があります。この場合、パターンが塗りや描線に使われる際には、カレントの塗り色や描線色が適用されます。*paintype=2* の場合、画像マスクを使うこともできます。パターンを使う前に、*PDF_setcolor()* を呼び出してカレント色を設定する必要があります。このカレント色の色空間は、それ自体が他のパターンに基づかないものでなければなりません。

paintype で誤った値を与えると、望ましくない動作が起こる可能性があります。

戻り値 パターンハンドル。以後の *PDF_setcolor()* への呼び出しで、**文書**スコープが終わるまで使えます。

詳細 この関数は、テキスト・グラフィック・色状態パラメタをすべてデフォルトにリセットして、グローバルな *topdown* パラメタに従って座標系を定義します。パターン定義の最中は、ハイパーテキスト関数と、画像を開く関数は使ってはいけませんが、テキスト・グラフィック・色関数（定義中のパターン以外で）はすべて使えます。

スコープ **文書・ページ**。この関数は**パターン**スコープを開始させます。対応する *PDF_end_pattern()* と必ず対にして呼び出す必要があります。

パラメタ *topdown*

C++ Java ***void end_pattern()***
Perl PHP ***end_pattern()***
C ***void PDF_end_pattern(PDF *p)***

パターン定義を完了します。

スコープ **パターン**。この関数は**パターン**スコープを終了させます。対応する *PDF_begin_pattern()* と必ず対にして呼び出す必要があります。

C++ Java ***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（要 PDF 1.4）。

shading *PDF_shading()* によって返されたシェーディングハンドル。

optlist シェーディングパターンのグラフィック書式を表 7.2 に従って記述したオプションリスト：*gstate*

戻り値 パターンハンドル。以後の *PDF_setcolor()* への呼び出しで、**文書**スコープが終わるまで使えます。

詳細 この関数を使うと、任意のオブジェクトをシェーディングで塗ることができます。そのためには、*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、それから、このシェーディングにもとづいて *PDF_shading_pattern()* を使ってパターンを定義する必要があります。そして最終的に、このパターンハンドルを *PDF_setcolor()* に与えれば、カレント色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 **文書・ページ・フォント**

C++ Java ***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、シェーディングオブジェクトにもとづいて塗ります（要 PDF 1.4）。

shading *PDF_shading()* によって返されたシェーディングハンドル。

詳細 この関数を使えば、*PDF_shading_pattern()* と *PDF_setcolor()* を使わなくても、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）、一般にこの関数は、*PDF_clip()* と組み合わせて使います。

スコープ **ページ・パターン**（パターンの *painttype* が 1 の場合のみ）・**テンプレート・グリフ**（Type 3 フォントの *colorized* オプションが *true* の場合のみ）・**文書**

C++ Java ***int shading(String shtype, double x0, double y0, double x1, double y1, double c1, double c2, double c3, double c4, String optlist)***

Perl PHP ***int shading(string shtype, float x0, float y0, float x1, float y1, float c1, float c2, float c3, float c4, string optlist)***

C ***int PDF_shading(PDF *p, const char *shtype, double x0, double y0, double x1, double y1, double c1, double c2, double c3, double c4, const char *optlist)***

カレント塗り色から別の色へのブレンドを定義します（要 PDF 1.4）。

shtype シェーディングの種類。線形シェーディングを表す *axial* か、または円形の放射シェーディングを表す *radial* にする必要があります。

x0・y0・x1・y1 線形シェーディングの場合、*(x0, y0)* と *(x1, y1)* はシェーディングの始点と終点の座標。放射シェーディングの場合、これらの点は開始円と終了円の中心を指定します。

c1・c2・c3・c4 シェーディングの終点の色値であり。*PDF_setcolor( )* の色引数と同様にカレント塗り色空間で解釈されます。カレント塗り色空間がスポットカラーのときは *c1* は無視され、*c2* で色の濃さを指定します。

optlist シェーディングの諸特性を表 8.6 に従って記述したオプションリスト。次のオプションが使えます : *antialias・boundingbox・domain・extend0・extend1・N・r0・r1・startcolor*

戻り値 シェーディングハンドル。以後の *PDF_shading_pattern( )・PDF_shfill( )* 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 カレント塗り色が開始色として使われます。それはパターンにもとづいた色であってはいけません。

スコープ **文書・ページ・フォント**

表 8.6 PDF_shading( ) のオプション一覧

オプション	説明・とりうる値
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	（矩形）シェーディングの外接枠をユーザー座標で定義した矩形。この外接枠は、シェーディングが塗ら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。
domain	（float 2 個のリスト）媒介変数 t の限界値を指定した 2 個の数値。この変数は、色勾配が軸の始点と終点の間を変動するにつれて、この 2 値の間を線形的に変動すると考えられます。デフォルト : {0 1}
extend0	（論理値）シェーディングを始点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より先へ拡げるかどうかを指定します。デフォルト : false
N	（float）色遷移関数に対する累乗指数。> 0 にする必要があります。デフォルト : 1
r0	（float。放射シェーディングのみ。その場合は必須）開始円の半径
r1	（float。放射シェーディングのみ。その場合は必須）終了円の半径

表 8.6 PDF_shading( ) のオプション一覧

オプション	説明・とりうる値
startcolor	（色）開始点の色。このオプションは、この関数をカレント色から独立にするのに有用でしょう。 デフォルト：カレント塗り色

9 画像・テンプレート関数

この節に関連するパラメタと値のキー名を表 9.1・表 9.2 に示します（19 ページ「2.2 パラメタ・オプション処理」参照）。

表 9.1 PDF_get/set_parameter( ) の画像関連キー一覧

キー	説明
honoriccprofile	画像内に埋め込まれている ICC カラープロファイルを読み込んで、それを画像データに適用します。デフォルト：true
renderingintent	画像に対するレンダリングインテント。デフォルト：Auto。 ***Auto*** PDF ファイルの中でレンダリングインテントを一切指定せずに、デバイスのデフォルトのインテントを使います。代表的用途：未知のケース ***AbsoluteColorimetric*** デバイスの白色点（紙白等）に対する補正を一切行いません。色域外にある色は、デバイスの色域の中のもっとも近い値にマップされます。代表的用途：特色の厳密な再現。それ以外の用途には推奨されません ***RelativeColorimetric*** 色データは、白色点を互いにマップし、色を若干シフトさせて、デバイスの色域にスケールされます。代表的用途：線画 ***Saturation*** 色の彩度を保つため、色値はシフトされることがあります。代表的用途：ビジネスグラフィック ***Perceptual*** 自然な見ばえになるよう、色間の関係を保つため、色域内外両方の色を変更します。代表的用途：スキャン画像

表 9.2 PDF_get/set_value( ) の画像関連キー一覧

キー	説明
imagewidth ***imageheight***	非推奨。PDF_info_image( ) で imagewidth・imageheight キーを指定してください。
image:iccprofile	非推奨。PDF_info_image( ) で iccprofile キーを指定してください。
orientation	非推奨。PDF_info_image( ) で orientation キーを指定してください。
resx・resy	非推奨。PDF_info_image( ) で resx・resy キーを指定してください。

9.1 画像

クックブック 完全なコードサンプルがクックブックの `images/starter_image` トピックにあります。

C++ Java ***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（これは CCITT・raw 画像に対しては不可能です）。画像形式を明示的に文字列 *bmp*・*ccitt*・*gif*・*jbig2*（PDF 1.4 以上）・*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。種別 *ccitt* は、CCITT 圧縮された画像データの入った TIFF ファイルとは別物です。

filename （名前文字列。グローバルな *filenamehandling* オプションまたはパラメタに従って解釈されます、表 2.2 参照）一般に、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）、*searchpath* 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`・`.ccitt`・`.g3`・`.g4`・`.fax`・`.gif`・`.jbig2`・`.jb2`・`.jpg`・`.jpeg`・`.jpx`・`.jp2`・`.jpf`・`.jpm`・`.j2k`・`.png`・`.raw`・`.tif`・`.tiff`

len （C 言語バインディングのみ）*filename* 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。

optlist 画像関連の特性を表 9.3 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：

- 一般オプション群：*errorpolicy*（表 2.6 参照）・*hypertextencoding*（表 12.1 参照）
- 色関連オプション群：*colorize*・*honoriccprofile*・*iccprofile*・*invert*・*renderingintent*
- クリッピング・マスク・等価オプション群：***alphachannelname***・*clippingpathname*・*honorclippingpath*・*ignoremask*・*mask*・*masked*
- 画像を使うための特殊な PDF 機能群：*georeference*・*iconname*・*template*
- JBIG2 画像のためのオプション群：*copyglobals*・*imagehandle*
- 画像データを処理するためのオプション群：*cascadedflate*・*ignoreorientation*・*inline*・*page*・*passthrough*
- その他のオプション群：*interpolate*・*layer*・*metadata*・*OPI-1.3*・*OPI-2.0*

戻り値 画像ハンドル。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。関数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 この関数は、*imagetype* パラメタで決定される、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。この関数は、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。

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()* 参照）、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。

imagetype = *raw* か *ccitt* の場合には、*width*・*height*・*components*・*bpc* オプションを与える必要があります。なぜなら、それらを PDFlib は画像データから取り出すことができないからです。画像に本当に合ったオプション値を与えるのは、ユーザー側の役割です。そうしないと、破損した PDF 出力が生成されることがあり、Acrobat から *Insufficient data for an Image* メッセージが出される可能性があります。

imagetype = *raw* の場合には、与える画像データのサイズは *[width × components × bpc ÷ 8] × height* バイトに等しくなければなりません。ここで角カッコ内は小数点以下を切り上げて整数とします。画像のサンプルは、標準の PostScript/PDF の順序であると見なされます。すなわち、上から下、左から右という順序です（座標変換が適用されていないものとして）。16 ビットのサンプルは、最上位バイトを先頭にして（ビッグエンディアン、または「Mac」バイト順序と呼ばれます）与える必要があります。ピクセル値の極性については 145 ページ「色空間」で解説したものと同様です。*bpc* が 8 未満であっても、各ピクセル行はバイトの区切りで開始され、そしてバイト内では色値は左から右へ格納されていきます。画像のサンプルは必ず独立しています。すなわち、最初のピクセルのすべての色値が最初に与えられて、次に 2 番目のピクセルのすべての色値が与えられ、以下同様です。

PDF/X-1a：RGB 画像は許されません。

PDF/X-3・4・5：グレースケール画像の場合は、出力インテントがグレースケールか CMYK のデバイスでないときは、*PDF_begin_page_ext()* で *defaultgray* オプションを設定しておく必要があります。RGB 画像の場合は、出力インテントが RGB デバイスでないときは、*PDF_begin_page_ext()* で *defaultrgb* オプションを設定しておく必要があります。CMYK 画像の場合は、出力インテントが CMYK デバイスでないときは、*PDF_begin_page_ext()* で *defaultcmyk* オプションを設定しておく必要があります。JBIG2 画像は許されません。

PDF/A：グレースケール画像の場合は、出力インテントが（いずれの種別も）指定されていないときは、*PDF_begin_page_ext()* で *defaultgray* オプションを設定しておく必要があります。

RGB 画像の場合は、出力インテントが RGB デバイスでないときは、*PDF_begin_page_ext()* で *defaultrgb* オプションを設定しておく必要があります。

CMYK 画像の場合は、出力インテントが CMYK デバイスでないときは、*PDF_begin_page_ext()* で *defaultcmyk* オプションを設定しておく必要があります。

スコープ *inline* オプションを与えていないときは、スコープは**文書・ページ・フォント**であり、この関数は、対応する *PDF_close_image()* と必ず対にして呼び出す必要があります。**文書**ス

コープか**フォント**スコープで画像を読み込むと、**ページ**スコープで読み込む場合に比べて、若干出力サイズを小さくすることができます。
inline オプションを与えているときは、スコープは**ページ・パターン・テンプレート・グリフ**であり、*PDF_close_image( )* を呼び出してはいけません。

パラメタ　表 9.1・表 9.2 参照

表 9.3 PDF_load_image( ) のオプション一覧

オプション	説明
alphachannel-name	（名前文字列。TIFF 画像では不可。ignoremask=true にすると無視されます）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
bitreverse	（論理値。imagetype=ccitt のみ）true にすると、圧縮データ内のすべてのバイトのビット反転をします。デフォルト：false
bpc	（整数。imagetype=raw のみ。その場合は必須）色要素あたりのビット数。1・2・4・8 のいずれかにする必要があります。PDF 1.5 では bpc=16 も使えます。
cascadedflate	（論理値。imagetype=jpeg のみ）true にすると、JPEG 圧縮された画像データに、Flate 圧縮の追加のレイヤーが適用されます。これは、同色の広い領域を持つ画像等、特定の場合には、出力ファイルサイズを減らす可能性があります。ただし、多くのタイプの画像内容について、このオプションはファイルサイズを減らさず、むしろ出力が大きくなる可能性もあります。デフォルト：false
clipping-pathname	（文字列。imagetype=tiff・jpeg のみ。honorclippingpath=false にすると無視されます）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	（スポットカラーハンドル。iccprofile オプションを与えると無視されます）スポットカラーハンドルで画像に着色します。このハンドルは、PDF_makespotcolor( ) で取得しておく必要があります。画像は、白黒かグレースケールの画像でなければなりません。
components	（整数。imagetype=raw のみ。その場合は必須）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。
copyglobals	（キーワード、または整数のリスト。imagetype=jbig2 のみ）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。
georeference	（オプションリスト。PDF 1.7ext3）地理空間測量に利用するために画像に関連づけられる地球ベース座標系の記述を指定します。詳しくは 232 ページ「13.2 地理空間機能」を参照。
height	（整数。imagetype=raw・ccitt のみ。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。

表 9.3 PDF_load_image( ) のオプション一覧

オプション	説明
honor-clippingpath	（論理値。imagetype=tiff・jpeg のみ）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・png・tiff のみ。colorize オプションを与えているときは false に設定されます）埋め込まれている ICC プロファイル（もしあれば）を読み込んで、それを画像に適用します。デフォルト：グローバルな honoriccprofile パラメタの値。
iccprofile	（ICC ハンドル。imagetype=jpeg・jbig2・png・tiff のみ）画像に適用される ICC プロファイルのハンドル。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。
iconname	（ハイパーテキスト文字列。inline=true にすると無視されます。template=true を強制します）画像に名前を付けて、JavaScript で参照できるようにします。たとえば、画像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。
ignoremask	（論理値。PDF/X-1・PDF/X-3・PDF/A モードの場合、アルファチャンネルを持つ画像に対しては 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 のみ）画像内の orientation タグ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：false
imagehandle	（画像ハンドル。imagetype=jbig2 のみ）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=ccitt・jpeg・raw のみ）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）。このオプションを用いることは、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 にしていなければ不可）画像を反転させます（明るい色と暗い色を入れ換え）。これを使うと、アプリケーションによって異なって解釈される画像を、適切に扱うことができます。デフォルト：false
K	（整数。imagetype=ccitt のみ）圧縮方式選択のための CCITT のパラメタ。デフォルト：0 ***-1*** G4 圧縮 ***0*** 一次元 G3 圧縮（G3-1D） ***1*** 一・二次元混合圧縮（G3, 2-D）
layer	（レイヤーハンドル。PDF 1.5）画像を属させたいレイヤー。ただし、この画像を配置する前に 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。この画像を配置する前に PDF_begin_layer( ) を呼び出してレイヤーを有効にさせると、この画像の layer オプションは上書きされます。この画像の layer オプションが上書きされないようにするには、この画像を配置する前に PDF_end_layer( ) を呼び出してください。

表 9.3 PDF_load_image( ) のオプション一覧

オプション	説明
mask	（論理値。インデックスカラーを含め、色要素が 1 個の画像のみ）画像はマスクとして使われます。これは 1 ビットのマスクでは必須ですが、ピクセルあたりのビット数が 1 より大きいマスクでは必須ではありません。ただし、ビット数が 1 より大きいマスクを使うには PDF 1.4 が必要です。デフォルト：false。マスクには 2 通りの使い方があります： ▸ 他の画像をマスク：返された画像ハンドルは、以後の呼び出しで他の画像を開くときに使うことができ、masked オプションに与えることができます。 ▸ 着色された透過画像を配置：画像内の、0 のビットのピクセルを透過として扱い、1 のビットのピクセルをカレント塗り色で着色します。 このオプションは ignoremask=true を強制します。なぜなら、マスクとして用いられる画像は、それ自身が内部マスクを持つことはできないからです。
masked	（画像ハンドル）カレント画像に対してマスクとして適用したい画像の画像ハンドル。この画像ハンドルは、あらかじめ PDF_load_image( ) を呼び出して返され、まだ閉じていないものです。 PDF 1.3 互換モードでは、マスクハンドルは 1 ビット画像を参照している必要があり、かつ、mask オプションを指定して読み込んである必要があります。PDF/A・PDF/X-1/3 モードでは、このオプションは 1 ビットマスクの場合のみ許されます。
metadata	（オプションリスト。PDF 1.4）画像に対するメタデータを与えます（237 ページ「14.2 XMP メタデータ」参照）。
OPI-1.3	（オプションリスト。PDF/A・PDF/X では不可）OPI 1.3 の PostScript コメントをオプション名として入れたオプションリスト。以下の項目は必須です：ALDImageFilename（文字列[1]）・ALDImageDimensions（整数のリスト）・ALDImageCropRect（整数による矩形）・ALDImagePosition（float のリスト） 以下の項目は必須ではありません： ALDImageID（文字列）・ALDObjectComments（文字列）・ALDImageCropFixed（矩形）・ALDImageResolution（float のリスト）・ALDImageColorType（キーワード。Process・Spot・Separation のうちのいずれか。デフォルト：Spot）・ALDImageColorType（範囲 0 ～ 1 の色値 4 個と色名 1 個のリスト）・ALDImageTint（float）・ALDImageOverprint（論理値）・ALDImageType（整数のリスト）・ALDImageGrayMap（整数のリスト）・ALDImageTransparency（論理値）・ALDImageAsciiTag（整数と文字列との対のリスト） normalizefilename サブオプションは、ファイル名の処理を制御します。true にすると、ファイル名は PDF リファレンスが義務付けているように正規化されます。false にすると、ファイル名は何も変更されずに出力へコピーされます。後者は、正規化されたファイル名を適切に処理しないいくつかの OPI サーバを扱う際に有用かもしれません。デフォルト：false
OPI-2.0	（オプションリスト。PDF/A・PDF/X では不可）OPI 2.0 の PostScript コメントをオプション名として入れたオプションリスト。以下の項目は必須です：ImageFilename（文字列[1]） 以下の項目は、両方入れるか、またはどちらも入れない必要があります： ImageCropRect（矩形）・ImageDimensions（float のリスト） 以下の項目は必須ではありません： MainImage（文字列）・TIFFASCIITag（整数と文字列との対のリスト）・ImageOverprint（論理値）・ImageInks（文字列 full_color か、または文字列 registration か、または文字列 monochrome とインキ名・濃度をそれぞれ指定した文字列・float 対（複数可）とを入れたリスト）・IncludedImageDimensions（整数のリスト）・IncludedImageQuality（整数で値 1・2・3 のいずれか） normalizefilename も使えます（OPI-1.3 参照）。
page	（整数。imagetype=gif・jbig2・tiff のみ。それ以外の形式で使うときは 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

表 9.3 PDF_load_image( ) のオプション一覧

オプション	説明
passthrough	（論理値。imagetype=tiff・jpeg のみ）TIFF・JPEG 画像データの処理を制御します。 ***tiff***　（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***　（デフォルト：false）false にすると、PDFlib は JPEG 画像データを変換して、Acrobat との互換性のためのデータをクリーンアップします。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 画像に対しては、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）画像に対するレンダリングインテント。とりうるキーワードとその意味については表 9.1 参照。デフォルト：グローバルな renderingintent パラメタの値
template	（論理値）true にすると、プレーンな Image XObject ではなく、Form XObject（PDFlib ではテンプレートと呼びます）の中に埋め込んで PDF Image XObject を生成します。これは、1 つの画像だけでできたフォームフィールドアイコンのためのテンプレートを作成するのに有用でしょう。これはまた、OPI-1.3 か OPI-2.0 オプションを使っているときは、ある種の OPI サーバとの互換性のために必要です。デフォルト：false。スコープ：文書
width	（整数。imagetype=raw・ccitt 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

1. Windows ユーザーは、オプションリスト内で結果的にパスの中に 1 個のバックスラッシュを作成するには、2 個のバックスラッシュキャラクタを並べる必要があることに留意してください（9 ページ「よくある落とし穴」参照）。

C++ Java ***void close_image(int image)***
Perl PHP ***close_image(int image)***
C ***void PDF_close_image(PDF *p, int image)***

画像を閉じます。

image　*PDF_load_image( )* で取得した有効な画像ハンドル。

詳細　この関数は、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、この関数で閉じた後はもう使えません。なぜならこの関数は、画像との PDFlib 内部の関連を切るからです。

スコープ　**文書・ページ・フォント**。*inline* オプションを使っていないときは、対応する *PDF_load_image( )* と必ず対にして呼び出す必要があります。

C++ Java ***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( )* 関数で取得した有効な画像またはテンプレートハンドル。

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・処理オプション群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群：
  boxsize・blind・dpi・fitmethod・matchbox・orientate・position・rotate・scale・showborder
- 表 9.4 に従った画像処理のためのオプション群：
  adjustpage・gstate・ignoreclippingpath・ignoreorientation

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。

スコープ **ページ・パターン**（パターンの paintype が 1 か、または画像がマスクの場合のみ）・**テンプレート・グリフ**（Type 3 フォントの *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）。この関数は、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 9.4 PDF_fit_image( )・PDF_fit_pdi_page( )・PDF_fill_*block( ) のオプション一覧

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ときは不可）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 は関数の引数です。MediaBox が調整され、それ以外のボックス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まれます。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 画像から水平と垂直の帯が切り取られます。 デフォルト：false
gstate	（グラフィック状態ハンドル）PDF_create_gstate( ) で取得したグラフィック状態のハンドル。このグラフィック状態は、この関数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：グラフィック状態なし（すなわち、カレント状態が用いられます）
ignore-clippingpath	（論理値。TIFF・JPEG 画像のみ）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 画像のみ）画像内の orientation タグ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 ***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)***

画像を組版する際の、寸法などの画像特性群を取得します。

image 関数 *PDF_load_image()*・*PDF_begin_template_ext()* のいずれかで取得した画像・テンプレートいずれかの有効なハンドル。

keyword 求める情報を表 9.5 に従って指定したキーワード。

optlist *PDF_fit_image()* 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定するのに関係のないオプションは無視されます。

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、この関数は 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）、この関数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。

詳細 この関数は、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 ページ・パターン・テンプレート・グリフ・文書・パス

表 9.5 PDF_info_image( ) のキーワード一覧

キーワード	説明
boundingbox	画像の外接枠のパスハンドル
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）
filename	画像ファイルの名前（あてはまる場合は searchpath ディレクトリを含む）
fitscalex・fitscaley	与えたオプション群に従って画像を枠にはめ込んだ結果の拡縮倍率
height	与えたオプション群に従った画像の高さをユーザー座標で表したもの
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）
imageheight	画像の高さをピクセル単位で表したもの
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）の文字列番号： bmp・ccitt・gif・jbig2・jpeg・jpeg2000・png・raw・tiff
imagewidth	画像の幅をピクセル単位で表したもの
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 TIFF 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。

表 9.5 PDF_info_image( ) のキーワード一覧

キーワード	説明
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 以外になります）
targetbox	（PDF_open_pdi_page( ) に reference オプションを与えているときのみ）ターゲットページの外接枠のパスハンドル（targetx1 等を参照）
targetx1・targety1・targetx2・targety2・targetx3・targety3・targetx4・targety4	（PDF_begin_template_ext( ) に reference オプションを与えているときのみ）ターゲットページの外接枠矩形の 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。これらの値は、reference オプションの pdiusebox オプションで選択されたページ枠に従って算出されます。
width	与えたオプション群に従った画像の幅をユーザー座標で表したもの
x1・y1・x2・y2・x3・y3・x4・y4	与えたオプション群に従った画像外接枠矩形の 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

9.2 テンプレート

注 この節で解説するテンプレート関数は、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_textblock()*・*PDF_fill_imageblock()*・*PDF_fill_pdfblock()* を使います（187 ページ「11 章 ブロック流し込み関数（PPS）」参照）。

C++ Java ***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()* で与える必要があります。*topdown* オプションを *true* に設定したときは、*height* は 0 以外にする必要があります。

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。

- *PDF_load_image()* の以下のオプションが使えます（表 9.3 参照）：
  iconname・*layer*・*metadata*・*OPI-1.3*・*OPI-2.0*
- *PDF_begin_page_ext()* の以下のオプションが使えます（表 3.7 参照）：
  topdown・*transparencygroup*
- *reference* オプション（表 9.6 参照）。

戻り値 テンプレートハンドル。以後の画像関連の呼び出し（特に *PDF_fit_image()*）で使えます。エラー発生時には −1（PHP では 0）を返します。

詳細 この関数は、テキスト・グラフィック・色状態パラメタをすべてデフォルトにリセットして、グローバルな *topdown* パラメタに従って座標系を定義します。テンプレート定義の最中は、ハイパーテキスト関数と、画像を開く関数は使ってはいけませんが、テキスト・グラフィック・色関数はすべて使えます。

スコープ **文書・ページ**。この関数は**テンプレート**スコープを開始させます。対応する *PDF_end_template()* と必ず対にして呼び出す必要があります。

C++ Java ***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()* で与えた値は上書きされます。

スコープ **テンプレート**。この関数は**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()* と必ず対にして呼び出す必要があります。

表 9.6 PDF_begin_template_ext( ) のオプション一覧

キー	説明
reference	（オプションリスト。PDFlib ソースコードパッケージでは入手不可。PDF 1.4、ただし適切なページ表示には Acrobat 9 以上が必須。PDF/X-1/2/3/4・PDF/A-1 モードでは不可）外部 PDF（「ターゲット」文書）内のページへの参照を指定します。ページまたはテンプレートは、この参照に対する代理として用いられます。ビューアの設定によって、かつターゲット PDF が得られるかどうかによって、内部の代理か、外部のターゲットのいずれかが表示・印刷されます。使えるサブオプションは表 9.7 を参照してください。 ターゲットはローカルで得られる必要があり、かつ、pagelabel または pagenumber サブオプションで指定されたページを含んでいる必要があります。ターゲットは、ユーザーパスワードまたはマスターパスワードを必要とするものではいけません。参照ページの寸法は、reference オプションの pdiusebox サブオプションに従って決定されます。この寸法は、PDF_info_image( )・PDF_info_pdi_page( ) の targetx1・targety1 等のキーワードで取得することができ、代理テンプレートを構築する際、または代理 PDF ページを配置する際には、これに従う必要があります。 PDF/X-5g・PDF/X-5pg モードでは、ターゲットは右記の規格のいずれかに準拠している必要があります：PDF/X-1a:2003・PDF/X-3:2003・PDF/X-4・PDF/X-4p・PDF/X-5g・PDF/X-5pg。かつ、同一の出力インテントに対して用意されている必要があります。

表 9.7 PDF_begin_template_ext( )・PDF_open_pdi_page( ) の reference オプションのサブオプション一覧

キー	説明
filename	（名前文字列。必須）ターゲット PDF を内容として持つファイルの名前。この名前は PDF 内に格納され、ビューアによって使われます。これは、target オプションを指定しないときは、ターゲット PDF をローカルで見つけるためにも使われます（すなわち、その PDF が存在している必要があります）。ディレクトリを一切付けない素のベース名を用いることを推奨します。
hypertext-encoding	（キーワード）pagelabel オプションのエンコーディングを指定します。空文字列は unicode と等価です。デフォルト：グローバルな hypertextencoding パラメタの値
pagelabel	（ハイパーテキスト文字列。pagenumber と同時に指定してはいけません）参照したいページのページラベル
pagenumber	（整数）参照したいページの番号。先頭ページの番号は 1 です。デフォルト：1（ただしこれは pagelabel によって上書きされる可能性があります）
pdiusebox	（キーワード。PDF/X-5g・PDF/X-5pg モードでは media を強制されます）ターゲット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。デフォルト：PDF/X-5g・PDF/X-5pg モードでは media、それ以外では crop。 ***media*** MediaBox を用います（これは必ず存在します） ***crop*** CropBox があるならそれを、ないなら MediaBox を用います ***bleed*** BleedBox があるならそれを、ないなら CropBox を用います ***trim*** TrimBox があるならそれを、ないなら CropBox を用います ***art*** ArtBox があるならそれを、ないなら CropBox を用います
strongref	（論理値。PDF/X-5g・PDF/X-5pg モードでは true を強制されます）true にすると、PDFlib は、ターゲットの ID 項目を用いて、そのターゲットへの強い参照を作成します。ターゲットが別の文書で置き換えられると、この参照は切れます（すなわち、ビューアは代理を用います）。ターゲットを柔軟に取り替える運用をしたいときは、このオプションは false に設定する必要があり、その場合、ローカルのターゲットと、文書が最終的に表示される際に用いられるターゲットとは、等しいページ枠と回転項目を持つ必要があります。デフォルト：true

表 9.7 PDF_begin_template_ext( )・PDF_open_pdi_page( ) の reference オプションのサブオプション一覧

キー	説明
target	（PDF 文書ハンドル）PDF_open_pdi_document( ) で取得したターゲット文書のハンドル。このターゲット PDF は、repair=none オプションを付けて、かつ password オプションを付けずに開いたものである必要があります。文書ハンドルをファイル名に加えて与えることは、2 つの場合に有用でしょう： ▸ 多数の生成文書が同じターゲット PDF を参照するとき、そのターゲットは 1 回だけ開く必要があり、その結果は内部的にキャッシュされることができます。 ▸ ローカルのターゲットのファイル名が、PDF 内に格納したいターゲットファイル名と異なるとき。

C++ Java ***void end_template( )***
Perl PHP ***end_template( )***
C ***void PDF_end_template(PDF *p)***

非推奨。*PDF_end_template_ext( )* を使ってください。

テンプレート定義を完了します。

スコープ **テンプレート**。この関数は**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( )* と必ず対にして呼び出す必要があります。

C++ Java ***int begin_template(double width, double height)***
Perl PHP ***int begin_template(float width, float height)***
C ***int PDF_begin_template(PDF *p, double width, double height)***

非推奨。*PDF_begin_template_ext( )* を使ってください。

9.3 サムネール

C++ Java ***void add_thumbnail(int image)***
Perl PHP ***add_thumbnail(int image)***
C ***void PDF_add_thumbnail(PDF *p, int image)***

画像を、カレントページに対するサムネールとして追加します。

image *PDF_load_image( )* で取得された有効な画像ハンドル。

詳細 この関数は、指定する画像を、カレントページに対するサムネール画像として追加します。サムネール画像は以下の制限を守る必要があります。

- 画像は、106 × 106 ピクセルより大きくてはいけません。
- 画像は、グレースケール・RGB・インデックス RGB のいずれかの色空間を使っている必要があります。
- 複数ページ TIFF 画像は、サムネールとして使えません。なぜならサムネールは、ただ 1 つの PDF 画像オブジェクトから構成される必要があるためです。

この関数は、ページに対してサムネール画像を生成するのではなく、既存の画像をサムネールとして追加する手段を提供しているだけです。サムネール画像自体は、クライアント側で生成する必要があります。クライアントは、サムネールの色・縦横比・内容を、実際のページ内容と一致させる必要があります。

Acrobat と Adobe Reader では、サムネールはページ表示時に生成されますし、またサムネールは生成 PDF 全体のファイルサイズを増大させますので、サムネールを追加せずに、クライアント側でのサムネール生成に任せることを推奨します。

スコープ **ページ**。ページごとに一度しか呼び出してはいけません。サムネールは、すべてのページに付けなければならないわけではありません。

10 PDF 取り込み（PDI）・pCOS 関数

注 この章で説明する関数はすべて、PDF 取り込み ライブラリ（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 文書関数

クックブック 完全なコードサンプルがクックブックの pdf_import/starter_pdfmerge トピックにあります。

この節に関連するパラメタのキー名を表 3.1・表 3.2 に示します（157 ページ「2.1 パラメタ処理」参照）。

表 10.1 PDF_get/set_parameter( ) の PDI 関連キー一覧

キー	説明
pdi[1]	背後のライブラリをビルドする際に PDI がインクルードされている場合は文字列 true を返します。これは、PDFlib GmbH によって頒布されているすべての結合された PDFlib・PDFlib+PDI・PPS のバイナリについて、ライセンスキーにかかわらずあてはまります。そうでないなら false を返します。スコープ：任意・null[2]

1. PDF_get_parameter( ) のみ
2. PDF* 引数を NULL か 0 にして呼び出せます

C++ Java ***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。グローバルな*filenamehandling*オプションまたはパラメタに従って解釈されます。表 2.2 参照）PDF ファイルの名前。

optlist PDF の開く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*（27 ページ「2.5 例外処理」参照）
- 表 10.2 に従った PDF 文書オプション群：*ignorepdfversion*・*infomode*・*inmemory*・*password*・*repair*・*requiredmode*

len （C 言語バインディングのみ）*filename* 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。

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。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。関数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* パラメタまたは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、カレント PDF 文書と非互換な PDF バージョンを用いている。PDF 1.6 以下の各 PDF バージョンに対しては、それ以下のすべてのバージョンが互換です。PDF 1.7・1.7 拡張レベル 3 に対しては、PDF 1.7・PDF 1.7 拡張レベル 3（Acrobat 9）・PDF 1.7 拡張レベル 8（Acrobat X）以下のすべてのバージョンが互換です（なお、Acrobat X の暗号化には未対応です）。ただし、PDF/A モードでは、PDF バージョンヘッダ群は PDF 内で無視されなければならないので、入力 PDF バージョン番号は無視されます。
- 文書が暗号化されているのに、そのパスワードが *password* オプションで与えられていない。
- 文書がカレントの PDF/X または PDF/A 出力準拠レベルに互換でないか、または非互換な出力インテントを使っている。
- 文書がタグ付き PDF であり、*PDF_begin_document( )* の *tagged* オプションが *true* である。

1 番目の原因以外のときは、*infomode* オプションを使って文書を開くこともできます。これは、*PDF_pcos_get_*( )* 関数群を使って、暗号化や PDF/X・PDF/A の状態、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、対応していない形式、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。

PDF/A の場合：取り込む文書は、*infomode=true* にしていないときは、カレントの PDF/A 出力準拠レベルと出力インテントに互換である必要があります。

PDF/X の場合：取り込む文書は、*infomode=true* にしていないときは、カレントの PDF/X 出力準拠レベルに互換である必要があり、かつ、生成する文書と同じ出力インテントを使っている必要があります。

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_pcos_get_*( )* 関数群でだけ使えます。

表 10.2　PDF_open_pdi_document( ) のオプション一覧

キー	説明
infomode	（論理値）true にすると、pCOS インタフェースで情報は取得できるけれども、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。特に、以下の類の文書を、infomode=true では開くことができます： ▸ カレントの PDF/X または PDF/A 準拠レベルと互換でない PDF ▸ カレント文書より高い PDF バージョンを持つ PDF ▸ 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 文書） ▸ PDF_begin_document( ) で tagged オプションを true にしたときのタグ付き PDF 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 MVS）、そのかわりメモリ使用が増えます。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
password	（文字列で最大長 32 文字）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。infomode=true にしているときは、文書情報を取得するにはユーザーパスワード（空でもかまいません）で充分です。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、その暗号化状態を取得するためだけに使えます。

表 10.2 PDF_open_pdi_document( ) のオプション一覧

キー	説明
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小の 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に対しては結果は full モードになります。デフォルト：full

C ***int PDF_open_pdi_callback(PDF *p, void *opaque, size_t filesize, size_t (*readproc)(void *opaque, void *buffer, size_t size), int (*seekproc)(void *opaque, long 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。このポインタは、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へ *size* バイトをコピーするコールバック関数。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表 10.2 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：
infomode・*inmemory*・*password*・*requiredmode*

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。関数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、特殊なインタフェースです。

スコープ **オブジェクト・文書・ページ**。**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 バインディングでのみ得られます。

C++ Java	***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 この関数は、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 ページ関数

C++ Java ***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群を指定したオプションリスト：

- 一般オプション群：*errorpolicy*（表 2.6 参照）・*hypertextencoding*（表 12.1 参照）
- 表 10.3 に従ったページオプション群：
  boxexpand・*clippingarea*・*cloneboxes*・*forcebox*・*iconname*・*layer*・*metadata*・*pdiusebox*・*reference*
- *PDF_begin_page_ext()* の右記オプション（表 3.7 参照）：*transparencygroup*

戻り値 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。関数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。文書を *PDF_open_pdi_document()* で開く際に *infomode* オプションを *true* にしていたときは、そのハンドルは *PDF_fit_pdi_page()* では使えません。

エラー動作は、*errorpolicy* パラメタまたはオプションで変更することができます。

詳細 この関数は、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。

取り込む文書の PDF バージョン番号が、生成する PDF 出力文書の PDF バージョン番号よりも高いときは、この関数は失敗します。PDF 取り込みに関連した問題（未対応の形式、PDF データの破損など）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。

取り込むページの中に参照 XObject があるときは、*PDF_open_pdi_page()* は、代理と参照の両方をターゲットへ複製します。

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。

PDF/A・PDF/X：この呼び出しは、取り込むページを含む文書が生成文書と非互換な出力インテントを用いているときは、失敗する可能性があります。

スコープ **文書・ページ**

表 10.3 PDF_open_pdi_page( ) のオプション一覧

キー	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 / 下 / 右 / 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
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 を用います（これは必ず存在します） ***pdiusebox*** pdiusebox オプションで指定した枠を用います ***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 のいずれかを与えたときは不可。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	（矩形）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 オプションを上書きします。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
iconname	（ハイパーテキスト文字列）取り込むページに名前を付けて、JavaScript で参照できるようにします。たとえば、ページ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）ページを属させたいレイヤー。ただし、このページを配置する前に 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。このページを配置する前に PDF_begin_layer( ) を呼び出してレイヤーを有効にさせると、このページの layer オプションは上書きされます。このページの layer オプションが上書きされないようにするには、このページを配置する前に PDF_end_layer( ) を呼び出してください。
metadata	（オプションリスト。PDF 1.4）取り込んだページに対するメタデータを与えます（237 ページ「14.2 XMP メタデータ」参照）
pdiusebox	（キーワード。cloneboxes を与えたときは不可）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 を用います（これは必ず存在します） ***trim*** TrimBox があればそれを、なければ CropBox を用います

表 10.3 PDF_open_pdi_page( ) のオプション一覧

キー	説明
reference	（オプションリスト。PDF 1.4、ただし適切なページ表示には Acrobat 9 以上が必須）ページを、外部 PDF 文書内のページ（ターゲットページ）への参照を保持する代理として定義します。今回の呼び出しで開かれたページが、参照されるターゲットページに対する代理として用いられます（詳しくは表 9.6 を参照）。代理ページとターゲットページは、互換なページ寸法を持っている必要があります。すなわち、pdiusebox オプションで選択するページ枠が、どちらのページもページ上の同じ位置に配置されるよう、同等である必要があります。

C++ Java ***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 この関数は、*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 **文書・ページ**

C++ Java ***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！）。文書を開く際に、*infomode* オプションを *false* にしておく必要があります。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 ページオプション群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群：
  blind・boxsize・fitmethod・matchbox・orientate・position・rotate・scale・showborder
- 表 9.4 に従ったページ処理のためのオプション群：*adjustpage・gstate*
- 表 10.4 に従った *cloneboxes* オプション。

詳細 この関数は *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。取り込んだページの中に *ExtGState* オブジェクトがあるときは、出力品質を向上させるために、*PDF_begin/end_page_ext( )* に下記のオプションを付けることを推奨します：*transparencygroup={colorspace=DeviceRGB}*。

スコープ **ページ・パターン・テンプレート・グリフ**

表 10.4 PDF_fit_pdi_page( ) の追加オプション

キー	説明
cloneboxes	(論理値。PDF_begin_page_ext( ) で topdown オプションを与えているときは不可。PDF_open_pdi_page( ) の cloneboxes オプションと整合させる必要があります。ページスコープ内のみ)。 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に存在する Rotate・MediaBox・TrimBox・ArtBox・BleedBox・CropBox 項目がすべて、カレント出力ページへ複製されます。 ► ページ内容が、入力ページが二重化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。よって、引数 x・y と右記オプションは無視されます : adjustpage・boxsize・fitmethod・orientate・position・rotate・scale。入力ページの二重化は、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 引数を上書きします。

C++ Java ***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 求める情報を表 10.5 に従って指定したキーワード。

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* (27 ページ「2.5 例外処理」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 9.4 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。それらは、*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*

戻り値 *keyword* で要求した何らかのページメトリックの値。*errorpolicy=return* にすると、この関数はエラー発生時に 0 (PHP では 0) を返します。*errorpolicy=exception* にすると、この関数はエラー発生時に例外を発生させます。

詳細 この関数は、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 *{0 0}* 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。

スコープ ページ・パターン・テンプレート・グリフ・文書・パス

表 10.5 PDF_info_pdi_page( ) のキーワード一覧

キーワード	説明
boundingbox	ページの外接枠のパスハンドル
fitscalex・fitscaley	与えたオプション群に従ってページを枠にはめ込んだ結果の拡縮倍率
height	与えたオプション群と、PDF_open_pdi_page( ) で用いられたオプション群に従ったページの高さを、ユーザー座標系で表したもの
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	元のページの高さをポイント単位で表したもの
pagewidth	元のページの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
width	与えたオプション群と、PDF_open_pdi_page( ) で用いられたオプション群に従ったページの幅を、ユーザー座標系で表したもの
x_1・y_1・x_2・y_2・x_3・y_3・x_4・y_4	与えたオプション群に従ったページ外接枠矩形の i 番目の隅の位置（i=1, 2, 3, 4）をユーザー座標で表したもの。 `cloneboxes=true` にすると、見える枠が用いられます（すなわち、CropBox が存在するならそれが、ないなら MediaBox が）。

10.3 その他の PDI 処理

C++ Java ***int process_pdi(int doc, int page, String optlist)***
Perl PHP ***int process_pdi(int doc, int page, string optlist)***
C ***int PDF_process_pdi(PDF *p, int doc, int page, 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。

page *optlist* でページハンドルを必要としているときは（表 10.6 参照）、*page* は、*PDF_open_pdi_page( )* で取得した有効な PDF ページハンドルにする必要があります（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。*optlist* でページハンドルを必要としていないときは、*page* は -1（PHP では 0）にする必要があります。

optlist PDI 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：*errorpolicy*（27 ページ「2.5 例外処理」参照）
- 表 10.6 に従った PDI 処理オプション群。右記のオプションが使えます：*action*

戻り値 関数が成功したときは値 1、関数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、この関数はエラー時に例外を発生させます。

詳細 PDF/X：出力インテントは、この関数で *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。

PDF/A：出力インテントは、この関数で *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。

スコープ 文書

表 10.6 PDF_process_pdi( ) のオプション

キー	説明
action	（キーワード。必須）PDF 処理の種類を指定します： *copyoutputintent* （出力文書が PDF/X にも PDF/A にも準拠していない場合には何もしません）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。標準出力インテント（ICC プロファイルが埋め込まれていない）をこの方式で複製することはできません。 入力文書と出力文書が PDF/X-4p か PDF/X-5pg 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。このオプション action=copyoutputintent は、入力が PDF/X-4p か PDF/X-5pg 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。

10.4 pCOS 関数

pCOS 関数はすべて、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 パスリファレンスで説明しています。

クックブック **PDFlib+PDI または PPS の中で pCOS を利用するための完全なコードサンプルがクックブックの `pdf_import/starter_pcos` トピックにあります。pCOS クックブックには pCOS プログラミングサンプルが多数あります。**

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 ***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 パスの値を得ます。

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 ***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**文字列**または**論理値**の pCOS パスの値を得ます。

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 この関数は、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。ただし例外として、オブジェクト群 */Info/** （文書情報キー）は、*nocopy=false* か *plainmetadata=true* にしていれば、制限 pCOS モードでも取得することができ、また、*bookmarks[…]/Title* と *annots[…]/contents* は、*nocopy=false* にしていれば、制限 pCOS モードで取得できます。

この関数は、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 形式で返されます。
C バインディング：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関数への呼び出しを 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。

C・C++ 言語バインディング：文字列は、BOM なしの UTF-8 形式で返されます。zSeries・i5/iSeries では、結果は BOM なしの EBCDIC-UTF-8 で返されます。

Java・.NET・Python：結果は Unicode 文字列として与えられます。テキストがそれ以上得られないときは 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。テキストがそれ以上得られないときは null オブジェクトが返されます。

RPG 言語バインディング：結果は EBCDIC-UTF-8 文字列として与えられます。

C++ Java ***const unsigned char *pcos_get_stream(int doc, int *length, 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 *length, const char *optlist, const char *path, ...)***

---

stream または *fstream* または**文字列**型の pCOS パスの内容を得ます。

doc *PDF_open_pdi_document( )* で取得した有効な文書ハンドル。

length （C・C++ 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。

optlist 表 10.7 に従ったストリーム取得オプション群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、制限 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データとテキストのどちらとして取得するか、適切な関数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：返されたデータバッファは、次にこの関数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。

注 この関数を使うと、PDF から、埋め込まれているフォントデータを取得することができます。ユーザーに注意を喚起したいのは、フォントは各フォントベンダーのライセンス契約に従属しているので、各知的所有権者の明示的許諾なしに再利用することはできないということです。お使いのフォントのベンダーに連絡をとり、関連するライセンス契約を話し合ってください。

表 10.7 PDF_pcos_get_stream( ) のオプション一覧

オプション	説明
convert	（キーワード。非サポートのフィルタで圧縮されているストリームに対しては無視されます）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、これは、データが BOM なしの UTF-8 形式へ変換されることを意味します。 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で圧縮されているストリームに対しては無視されます）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に対しては true、そうでないなら false

11 ブロック流し込み関数（PPS）

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関数を提供します。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、ブロック流し込み関数を使う前に、出力ページに ***PDF_fit_pdi_page()*** で配置しておく必要があります。ブロック関数は、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関数は、PDFlib Personalization Server（PPS）が必要です。PDF テンプレートに PDFlib ブロックを作成するには、Adobe Acrobat 用 PDFlib Block Plugin が必要です。

クックブック 完全なコードサンプルがクックブックの `blocks/starter_block` トピックにあります。

11.1 ブロック流し込み関数の矩形オプション

表 11.1 に、***PDF_fill_textblock()*** ・ ***PDF_image_block()*** ・ ***PDF_fill_pdfblock()*** の矩形オプションを挙げます。各ブロック種別（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。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上書きできますが、ただし以下のプロパティだけはオプションで上書きできません：

```
Name, Description, Subtype, Type
defaulttext, defaultimage, defaultpdf, defaultpdfpage
```

表 11.1 PDF_fill_*block( ) 関数の矩形オプション一覧

キー	説明
Rect	（矩形）ブロックの座標をブロック PDF の座標系で指定します。ブロック矩形は `refpoint` ・ `boxsize` オプションで指定できます（ユーザー座標系で）。
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完全に処理されます。 ***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、ブロック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。
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覆い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）ブロック矩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

11.2 テキスト行・テキストフローブロック

C++ Java ***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* 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。

optlist テキストブロック流し込みオプション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*（27 ページ「2.5 例外処理」参照）
- 表 11.1 に従ったブロック流し込み関数のための矩形オプション群：
  Rect・*Status*・*backgroundcolor*・*bordercolor*・*linewidth*
- はめ込みオプション群（111 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（80 ページ「5.2 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（85 ページ「5.3 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 5.14 参照）
- 表 11.2 に従ったテキストブロックオプション群：*textflow*・*textflowhandle*

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）、またはブロックがもっと新しい PDFlib 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ときは、有効な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。

PDF 文書が破損していることがわかったときは、この関数は *errorpolicy* パラメタまたは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、この関数は、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、デフォルトでは下記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。

embedding オプションについては特に注意を払う必要があります：フォントがブロックのプロパティに基づいて暗黙的に読み込まれた場合には、それは自動的に埋め込まれるわけではありません。フォントを埋め込ませたいときは、*embedding* オプションを指定する必要があります。

テキストフローブロックの連結：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、任意の回数繰り返すことができます。テキストフローハンドルを最後に *PDF_delete_textflow( )* で削除するのはユーザー側の役割です。

スコープ　ページ・テンプレート

表 11.2 PDF_fill_textblock( ) の追加オプション一覧

キー	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`
textflow-handle	（テキストフローハンドル。PDF_fill_textblock( ) で `textflow=true` の場合のみ）このオプションを利用すると、テキストフローブロック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。この関数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。このオプションは `fitmethod` のデフォルトを `clip` に変更します。

11.3 イメージブロック

C++ Java ***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*（27 ページ「2.5 例外処理」参照）
- 表 11.1 に従ったブロック流し込み関数のための矩形オプション群：
  Rect・*Status*・*backgroundcolor*・*bordercolor*・*linewidth*
- 流し込みオプション群（111 ページ「6.1 オブジェクトのはめ込み」参照）
- 表 9.4 に従った画像処理のためのオプション群。

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、またはブロックがもっと新しい PDFlib 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、この関数は、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、この関数は *errorpolicy* パラメタまたはオプションに従って、例外を発生させるか、-1 を返します。

スコープ ページ・テンプレート

11.4 PDF ブロック

C++ Java ***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*（27 ページ「2.5 例外処理」参照）
- 表 11.1 に従ったブロック流し込み関数のための矩形オプション群：
  Rect・Status・backgroundcolor・bordercolor・linewidth
- 流し込みオプション群（111 ページ「6.1 オブジェクトのはめ込み」参照）
- 表 9.4 に従ったページ処理のためのオプション群。

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、またはブロックがもっと新しい PDFlib 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、この関数は、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、この関数は *errorpolicy* パラメタまたはオプションに従って、例外を発生させるか、-1 を返します。

スコープ ページ・テンプレート

12 インタラクティブ機能

12.1 インタラクティブ要素のパラメタ

インタラクティブ要素に関連するパラメタのキー名を表 12.1 に示します（19 ページ「2.2 パラメタ・オプション処理」参照）。これらのパラメタは、Unicode 対応の言語バインディングでは得られません。

表 12.1 PDF_get/set_parameter( ) の文字列関連キー一覧

キー	説明
hypertextencoding	（オプションとしても得られます）ハイパーテキスト型のオプションとパラメタのエンコーディング。空文字列なら、unicode と同義になります。デフォルト：auto。スコープ：任意
hypertextformat	ハイパーテキスト型のオプションとパラメタの形式。とりうる値は bytes・utf8・utf16・utf16le・utf16be・auto。デフォルト：auto。スコープ：任意
usehypertextencoding	true なら、hypertextencoding パラメタで指定されているエンコーディングは、名前文字列に対しても使われます。false なら、UTF-8 BOM がない名前文字列のエンコーディングは host です。デフォルト：false。スコープ：任意
usercoordinates	false なら、ハイパーテキスト矩形の座標はデフォルト座標系で与えられると見なされます。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：false。スコープ：任意

12.2 アクション

C++ Java ***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 12.2 に従ったアクション種別。

表 12.2 アクション種別一覧

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 アニメーション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
Hide	（PDF/A では不可）注釈またはフォームフィールドを、隠すか、または表示させます：hide・namelist
ImportData	（PDF/A では不可）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。

表 12.2 アクション種別一覧

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）
JavaScript	（PDF/A では不可）JavaScript コードによるスクリプトを実行します：script・scriptname
Launch	（PDF/A では不可）アプリケーションまたは文書を起動します：defaultdir・filename・newwindow・operation・parameters
Movie	（PDF/A では不可）外部のサウンドファイルかムービーファイルを、フローティングウィンドウ内で、またはムービー注釈の矩形内で再生します：operation・target
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
ResetForm	（PDF/A では不可）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。
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（ただし、ベーシック認証を要する送信は Acrobat では動作しません）：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：

- 一般オプション群：*errorpolicy*（表 2.6 参照）・*hypertextencoding*（表 12.1 参照）
- 表 12.3 に従ったアクションプロパティオプション群。

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 この関数は、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()* を呼び出して生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/X：PDF/X ではアクションは禁じられています。
PDF/A：PDF/A ではいくつかのアクションは禁じられています（表 12.2 参照）。

スコープ **ページ・文書**。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 12.3 PDF_create_action( ) のアクションプロパティオプション一覧

オプション	説明
3Dview	（キーワードまたは 3D ビューハンドル。GoTo3DView。必須）対象とする 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

表 12.3 PDF_create_action( ) のアクションプロパティオプション一覧

オプション	説明
defaultdir	（文字列。Launch）起動するアプリケーションのためのデフォルトディレクトリを設定します。これは、Windows 版の Acrobat でだけ使えます。デフォルト：なし
destination	（オプションリスト。GoTo・GoToE・GoToR。destname を与えていないときは必須）飛ばしたい移動先を表 12.5 に従って指定したオプションリスト。
destname	（ハイパーテキスト文字列） GoTo（destination を与えていないときは必須）：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。 GoToR・GoToE（destination を与えていないときは必須）：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
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、フィールドの矩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。Acrobat 8 では、完全修飾ファイル名（パスを含む）を Launch アクションで指定しても動きませんので、ディレクトリ名は defaultdir オプションで与え、filename オプションには単純なファイル名のみ（ディレクトリ要素なしの）を与えることを推奨します。 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。
hide	（論理値。Hide）注釈を隠すか（true）、それとも表示するか（false）を指定します。デフォルト：true

表 12.3 PDF_create_action( ) のアクションプロパティオプション一覧

オプション	説明
ismap	（論理値。URI）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
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から成る対のリスト。使えるキーワード： ***on*** レイヤーを表示します ***off*** レイヤーを隠します ***toggle*** レイヤーの状態を反転させます。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 モードでは、よく知られている名前 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完全なコードサンプルがクックブックの 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または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ィ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ィ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	このオプションは type=Launch と type=Movie で使い方が異なります： （キーワード。Launch）filename オプションで指定している文書に適用したい操作を指定するキーワード。これは、Windows 版の Acrobat でだけ使えます。filename オプションが文書でなくアプリケーションを指しているときは、このオプションは無視されて、アプリケーションが起動されます。とりうるキーワード（デフォルト：open）： ***open*** 文書を開きます ***print*** 文書を印刷します （キーワード。Movie）ムービーまたはサウンドに適用したい操作を指定するキーワード。とりうるキーワード（デフォルト：play）： ***play*** ムービーの再生を開始します。その際、そのムービー注釈の playmode オプションで指定されているモードを使用します。その時点でそのムービーが一時停止されているときは、再生位置を先頭へ戻してから再生されます。 ***stop*** ムービーの再生を停止します。 ***pause*** ムービーの再生を一時停止します。 ***resume*** 一時停止されているムービーを再開します。
parameters	（文字列。Launch）filename オプションで指定しているアプリケーションに渡したい引数文字列。これは、Windows 版の Acrobat でだけ使えます。複数の引数はスペースキャラクタで区切ることができますが、個々の引数には一切スペースキャラクタを入れてはいけません。filename が文書を指しているときは、このオプションは指定するべきではありません。デフォルト：なし

表 12.3 PDF_create_action( ) のアクションプロパティオプション一覧

オプション	説明
preserve-radio	（論理値。SetOCGState）true にすると、レイヤー間のラジオボタン状態関係を保持します。デフォルト：true
script	（ハイパーテキスト文字列。JavaScript。必須）実行させたい JavaScript コードを入れた文字列。
scriptname	（ハイパーテキスト文字列。JavaScript）指定すると、script 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、最後のスクリプトだけが使われ、他は無視されます。文書レベルの JavaScript は、Acrobat で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。PDF 1.4）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・Movie。必須）PDF_create_annotation( ) の name オプションで指定するのと同じ、対象としたい 3D 注釈またはムービー注釈の名前。
targetpath	（オプションリスト。GoToE。filename を指定していなければ必須）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 12.4 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入れ子の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 3.7 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。textx/texty・currentx/currenty・imagewidth/imageheight パラメタは、リンクの矩形の寸法を算出するための位置情報を取得するのに有用かもしれません。

表 12.4 PDF_create_action( ) の targetpath オプションのサブオプション一覧

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を指定します。
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は必須）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を指定します。このオプションは、pagenumber を指定すると無視されます。
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は必須。それ以外の場合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を指定します。

表 12.4 PDF_create_action( ) の targetpath オプションのサブオプション一覧

オプション	説明
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）の関係を指定します。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 12.4 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。

12.3 名前付き移動先

C++ Java ***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、最後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* 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。

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：

- 一般オプション群：*errorpolicy*（表 2.6 参照）・*hypertextencoding*・*hypertextformat*（表 12.1 参照）
- 表 12.5 に従った移動先制御オプション群：
  bottom・*group*・*left*・*page*・*right*・*top*・*type*・*zoom*

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、これらの関数で *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 文書・ページ

表 12.5 PDF_add_nameddest( ) の移動先オプション一覧。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 のみ）ウィンドウの下端に合わせたい、ページの y 座標。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なら必須。それ以外なら禁止）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 のみ）ウィンドウの左端に合わせたい、ページの x 座標。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
right	（float。type=fitrect のみ）ウィンドウの右端に合わせたい、ページの x 座標。デフォルト：1000
top	（float。type=fixed・fitwidth・fitrect・fitvisiblewidth のみ）ウィンドウの上端に合わせたい、ページの y 座標。デフォルト：1000

表 12.5 PDF_add_nameddest( ) の移動先オプション一覧。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
type	（キーワード）対象ページ上でのウィンドウの位置を指定します。とりうるキーワード（デフォルト：fitwindow）： ***fitheight*** ページの高さをウィンドウに収めて、x 座標 left をウィンドウの左端に合わせます。 ***fitrect*** left・bottom・right・top で指定している矩形をウィンドウに収めます。 ***fitvisible*** ページの描画領域（ArtBox）をウィンドウに収めます。 ***fitvisibleheight*** ページの描画領域をウィンドウに収めて、x 座標 left をウィンドウの左端に合わせます。 ***fitvisiblewidth*** ページの描画領域をウィンドウに収めて、y 座標 top をウィンドウの上端に合わせます。 ***fitwidth*** ページの幅をウィンドウに収めて、y 座標 top をウィンドウの上端に合わせます。 ***fitwindow*** ページ全体をウィンドウに収めます。 ***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 のみ）ページ内容の表示に使いたい倍率（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。

12.4 注釈

C++ Java ***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 注釈の矩形の左下隅と右上隅の *x・y* 座標を、デフォルト座標で（*usercoordinates* パラメタまたはオプションを *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。Acrobat は、注釈の左上隅を、指定矩形の左上隅に合わせます。

注釈の座標は、*PDF_rect( )* 関数の引数とは違うことに注意してください。*PDF_create_annotation( )* では引数が 2 個の隅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。

usematchbox オプションを指定しているときは、引数 *llx/lly/urx/ury* は無視されます。

type 表 12.6 に従った注釈種別。マークアップ注釈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 12.6 注釈種別一覧

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）
3D	（PDF 1.6）アニメーション 3D モデル：3Dactivate・3Ddata・3Dinteractive・3Dshared・3Dinitialview
Circle[1]	楕円形：cloudy・createrichtext・inreplyto・interiorcolor・replyto
File-Attachment[1]	（PDF/A では不可）ファイル添付：calloutline・createrichtext・filename・iconname・inreplyto・mimetype・replyto
FreeText[1]	テキストボックス：alignment・calloutline・cloudy・createrichtext・endingstyles・fillcolor・font・fontsize・inreplyto・orientate・replyto
Highlight[1]	ハイライト：createrichtext・inreplyto・polylinelist・replyto
Ink[1]	鉛筆：createrichtext・inreplyto・polylinelist・replyto
Line[1]	線：captionoffset・captionposition・createrichtext・endingstyles・interiorcolor・inreplyto・leaderlength・leaderoffset・line・showcaption・replyto
Link	リンク：destination・destname・highlight
Movie	ムービーまたはサウンド注釈：filename・movieposter・playmode・showcontrols・soundvolume・windowposition・windowscale
Polygon[1]	（PDF 1.5）多角形。頂点群を直線で結んだものです：cloudy・createrichtext・inreplyto・polylinelist・replyto
PolyLine[1]	（PDF 1.5）折れ線。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	ポップアップ：open・parentname

表 12.6 注釈種別一覧

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）
Square[1]	長方形：cloudy・createrichtext・inreplyto・interiorcolor・replyto
Squiggly[1]	（PDF 1.4）波線。波型の下線注釈です：createrichtext・inreplyto・polylinelist・replyto
Stamp[1]	スタンプ：createrichtext・iconname・inreplyto・orientate・replyto
StrikeOut[1]	取り消し線：createrichtext・inreplyto・polylinelist・replyto
Text[1]	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[1]	下線：createrichtext・inreplyto・polylinelist・replyto

1. マークアップ注釈。これは createrichtext オプションで意味を持ちます。

optlist 注釈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：*hypertextencoding*（表 12.1 参照）
- 表 12.7 に従った以下の共通オプション群がすべての注釈種別で使えます：
  action・annotcolor・borderstyle・cloudy・contents・createdate・custom・dasharray・display・layer・linewidth・locked・lockedcontents・name・opacity・popup・readonly・rotate・subject・template・title・usematchbox・usercoordinates・zoom
- 表 12.7 に従った以下の種別特有オプション群が、表 12.6 に従った何らかの注釈種別で使えます：
  3Dactivate・3Dbox・3Ddata・3Dicon・3Dinteractive・3Dshared・3Dinitialview・alignment・calloutline・captionoffset・captionposition・createrichtext・destname・endingstyles・filename・fillcolor・font・fontsize・highlight・iconname・interiorcolor・inreplyto・leaderlength・leaderoffset・line・mimetype・movieposter・open・orientate・parentname・playmode・polylinelist・replyto・showcaption・showcontrols・soundvolume・windowposition・windowscale

詳細 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。
PDF/A：いくつかの注釈種別とオプションが制約されます。表 12.6・表 12.7 を参照。
タグ付き PDF：カレントのアクティブなアイテムがあるときは、注釈はカレントアイテムの子として挿入されます。

スコープ ページ

表 12.7 PDF_create_annotation( ) のオプション一覧

オプション	説明
3Dactivate	（オプションリスト。type=3D のみ）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。表 12.8 に挙げるオプションが使えます：
3Ddata	（オプションリスト。type=3D のみ。必須）PDF_load_3ddata( ) で作成した 3D ハンドル。
3Dinteractive	（論理値。type=3D のみ）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

表 12.7 PDF_create_annotation( ) のオプション一覧

オプション	説明
3Dshared	（論理値。type=3D のみ）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。同じデータを間接的に参照する複数の 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
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指定します。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、または PDF_load_3ddata( ) で作成した 3D ビューハンドル。デフォルト：default
action	（アクションリスト）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： ***activate*** （type=Link のみ）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）その注釈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。 ***open*** （PDF 1.5）その注釈を含むページが開かれた時に実行させたいアクション。 ***invisible*** （PDF 1.5）その注釈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。 ***visible*** （PDF 1.5）その注釈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。type=FreeText のみ）注釈の中のテキストの整列：left・center・right。デフォルト：left
annotcolor	（色）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ィンドウのタイトルバーと、リンク注釈の枠の色。使える色空間：none（type=Square・Circle では不可）・gray・rgb・（PDF 1.6 で）cmyk。 PDF/A モードではこのオプションは、RGB の出力インテントを指定しているときのみ使うことができ、かつ gray または rgb カラーを使う必要があります。 デフォルト：type=Square・Circle なら white、それ以外なら none
borderstyle	（キーワード）注釈の枠か、または種別 Polygon・PolyLine・Line・Square・Circle・Ink の注釈の線のスタイル：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	（float 4 個か 6 個のリスト。PDF 1.6。type=FreeText のみ）テキストボックス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 個の数値 {x1 y1 x2 y2 x3 y3} は、線の開始・折れ点・終了座標を表します。4 個の数値 {x1 y1 x2 y2} は、線の開始・終了座標を表します。座標は、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）解釈されます。 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。
captionoffset	（float 2 個。type=Line のみ。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。type=Line のみ。PDF 1.7）注釈のキャプション位置。このオプションは、showcaption=false にすると無視されます。使えるキーワード（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。
cloudy	（float。type=Circle・FreeText・Polygon・Square のみ。PDF 1.5）多角形の変形に用いられる「雲型」効果の強度を指定します。とりうる値は 0（効果なし）・1・2。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

表 12.7 PDF_create_annotation( ) のオプション一覧

オプション	説明
contents	(type=FreeText の場合は文字列、それ以外の場合は最大長 65535 バイトのハイパーテキスト文字列) 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を使うと、段落替えを強制することができます。 PDF/A-1a モードでは、このオプションは必須であり、かつ、空でない文字列を入れる必要があります。
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 : false
createrich-text	(オプションリスト。マークアップ註釈のみ。contents オプションを空にする必要があります。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生成するのに有用でしょう。使えるサブオプション : ***textflow*** (テキストフローハンドル) 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 注意 : font と alignment を設定しても、Acrobat では期待した通りに動作しません。
custom	(オプションリストのリスト。高度なユーザーのみ) このオプションを使うと、注釈辞書の中に任意の数のプライベート項目を挿入することができます。これは、デジタル出力機のための処理命令を挿入するなど、専門的な応用において有用でしょう。このオプションを使うには、PDF ファイル形式と対象応用分野に関する知識が必要です。不適切な値を与えると、破損した PDF 出力が生成されることがあります。使えるサブオプション : ***key*** (文字列。必須) 辞書キーの名前 (キャラクタ / を除いたもの)。あらゆる非標準の PDF キーのほか、右記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。それぞれオプションはこの場合無視されます。 ***type*** (キーワード。必須) その値の型。boolean・name・string のいずれかにする必要があります。 ***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) PDF 出力内に現れるとおりの値。PDFlib は、文字列と名前に必要な修飾をすべて自動的に行います。
dasharray	(float のリスト。borderstyle=dashed のみ) 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 (PDF_setdash( ) 参照)。デフォルト : 3 3
destination	(オプションリスト。type=Link のみ。activate アクションを指定していると無視されます) 飛ばしたい移動先を表 12.5 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 のみ。destination オプションを指定していると無視されます) 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。
display	(キーワード) 画面と紙の上での表示・非表示。visible・hidden・noview・noprint。デフォルト : visible

表 12.7 PDF_create_annotation( ) のオプション一覧

オプション	説明
endingstyles	（キーワードのリスト。type=FreeText・Line・PolyLine のみ）線端スタイルを指定したキーワード 2 個のリスト。type=FreeText の場合は、2 番目のキーワードは無視されます（デフォルト：{none none}）： none・square・circle・diamond・openarrow・closedarrow これに加えて PDF 1.5 では：butt・ropenarrow・rclosedarrow これに加えて PDF 1.6 では：slash
filename	（文字列。type=FileAttachment・Movie のみ。必須）このファイル名は、グローバルな filenamehandling オプションまたはパラメタに従って解釈されます。表 2.2 参照。 type=FileAttachment の場合：注釈に関連づけたいファイルの名前。 type=Movie の場合：右記のいずれかの形式のメディアファイルの名前：AVI・QuickTime ムービー、WAV・AIFF サウンド
fillcolor	（色。type=FreeText のみ）テキストの塗り色。使える色空間：gray・rgb・cmyk。PDF/A モードではこのオプションは、RGB か CMYK の出力インテントを指定しているときのみ使うことができ、かつそれぞれ rgb か cmyk の色空間を使う必要があります。デフォルト：{gray 0}（＝黒）
font	（フォントハンドル。type=FreeText のみ。必須）注釈で使いたいフォントを指定します。
fontsize	（文字サイズ。type=FreeText のみ。必須）文字サイズを、usercoordinates オプションまたはパラメタに応じて、デフォルトかユーザー座標かで指定します。値 0 またはキーワード auto を指定すると、Acrobat が文字サイズを矩形に合わせて調節します。
highlight	（キーワード。type=Link のみ）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	（文字列。type=Text・Stamp・FileAttachment のみ）注釈の表示に使いたいアイコンの名前（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）： comment・key・note・help・newparagraph・paragraph・insert type=Stamp の場合（デフォルト：draft）： approved・experimental・notapproved・asis・expired・notforpublicrelease・confidential・final・sold・departmental・forcomment・topsecret・draft・forpublicrelease type=FileAttachment の場合（デフォルト：pushpin）： graph・pushpin・paperclip・tag template オプションを使うとカスタムアイコンを作成することができます。
interiorcolor	（色。type=Line・PolyLine・Square・Circle のみ）それぞれ、注釈の線端・矩形・楕円の色。使える色空間：none・gray・rgb・（PDF 1.6 で）cmyk。PDF/A モードではこのオプションは、RGB の出力インテントを指定しているときのみ使うことができ、かつ gray または rgb カラーを使う必要があります。デフォルト：none
inreplyto	（ハイパーテキスト文字列。PDF 1.5。マークアップ注釈のみ。replyto オプションを与えたときは必須）この注釈の返信先である注釈の名前（name オプション参照）。両方の注釈が文書の同じ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 オプションで指定する必要があります。
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。

表 12.7 PDF_create_annotation( ) のオプション一覧

オプション	説明
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。type=Line のみ。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。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、線自身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、線から、補助線からちょうど 180° を成すよう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。type=Line のみ。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空隙の幅を、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。デフォルト：0
line	（線。type=Line のみ。必須）線の開始・終了座標を、デフォルト座標で（usercoordinates パラメタを false にしたとき）、またはユーザー座標で（true にしたとき）指定した 4 個の数値 x1・y1・x2・y2 のリスト。
linewidth	（整数）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（＝ポイント）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 Acrobat で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
locked-contents	（論理値。PDF 1.7）true にすると、注釈の内容を Acrobat で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、注釈のプロパティ（位置・寸法等）を変更することは可能です。デフォルト：false
mimetype	（文字列。type=FileAttachment のみ）ファイルの MIME タイプ。Acrobat は、注釈がアクティブにされた時にこれを使って適切なアプリケーションを起動します。
movieposter	（キーワード。type=Movie のみ）ページ上でムービーを代表するポスター画像を指定するキーワード。使えるキーワード：auto（ポスター画像がムービーファイルから抽出されます）・none（ポスターが表示されません）。デフォルト：none
name	（ハイパーテキスト文字列）注釈を一意に同定する名前。この名前は、いくつかのアクションで必要であり、ページ上で一意でなければなりません。デフォルト：なし
opacity	（float またはパーセント値。PDF 1.4）注釈を描く際に使わせたい、固定した不透明度の値（0 ～ 1 または 0% ～ 100%）。デフォルト：1
open	（論理値。type=Text・Popup のみ）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 のみ）注釈をその矩形の中で向けたい向きを指定します。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 のみ）注釈の親注釈の名前。
playmode	（キーワード。type=Movie のみ）ムービーまたはサウンドを再生するモード。使えるキーワード：once（1 回再生して停止します）・open（再生してムービーコントローラバーを開いたままにします）・repeat（最初から最後まで繰り返し、停止されるまで再生します）・palindrome（順方向再生と逆再生を連続して、停止されるまで行います）。デフォルト：once

表 12.7 PDF_create_annotation( ) のオプション一覧

オプション	説明
polylinelist	（折れ線か四辺形のリスト。type=Polygon・PolyLine・Ink・Highlight・Underline・Squiggly・Strikeout のみ）座標は、デフォルト座標系で（usercoordinates オプションを false にしたとき）、またはユーザー座標系で（true にしたとき）解釈されます。デフォルト：注釈矩形の頂点群を結んだ折れ線。 ***type=Polygon・PolyLine・Ink*** n 本の線分（最小：n=2）による折れ線 1 個のリスト。 **その他**　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（最小：n=1）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。
popup	（文字列）この注釈に関連づけたテキストの入力や編集のためのポップアップ注釈の名前。デフォルト：なし
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
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）を指定します。使えるキーワード（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。
rotate	（論理値。テキスト注釈に対しては PDF/A モードでは 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：テキスト注釈に対しては PDF/A モードでは false、それ以外では true
showcaption	（論理値。type=Line のみ。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
showcontrols	（論理値。type=Movie のみ）true にすると、ムービーまたはサウンドの再生中にコントローラーが表示されます。デフォルト：false
soundvolume	（float。type=Movie のみ）ムービーを再生する最初の音量を、−1.0 ～ 1.0 の範囲で指定します。 値を大きくするほど高い音量を示します。負の値にすると消音されます。デフォルト：1.0
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を表現したテキスト。デフォルト：なし

表 12.7 PDF_create_annotation( ) のオプション一覧

オプション	説明
template	（オプションリスト）注釈の各状態の体裁。これはカスタムスタンプ等で有用でしょう。使えるサブオプション： ***normal/rollover/down*** （テンプレートハンドルまたはキーワード）注釈の通常・マウスロールオーバー・マウスボタン押下時にそれぞれ用いられるテンプレート。キーワード viewer は、Acrobat に対し、ページを表示する際に体裁を作成するよう指示します。normal のデフォルト：viewer。rollover・down のデフォルト：normal の値 ***fitmethod***（キーワード）テンプレートを注釈矩形内にはめ込む方式。fitmethod を entire 以外にすると、注釈矩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、矩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、対応する矩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、矩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。ムービー注釈では推奨）注釈のポップアップウィンドウが開いていてアクティブな時にタイトルバーに表示させたいテキストラベル。この文字列は Acrobat の「作成者」フィールドに対応します。title の最大長は、シングルバイトなら 255 キャラクタ、Unicode なら 126 キャラクタです。ただし title の実用上の上限としては、32 キャラクタを推奨します。デフォルト：なし
usematchbox	（文字列のリスト）引数 llx/lly/urx/ury を無視して、かわりに範囲枠を用います。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、使いたい矩形の番号を指定する整数か、またはキーワード all で、選んでいる範囲枠を参照するすべての矩形を指定します。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。 範囲枠自体か、または指定した矩形が、カレントページに存在しないときは、関数は注釈を作成せずに、警告を出さずに返ります。
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：グローバルな usercoordinates パラメタの値
window-position	（float 2 個かキーワード 2 個のリスト。type=Movie のみ）フローティングムービーウィンドウについて、画面上におけるそのウィンドウの相対位置。2 個の値で、画面上におけるフローティングウィンドウの位置を示します。{0 0} が画面の左下隅を、{100 100} が右上隅を表します。キーワード left・center・right（画面横方向で）、または bottom・center・top（画面縦方向で）を、値 0・50・100 と同じ意味で使うこともできます。デフォルト：{center center}
windowscale	（float またはキーワード。type=Movie のみ）フローティングウィンドウ内でムービーを再生する表示倍率。このオプションを指定しないと、ムービーは注釈矩形内で再生されます。このオプションの値は、ムービーの表示倍率か、または以下のキーワードです（デフォルト：オプションなし、すなわちムービーは注釈矩形内で再生されます）。 ***fullscreen*** ムービーは、得られる画面領域をいっぱいに使って再生されます。

表 12.7 PDF_create_annotation( ) のオプション一覧

オプション	説明
zoom	（論理値。テキスト注釈に対しては PDF/A モードでは 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：テキスト注釈に対しては PDF/A モードでは false、それ以外では true

表 12.8 PDF_create_annotation( ) の 3Dactivate オプションのサブオプション一覧

オプション	説明
enable	（キーワード）注釈をいつアクティブに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アクティブにします。 ***visible*** ページが見えた時にアクティブにします。 ***click*** 注釈は、スクリプトかユーザーアクションで明示的にアクティブにする必要があります。
enablestate	（キーワード）初期アニメーション状態（デフォルト：play）： ***pause*** 3D モデルが実体化されますが、スクリプトアニメーションは無効になります。 ***play*** 3D モデルが実体化されます。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）注釈をいつ非アクティブに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非アクティブにします。 ***invisible*** ページが見えなくなった時に非アクティブにします。 ***click*** 注釈は、スクリプトかユーザーアクションで明示的に非アクティブにする必要があります。
disablestate	（キーワード）注釈を非アクティブにした際の状態（デフォルト：reset）： ***pause*** 3D モデルはレンダリングできますが、アニメーションは無効になります。 ***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ります。 ***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）

12.5 フォームフィールド

クックブック 完全なコードサンプルがクックブックの webserver/starter_webform トピックにあります。

C++ Java ***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)***

フォームフィールドを、カレントページ上に、さまざまなオプションに従って作成します。

llx・lly・urx・ury フィールドの矩形の左下隅と右上隅の *x・y* 座標を、デフォルト座標で（*usercoordinates* パラメタまたはオプションを *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。

フォームフィールドの座標は、*PDF_rect()* 関数の引数とは違うことに注意してください。*PDF_create_field()* では引数が 2 個の隅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()* で作成しておいたグループ（複数可）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「.」で区切る必要があります。フィールド名は、ページ上で一意にする必要があり、また、ピリオドキャラクタ「.」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* 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。

type フィールドの種別を表 12.9 に従って指定します。

表 12.9 フォームフィールド種別一覧

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）
pushbutton		buttonlayout・caption・captiondown・captionrollover・charspacing・fitmethod・icon・icondown・iconrollover・position・submitname
checkbox		currentvalue・itemname
radiobutton		buttonstyle・currentvalue・itemname・toggle・unisonselect ラジオボタンは、必ずグループに属させる必要があるので、名前の頭にグループ名をつける必要があります。その他のすべてのフィールド種別では、グループ所属は必須ではありません。
listbox		charspacing・currentvalue・itemnamelist・itemtextlist・multiselect・sorted・topindex
combobox		commitonselect・charspacing・currentvalue・editable・itemnamelist・itemtextlist・sorted・spellcheck

表 12.9 フォームフィールド種別一覧

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
		テキストフィールドはバーコードにも使われます：barcode
signature		charspacing・lockmode

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：

- 一般オプション群：*errorpolicy*（表 2.6 参照）・*hypertextencoding*・*hypertextformat*（表 12.1 参照）
- フィールドのプロパティ群を表 12.10 に従って指定したオプションリスト。文字列オプションは、表に示すとおりにハイパーテキスト文字列か内容文字列として解釈されます。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（これに加えて種別特有のオプション群については表 12.9 を参照）：
  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*

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、フィールドを生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* オプションで上書き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。しかし、これは PDF リ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（しかし標準化されていない）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。

PDF/A モードでは、この関数は呼び出してはいけません。

すべての PDF/X モードにおいて、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。

タグ付き PDF の場合：カレントのアクティブなアイテムがあるときは、フィールドはカレントアイテムの子として挿入されます。

スコープ ページ

C++ Java ***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の入れ子にすることが可能です。グループ名どうしはピリオドキャラクタ「.」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「.」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* 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。

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、1 つのフィールドの内容を他のフィールド（複数可）へミラーするのに有用です。*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の名前を付けると、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プロパティオプションはすべて、そのグループに属する全フィールドに継承されます。

スコープ **ページ・文書**

表 12.10 PDF_create_field( )・PDF_create_fieldgroup( ) によるフィールドプロパティオプション一覧

オプション	説明
action	（アクションリスト）以下のイベントのいずれか（複数可）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）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 のみ。readonly を暗示。PDF 1.7ext3）表 12.11 のオプション群に従ってバーコードフィールドを生成します。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=...}. このバーコードは、Acrobat 9 以上で表示されますが、Adobe Reader ではどのバージョンでも表示されません。Acrobat 9 と X は、1 つのページの最初のフィールドがバーコードフィールドだとクラッシュします。この問題を回避するには、バーコードフィールドを追加する前に別のフィールドを作る必要があります。この最初のフィールドは、このクラッシュを避けるには、幅・高さゼロのダミーのテキストフィールドであれば充分です。
borderstyle	（キーワード）フィールドの枠のスタイル。solid・beveled・dashed・inset・underline のいずれか。デフォルト：solid
button-layout	（キーワード。type=pushbutton のみ）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

表 12.10 PDF_create_field( )・PDF_create_fieldgroup( ) によるフィールドプロパティオプション一覧

オプション	説明
buttonstyle	（キーワード。type=radiobutton・checkbox のみ）フィールドで使いたい記号を指定します：check・cross・diamond・circle・star・square。デフォルト：check
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を指定します。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 のみ。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。font オプションで指定したフォントで表示されます。テキストもアイコンも表示させたくないときは、空文字列（すなわち caption { }）を使います。デフォルト：なし
caption-down	（内容文字列。type=pushbutton のみ）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。font オプションで指定したフォントで表示されます。デフォルト：なし
caption-rollover	（内容文字列、type=pushbutton のみ）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。font オプションで指定したフォントで表示されます。デフォルト：なし
charspacing	（float。type=radiobutton・checkbox 以外のみ）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。このオプションは、Acrobat 7 では無視されます。デフォルト：0
comb	（論理値。type=textfield のみ。PDF 1.5）true にすると、multiline・fileselect・password オプションを false にしているときは、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 のみ。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）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。文字列 Off にすると、ボタンは非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は設定する必要があります。デフォルト：Off ***textfield・combobox*** （内容文字列）フィールドの内容。font オプションで指定したフォントで表示されます。デフォルト：空 ***listbox*** （整数のリスト）itemtextlist の中で選択させておきたい項目群の番号。デフォルト：なし
dasharray	（float のリスト。borderstyle=dashed のみ）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（PDF_setdash( ) 参照）。デフォルト：3 3
defaultvalue	reset アクションの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）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 のみ）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 12.10 PDF_create_field( )・PDF_create_fieldgroup( ) によるフィールドプロパティオプション一覧

オプション	説明
fieldtype	（キーワード。PDF_create_fieldgroup( ) のみ）グループに入れたいフィールドの種別：mixed・pushbutton・checkbox・radiobutton・listbox・combobox・textfield・signature。 fieldtype=mixed にしないと、グループには、指定種別のフィールドしか入れられなくなります。特定の fieldtype をグループに対して指定しておくと、たとえその中のフィールドが別々のページに置いてあったとしても、すべてのフィールドにそのカレント値が同時に表示されます。 fieldtype=radiobutton にしたときは、unisonselect オプション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( ) のオプションで指定する必要があり、PDF_create_field( ) のオプションで指定してはいけません。デフォルト：mixed
fileselect	（論理値。type=textfield のみ）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）テキストの塗り色。使える色空間：gray・rgb・cmyk。デフォルト：{gray 0}（＝黒）
fitmethod	（キーワード。type=pushbutton のみ）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。つねに ZapfDingbats を使う type=radiobutton・checkbox 以外では必須。type=pushbutton の場合は、caption・captionrollover・captiondown のうちの 1 個ないし複数のオプションを指定したときのみ必須）。 フィールドで使いたいフォントを指定します。右記の種類のフォントはテキストフォームフィールドでは許されません：レガシ CMap による日中韓フォント、TrueType または OpenType サブセット（すなわち subsetting=true）、CID フォント（すなわち TrueType に対して autocidfont=true または encoding=unicode）。Acrobat はキャラクタを、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 キャラクタを与えることが可能です。
fontsize	（文字サイズ）文字サイズを、ユーザー座標で指定します。値 0 またはキーワード auto を与えると、Acrobat が文字サイズを矩形に合わせて調整します。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 のみ。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：なし
icondown	（テンプレートハンドル[1]。type=pushbutton のみ）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：なし
iconrollover	（テンプレートハンドル[1]。type=pushbutton のみ）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：なし
itemname	（ハイパーテキスト文字列。type=radiobutton・checkbox のみ。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

表 12.10 PDF_create_field( )・PDF_create_fieldgroup( ) によるフィールドプロパティオプション一覧

オプション	説明
item-namelist	（ハイパーテキスト文字列。type=listbox・combobox のみ）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 のみ。その場合は必須）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	（整数）フィールドの枠の線幅を、デフォルト単位（＝ポイント）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 のみ。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 のみ）フィールドの中のテキストのキャラクタ数の上限か、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 のみ）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 のみ）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 のみ）true にすると、テキストが入力時にビュレットかアスタリスクで表されます。デフォルト：false
position	（float かキーワードのリスト。type=pushbutton のみ）icon…オプション群で与えているテンプレートの、フィールド矩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、フィールド矩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 のみ。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 のみ）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

表 12.10 PDF_create_field( )・PDF_create_fieldgroup( ) によるフィールドプロパティオプション一覧

オプション	説明
sorted	（論理値。type=listbox・combobox のみ）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 のみ）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：true
strokecolor	（色）テキストの描線色。使える色空間：gray・rgb・cmyk。デフォルト：{gray 0}（＝黒）。
submitname	（ハイパーテキスト文字列。type=pushbutton の場合にのみ推奨）フォームを送信させたいインターネットアドレスの URL エンコード済み文字列。デフォルト：なし
taborder	（整数）フィールドの、他のフィールド群に対するタブ順序を指定します。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、作成順序がそのままタブ順序になります。
toggle	（論理値。PDF_create_fieldgroup( ) で type=radiobutton のみ）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）フィールドのツールヒントに表示させたいテキスト。ラジオボタンとグループでは、Acrobat はつねに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 のみ）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
unisonselect	（論理値。PDF_create_fieldgroup( ) で type=radiobutton かつ PDF 1.5 のみ）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）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：グローバルな usercoordinates パラメタの値

1. アイコンのテンプレートは、PDF_begin_template( ) 関数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。

表 12.11 PDF_create_field( )・PDF_create_fieldgroup( ) の barcode オプションのサブオプション一覧

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生成します。
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 の場合のみ。かつその場合は必須）2 個のバーコードモジュール間の垂直距離をピクセル単位で。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水平距離をピクセル単位で

12.6 しおり

C++ Java ***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）しおりのテキストを入れます。

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。

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：

- 一般オプション群：*errorpolicy*（表 2.6 参照）・*hypertextencoding*・*hypertextformat*（表 12.1 参照）
- 表 12.12 に従ったしおり制御オプション群：
  action・*destination*・*destname*・*fontstyle*・*index*・*open*・*parent*・*textcolor*

戻り値 生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 この関数は、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最終ページ、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。

スコープ 文書・ページ

表 12.12 PDF_create_bookmark( ) のオプション一覧

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ときは無視されます）しおりの移動先を表 12.5 に従って指定したオプションリスト。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ときは無視されます）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たないので、しおりを区切りとして使いたいときに有用でしょう。
fontstyle	（キーワード）しおりテキストのフォントスタイルを指定します：normal・bold・italic・bolditalic。デフォルト：normal

表 12.12 PDF_create_bookmark( ) のオプション一覧

オプション	説明
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生成したかのようにしおりが配置されます（しおりがページ順序を反映します）。
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生成されます。デフォルト：0
textcolor	（色）しおりテキストの色を指定します。使える色空間：none・gray・rgb。 デフォルト：rgb {0 0 0 }（＝黒）

12.7 PDF パッケージ・PDF ポートフォリオ

C++ Java ***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を追加します（要 PDF 1.7ext3）。

parent 親フォルダを、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。キャラクタ / ¥ : * ″ < > | は使ってはいけません。最後のキャラクタはピリオド「.」にしてはいけません）フォルダの名前。同じ親を持つ 2 個のフォルダが、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* が UTF-16 文字列の場合の長さ（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 2.6 参照）・*hypertextencoding*・*hypertextformat*（表 12.1 参照）
- 表 12.13 に従ったポートフォリオオプション群：*description*・*fieldlist*・*thumbnail*

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。このフォルダ構造は *PDF_end_document( )* の後に削除されます。この関数は、*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 文書

表 12.13 PDF_add_portfolio_folder( ) のオプション一覧

オプション	説明
description	（ハイパーテキスト文字列）フォルダの説明
fieldlist	（オプションリストのリスト）フォルダ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 12.15 に挙げます。
thumbnail	（画像ハンドル）フォルダのサムネールとして用いたい画像を指定します。このハンドルは、PDF_load_image( ) で作成しておく必要があり、この画像は、PDF_add_thumbnail( ) に対して挙げた条件を満たしている必要があります。

C++ Java ***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またはパッケージにファイルを追加します（要 PDF 1.7）。

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の場合は -1（PHP では 0）。ルートフォルダでないフォルダは要 PDF 1.7ext3 です。

filename （名前文字列。グローバルなオプションまたはパラメタに従って解釈されます。表 2.2 参照）PDF ポートフォリオの指定したフォルダ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* が UTF-16 文字列の場合の長さ（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：

- 一般オプション群：*errorpolicy*（表 2.6 参照）・*hypertextencoding*（表 12.1 参照）
- 表 12.14 に従ったファイルプロパティオプション群：
  description・*fieldlist*・*mimetype*・*name*・*password*・*thumbnail*

戻り値 ファイルの追加が成功したときは値 1、関数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、この関数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。この関数は、*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 文書

表 12.14 PDF_add_portfolio_file( ) のオプション一覧

オプション	説明
description	（ハイパーテキスト文字列）ファイルに関連づけられた説明テキスト。
fieldlist	（オプションリストのリスト）ファイル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 12.15 に挙げます。

表 12.14 PDF_add_portfolio_file( ) のオプション一覧

オプション	説明
mimetype	（文字列）ファイルの MIME タイプ。ただし Acrobat は、ファイルがアクティブにされたときに、適切なアプリケーションを起動するために、MIME 種別でなくファイル名の拡張子を用います。ファイルを PDF 文書として開くことに成功したときは、MIME タイプ application/pdf が自動的に設定されます。
name	（ハイパーテキスト文字列）ポートフォリオ内で用いたいファイルの名前を、filename と違う名前にしたいときに指定します。Acrobat が、PDF 文書を正しくプレビューし、その他の種類のファイルに対して適切なアプリケーションを起動するように、通常の種別固有の拡張子（.pdf 等）を持った名前を用いることを推奨します。同じフォルダの中の 2 個のファイルの名前は、大文字・小文字を無視したときに異なっている必要があります。名前の中にスラッシュキャラクタ「/」は入れないことを推奨します。Acrobat はスラッシュより前のキャラクタをすべて落としてしまうためです。デフォルト：filename からパス要素をすべて除いたもの
password	（文字列で最大 127 キャラクタ。PDF が利用できる場合のみ）保護された PDF 文書を開いてその日付項目群を取得するために必要な PDF マスターパスワード。
thumbnail	（画像ハンドル）ファイルのサムネールとして用いたい画像を指定します。このハンドルは、PDF_load_image( ) で作成しておく必要があり、この画像は、PDF_add_thumbnail( ) に対して挙げた条件を満たしている必要があります。

表 12.15 PDF_add_portfolio_folder( ) と PDF_add_portfolio_file( ) の fieldlist オプションのサブオプション一覧

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を指定します。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 12.16 PDF_end_document( ) の portfolio オプションのサブオプション一覧

オプション	説明
coversheet	（ハイパーテキスト文字列）ユーザーインタフェー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

表 12.16 PDF_end_document( ) の portfolio オプションのサブオプション一覧

オプション	説明
coversheet-folder	（フォルダハンドル）coversheet オプションで指定されたファイルを含んでいるポートフォリオの中のフォルダの名前。その coversheet 名のファイルが、複数のポートフィリオフォルダ内に存在しており、coversheetfolder が指定されていない場合には、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）初期表示を指定します。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」）。 ***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」）。 ***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」）。
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。右記の名前（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）ユーザーインタフェー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。右記の種別を用いて、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ー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。

表 12.16 PDF_end_document( ) の portfolio オプションのサブオプション一覧

オプション	説明
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フェースで並べ替える順序を指定します。各サブリストは、フィールド名（必須）とキーワード 1 個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を指定します。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。使えるサブオプション： ***direction*** （キーワード）分割バーの向き。使えるキーワード： ***horizontal*** ウィンドウを横に分割します。 ***vertical*** ウィンドウを縦に分割します。 ***none*** ウィンドウを分割しません。ウィ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）分割バーの初期位置を、得られるウィ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。この項目は、direction=none にすると無視されます。デフォルト：ビューア依存

13 3D・地理空間機能

13.1 3D アートワーク

クックブック　完全なコードサンプルがクックブックの multimedia/starter_3d トピックにあります。

C++ Java　***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（要 PDF 1.6）。

filename　（名前文字列。グローバルな*filenamehandling*オプションまたはパラメタに従って解釈されます。表 2.2 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* 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。

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：
- 一般オプション群：*errorpolicy*（表 2.6 参照）・*hypertextencoding*（表 12.1 参照）
- 表 13.1 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群：*defaultview*・*script*・*type*・*views*

戻り値　3D ハンドル。*PDF_create_annotation()* で 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　PRC または U3D 形式を内容として持つファイルから 3D モデルを読み込みます。

スコープ　**ページ・文書**。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 13.1 PDF_load_3ddata( ) のオプション一覧

オプション	説明
defaultview	（キーワードまたは 3D ビューハンドル）3D 注釈の初期ビューを指定します。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
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 13.1 PDF_load_3ddata( ) のオプション一覧

オプション	説明
type	（キーワード）3D データの種類を指定します（デフォルト：U3D）： ***PRC*** （PDF 1.7ext3）Product Representation Compact（PRC）形式（ISO 14739-1） ***U3D*** Universal 3D File 形式（U3D）。下記の種類が使えます（www.ecma-international.org 参照）： PDF 1.6、ただし要 Acrobat 7.0.7 以上：ECMA-363, Universal 3D File Format (U3D), 1st Edition。 PDF 1.6、ただし要 Acrobat 8.1 以上：ECMA-363, Universal 3D File Format (U3D), 3rd Edition。 なお、Acrobat 9.3.x・9.4.x では、U3D アートワークが表示されず、エラーメッセージ「A 3D data parsing error has occurred」が出る問題があります（Acrobat 8・X では大丈夫です）。
views	（3D ビューハンドルのリスト）3D モデルに対する定義済みビューのリスト。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

C++ Java ***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（要 PDF 1.6）。

username （ハイパーテキスト文字列）3D ビューのユーザーインタフェース名。

len （C 言語バインディングのみ）*username* 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。

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：

- 一般オプション群：*errorpolicy*（表 2.6 参照）・*hypertextencoding*（表 12.1 参照）
- 3D ビューのプロパティ群を表 13.2 に従って指定するオプション群：
  background・camera2world・cameradistance・lighting・name・rendermode・type・U3Dpath

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 **ページ・文書**。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 13.2 PDF_create_3dview( ) のオプション一覧

オプション	説明
background	（オプションリスト）3D モデルの背景を指定します： ***fillcolor*** （色）背景色を、RGB 色空間で表現します。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矩形にだけ適用されます。デフォルト：false

表 13.2 PDF_create_3dview( ) のオプション一覧

オプション	説明
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 (下記説明参照)。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 を指定しないと無視されます) 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を指定します。以下のオプションが使えます : *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ます。 *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 アクションで使うことができます。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モードを指定します。使えるサブオプションを表 13.3 に挙げます。
type	(キーワード。このビューが PDF_load_3ddata( ) で type=PRC で使われるなら必須) 3D データの種別を指定します (デフォルト : U3D) : *PRC* このビューは、PDF_load_3ddata( ) で type=PRC で使われます。 *U3D* このビューは、PDF_load_3ddata( ) で type=U3D で使われます。
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、このような 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。

下記の値を *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 (後述参照) :
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ちょうど 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 13.3 PDF_create_3dview( ) の rendermode オプションのサブオプション一覧

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。この色はいくつかのレンダモードで用いられます。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）いくつかのレンダモードにおける不透明値。デフォルト：0.5
rendercolor	（RGB カラー）補助色。この色はいくつかのレンダモードで用いられます。デフォルト：黒

表 13.3 PDF_create_3dview( ) の rendermode オプションのサブオプション一覧

オプション	説明
type	（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モードを指定します。使えるオプション： （キーワード）レンダモードを指定します。使えるキーワード（デフォルト：Artwork）： ***Artwork*** ランダモードは 3D アートワーク内で指定されます。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します。 ***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）群を表示します。 ***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）群を表示します。 ***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）群を表示します。 ***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）群を表示します。 ***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します。 ***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します。 ***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します。 ***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します。 ***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します。 ***Vertices*** 頂点のみを単色で表示します。 ***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します。 ***Illustration*** 表面を持ったシルエット辺を、隠線を除いて表示します。 ***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、隠線を除いて表示します。 ***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します。

13.2 地理空間機能

注 PDF 内の地理空間機能をインタラクティブ操作するには Acrobat 9 以上が必要です。

地理空間機能は、以下の関数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、ページに地理参照付き領域（複数可）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、地球ベース座標系を画像に割り当てることができます。

地理空間機能のためのオプション群を、表 13.4 以降に詳しく示します。

表 13.4 PDF_begin/end_page_ext( ) の viewports オプションのサブオプション一覧

オプション	説明
bounding-box	（矩形。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矩形。
georeference	（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を指定します。使えるオプションは表 13.5 を参照。
hypertext-encoding	（キーワード）name オプションのエンコーディングを指定します。空文字列は unicode と同等です。デフォルト：グローバルな hypertextencoding パラメタの値
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フェースに表示しません。

表 13.5 PDF_load_image( ) の georeference オプションのサブオプション一覧。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を指定します（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を指定します（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、右記の 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を指定します（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・画像領域全体が地図に用いられます。

表 13.5 PDF_load_image( ) の georeference オプションのサブオプション一覧。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 13.6 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を指定します（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、右記の 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 対 2 個以上のリスト。必須）数値のリストで、各対が 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 オプションを含む画像の矩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 対 2 個以上のリスト。必須）座標対のリストで、各対が、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。この対の数は、mappoints オプション内の対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 13.6 に従った世界座標系（worldpoints の解釈のための）。

表 13.6 PDF_load_image( ) 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一覧。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 9 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）座標系の種類を指定します： ***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。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します。また WKT は、Acrobat 9 では type=geographic のときは必須のようです。

14 文書交換

14.1 文書情報フィールド

C++ Java ***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。標準フィールド名のいずれか、または任意のカスタムフィールド名を用いることができます（表 14.1 参照）。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 要素セットを一読することを PDFlib ユーザーに奨励します。[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。なお、Windows 版の Adobe Reader 6 では、バグのため、情報文字列によってはキャラクタ *&* が正しく表示されないことがあります。

len （*PDF_set_info2()* のみ、かつ C バインディングのみ）*value* 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。

詳細 与える情報の値は、カレント文書にだけ使われるので、同じ**オブジェクト**スコープの中で生成する文書すべてに使われるわけではありません。*PDF_begin/end_document()* に *autoxmp* オプションを与えてあると、*PDF_set_info()* に与えた情報項目から、PDFlib が自動的に XMP 文書メタデータを同期作成します。

文書情報項目は、*PDF_begin/end_document()* の *metadata* オプションに与えた XMP 文書メタデータによって上書きされます。

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 14.1 文書情報フィールドキーの値

キー	説明
Subject	文書のサブタイトル
Title	文書のタイトル。この項目は PDF/X モードでは与える必要があります。
Creator	文書の作成に使用されたソフトウェア（PDF 生成ソフトウェアを示す Producer とは別です。Producer はつねに PDFlib になります）。Acrobat では、この項目は「アプリケーション」と表示されます。この項目は PDF/X モードでは与える必要があります。
Author	文書の作成者
Keywords	文書の内容を記述したキーワード

1. dublincore.org 参照

表 14.1 文書情報フィールドキーの値

キー	説明
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/X モードでは Unknown は使えません。
その他任意の名前	ユーザー定義の文書情報フィールド。PDFlib は、任意の数のフィールドに対応しています。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。同じフィールド名が複数回現れた場合は、最後のものが使われます。PDF_begin/end_document( ) の moddate オプションも参照。 XMP メタデータが作成される（autoxmp または metadata オプションによって）場合は、カスタム文書情報フィールドにはスペースキャラクタを一切含めてはいけません。 標準識別に用いられるフィールドは拒否されます。

14.2 XMP メタデータ

文書情報フィールドのかわりとして、またはそれに加えて、メタデータを指定するためのフレームワークとして、PDFlib は XMP（*Extensible Metadata Platform*[1]）に対応しています。XMP は、たとえば PDF/A 準拠のためには必須であり、これに対応するアプリケーションは増えてきています。PDFlib では、以下に述べるように、いくつかの種類の XMP 対応があります。

文書情報フィールドに対する自動 XMP 同期　*PDF_begin/end_document()* の *autoxmp* オプションを *true* にすると、PDFlib は、*PDF_set_info()* に与えた文書情報フィールドと、いくつかの内部的に生成された項目（*CreationDate* 等）を、文書レベルの XMP メタデータの中のそれぞれ対応する項目に同期させます。

標準 XMP スキーマの中のよく知られた要素に対応する文書情報フィールドは、適切なスキーマの中に配置されます。未知の情報フィールドは通常、拡張 PDF（*pdfx*）スキーマの中に配置されますが、ただし PDF/A モードでは無視されます。

ユーザーが与える XMP ストリーム　ユーザーは、XMP メタデータストリームの全体または一部分を、さまざまな関数の *metadata* オプションに与えることができます。このオプションは XMP ストリームをとり、それを検証します。PDFlib は自動的に、XDP パケットのヘッダとトレーラを生成します。

クックブック **簡単な XMP のサンプルがクックブックの `interchange/embed_xmp` トピックにあります。**

文書レベルのメタデータに対しては、PDFlib はいくつかの内部的に生成したプロパティを追加します（*CreationDate* 等）。PDF/A モードでは、PDFlib は、ユーザーが与える XMP ストリームの中の関連する項目を、標準文書情報フィールドに同期させます（*autoxmp* モードが文書情報フィールドを XMP に同期させるのと似ています）。しかし PDFlib は、その他の XMP 項目を、カスタムの文書情報フィールドに同期はさせません。PDF/A では XMP 文書メタデータについてさらなる要請がありますので、それについては PDFlib チュートリアルで解説します。

文書レベルのメタデータだけでなく、XMP は、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 ページに対しても与えることができます。さまざまな関数の *metadata* オプションのサブオプションを表 14.2 に示します。例：

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

1. *www.adobe.com/products/xmp* 参照

表 14.2 PDF_begin/end_document( )・PDF_begin/end_page_ext( )・PDF_load_font( )・PDF_load_iccprofile( )・PDF_load_image( )・PDF_begin_template_ext( )・PDF_open_pdi_page( ) の XMP メタデータオプション

オプション	説明
metadata	（オプションリスト。PDF 1.4）文書やその他のオブジェクトに対するメタデータを与えます。オプションリストには以下のオプションを入れることができます。 ***inputencoding*** （キーワード）与えるデータを解釈させたいエンコーディング。デフォルト：unicode ***inputformat*** （キーワード）与えるデータの形式。デフォルト：utf8（EBCDIC ベースのシステムでは ebcdicutf8）、ただし inputencoding を 8 ビットエンコーディングにしているときは bytes ***filename*** （名前文字列。必須）整形式の XMP メタデータの入ったディスクベースまたは仮想ファイルの名前。 例：metadata={filename=info.xmp inputencoding=winansi}

表 14.3 PDF_begin/end_document( )・PDF_begin/end_page_ext( )・PDF_load_font( )・PDF_load_iccprofile( )・PDF_load_image( )・PDF_begin_template_ext( )・PDF_open_pdi_page( ) の metadata オプションのサブオプション一覧

オプション	説明
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・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( ) のみ。filename との組み合わせ不可）画像内に存在する 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 画像形式内で効力を持ちます。画像ファイル内に 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 を与えなければ必須）整形式 XMP メタデータを内容として持つファイルの名前。これはグローバルな filenamehandling オプションまたはパラメタに従って解釈されます（表 2.2 参照）。

14.3 タグ付き PDF

タグ付き PDF を生成するには、*PDF_begin_document()* で *tagged* オプションを *true* に設定しておく必要があります。*lang* オプションも与える必要があります。タグ付き PDF モードは、*pdfa* オプションに *PDF/A-1a:2005* を設定しておくと自動的に有効になります。

クックブック 完全なコードサンプルがクックブックの interchange/starter_tagged トピックにあります。

C++ Java ***int begin_item(String tag, String optlist)***

Perl PHP ***int begin_item(string tag, string optlist)***

C ***int PDF_begin_item(PDF *p, const char *tag, const char *optlist)***

構造要素やその他の内容アイテムを開いて、オプションで与える属性を持たせます。

tag 表 14.4 に従ったアイテムの要素種別：

- 標準要素種別群
- 構造要素ではない擬似要素種別群
- タグ名 *Plib_custom_tag* は、カスタム要素種別を用いることを意味します。実際のタグ名は *tagname* オプションで与える必要があります。

表 14.4 PDF_begin_item( )・PDF_begin_mc( )・PDF_mc_point( ) の標準・擬似・カスタム種別一覧

カテゴリ	タグ
標準要素種別群	
グループ化	Document・Part・Art・Sect・Div・BlockQuote・Caption・TOC・TOCI・Index・NonStruct・Private
段落的	P・H・H1 ～ H6（BLSE）
リスト	L・LI・Lbl・LBody（BLSE）
表	Table（BLSE）・TR・TH・TD・THead[1]・TBody[1]・TFoot[1]
インラインレベル	Span・TagSuspect[2]・Quote・Note・Reference・BibEntry・Code（ILSE）
イラストレーション	Figure・Formula・Form
日本語	Ruby[1]（グループ化）・RB[1]・RT[1]・RP[1]・Warichu[1]（グループ化）・WT[1]・WP[1]
擬似要素種別群	
非構造化要素群	***Artifact*** ページの本文から区別するべきアーティファクトを指定します。 ***ASpan*** （アクセシビリティスパン。PDF には Span として書き込まれますが、インラインレベルアイテムの Span とは区別する必要があります）これを使うと、構造要素に属さない内容や、構造要素の一部分だけに似た内容に対して、アクセシビリティプロパティを与えることができます。 ***ReversedChars*** （非推奨）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を指定します。 ***Clip*** マーク付き切り抜きシーケンスを指定します。これは、クリッピングパスまたは表現モード 7 のテキストだけを含むシーケンスであり、表示されるグラフィックまたは PDF_save( ) / PDF_restore( ) を含みません。

表 14.4 PDF_begin_item( )・PDF_begin_mc( )・PDF_mc_point( ) の標準・擬似・カスタム種別一覧

カテゴリ	タグ
カスタム要素種別群	
ユーザー定義要素群	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。

1. 要 PDF 1.5 以上
2. 要 PDF 1.6 以上

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要素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：

- 一般オプション：*hypertextencoding*（表 12.1 参照）
- 表 14.5 に従ったタグ制御オプション群：
  ActualText・Alt・artifactsubtype・artifacttype・Attached・BBox・ColSpan・E・index・inline・Lang・parent・RowSpan・Scope・tagname・Title

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 この関数は、タグ付き PDF の根本である、文書の構造ツリーを生成します。構造ツリー内における新規要素の位置は、*parent・index* オプションで制御することができます。構造要素は任意の階層の入れ子が可能です。擬似要素種別は親アイテムとしては許されません。通常アイテムは、それを開いたページの中に縛られず、任意の数のページにわたって続けることもできます。

スコープ インラインアイテムの場合は**ページ**、通常アイテムの場合は**文書**も。対応する *PDF_end_item()* と必ず対にして呼び出す必要があります。この関数はタグ付き PDF モードでのみ使えます。

表 14.5 PDF_begin_item( )・PDF_begin_mc( )・PDF_mc_point( ) の構造・擬似タグオプション一覧

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。TagSuspect では不可）内容アイテムに対する等価な代替テキスト。何らかの標準的でない方法で表現しているテキスト内容に対して与える必要があります。たとえば合字や、イラストの飾り文字や、ドロップキャップなどです。このオプションを PDF 1.4 モードで使うときは、inline オプションを false に設定する必要があります。
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。TagSuspect では不可）内容アイテムに対する代替説明。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。画像に対する代替テキストは、アクセシビリティのためには必要です。このオプションを PDF 1.4 モードで使うときは、inline オプションを false に設定する必要があります。
artifact-subtype	（キーワード。tag=Artifact かつ artifacttype=Pagination。PDF 1.7）アーティファクトの副種別：Header・Footer・Watermark
artifacttype	（キーワード。tag=Artifact のみ）内容アイテムのアーティファクト種別を指定します：Pagination・Layout・Page・Background（PDF 1.7）
Attached	（キーワードのリスト。tag=Artifact かつ artifacttype=Pagination または artifacttype=Background でページ全体の背景アーティファクトを持つ場合のみ）キーワード Top・Bottom・Left・Right のいずれかを 1 個から 4 個まで入れたリスト

表 14.5 PDF_begin_item( )・PDF_begin_mc( )・PDF_mc_point( ) の構造・擬似タグオプション一覧

オプション	説明
BBox	（矩形。tag=Artifact、およびすべての表・イラストレーションタグのみ。artifacttype=Background なら必須、そうでないなら必須ではありませんが、折り返しのためには推奨）アーティファクトの外接枠を、デフォルト座標系で（usercoordinates=false にしているとき）、またはユーザー座標系（usercoordinates=true にしているとき）で指定します。このオプションを与えないと、PDFlib は、取り込んだ画像・PDF ページに対して自動的に BBox 項目を作成します。
ColSpan	（整数。tag=TH・TD のみ）表のセルがわたる列の数
E	（ハイパーテキスト文字列。ASpan 以外の擬似タグでは不可。TagSuspect では不可。構造タグに対しては要 PDF 1.5）内容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。
index	（整数。擬似タグと TagSuspect では不可）要素を親の中に挿入したい位置の番号。0 から、その時点の子の数までの値を使って、アイテムを親の中のその位置に挿入します。値 -1 を使うと、要素を最後のアイテムとして挿入することができます。デフォルト：-1
inline	（論理値。tag=ASpan と、TagSuspect 以外のすべてのインラインレベルタグのみ）true にすると、内容アイテムはインラインで書き込まれ、構造要素は作成されません。デフォルト：true
Lang	（文字列。TagSuspect と、ASpan 以外の擬似タグでは不可）内容アイテムの言語識別子を、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、文書全体の言語を、個々の内容アイテムについて上書きすることが可能です。
parent	（アイテムハンドル。TagSuspect と擬似タグでは不可）要素の親のアイテムハンドル。以前に PDF_begin_item( ) に呼び出して返されたハンドルです。値 0 にすると、構造ツリーのルートを参照します。-1 にすると、カレントページ上でもっとも最近に開かれた要素の親を参照します。いいかえれば、parent=-1 にすると、カレント要素の兄弟を開きます。擬似要素種別は親アイテムとしては許されません。デフォルト：-1
RowSpan	（整数。tag=TH・TD のみ）表のセルがわたる表行の数
Scope	（キーワード。tag=TH のみ。PDF 1.5 以上）その見出しセルが、それを含む表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表列内のそれなのか、あるいはそれを含む表行と表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。
tagname	（名前文字列。tag=Plib_custom_tag の場合のみ。その場合は必須）カスタムタグの任意の名前。これに対して、標準要素種別へのマッピングが、PDF_begin_document( ) の rolemap オプションで定義されている必要があります。カスタム要素種別は、winansi の 127 キャラクタか、または任意の Unicode キャラクタ群を内容とするキャラクタ列に限られます。後者の場合には、対応する UTF-8 列の長さが 127 バイトを超えないことが条件です。
Title	（ハイパーテキスト文字列。インライン・擬似タグでは禁止）構造要素の名前

C++ Java ***void end_item(int id)***

Perl PHP ***end_item(int id)***

C ***void PDF_end_item(PDF *p, int id)***

構造要素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 インラインアイテムはすべて、ページを終える前に閉じる必要があります。通常アイテムはすべて、文書を終える前に閉じる必要があります。ただし、メモリ消費を抑えるため

に、通常アイテムはすべて、完了次第閉じることを強く推奨します。アイテムは、そのすべての子をあらかじめ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 インラインアイテムの場合は**ページ**、通常アイテムの場合は**文書**も。対応する *PDF_begin_item()* と必ず対にして呼び出す必要があります。この関数はタグ付き PDF モードでのみ使えます。

C++ Java ***void activate_item(int id)***

Perl PHP ***activate_item(int id)***

C ***void PDF_activate_item(PDF *p, int id)***

以前に作成した構造要素やその他の内容アイテムをアクティブ化します。

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。まだ閉じられていてはいけません。擬似・インラインレベルアイテムはアクティブ化できません。

詳細 構造要素をいったん横に置いておき、後でまたアクティブ化するという方式を採用すると、多段組のレイアウトや、本文中に割り込むコラムなど、たとえ 1 つのページ上に複数の構造の枝が同時併行しているような場合でも、タグ付き PDF のページを柔軟に効率よく作成することができるようになります。

スコープ **文書・ページ**。この関数はタグ付き PDF モードでのみ使えます。

14.4 マーク付き内容

C++ Java ***void begin_mc(String tag, String optlist)***
Perl PHP ***begin_mc(string tag, string optlist)***
C ***void PDF_begin_mc(PDF *p, const char *tag, const char *optlist)***

マーク付き内容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。

tag マーク付き内容シーケンスの名前。以下のタグが使えます：
- 表 14.5 のすべてのインラインレベル・擬似タグ群。
- 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 *Plib_custom* を用いることができます。
- タグ *Plib* は予約済みです。

optlist 以下のマーク付き内容シーケンスオプション群が使えます：
- 表 14.5 に従った標準プロパティオプション群。
- タグ *Plib_custom*・*Plib* では表 14.6 のオプションも使えます。

詳細 指定したタグとプロパティ群を持ったマーク付き内容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。マーク付き内容シーケンスは任意の階層の入れ子が可能です。*PDF_begin/end_item()*・*PDF_begin/end_mc()* の入れ子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()* と必ず対にして、同じ**ページ・パターン・テンプレート・グリフ**スコープ内で呼び出す必要があります。

表 14.6 PDF_begin_mc( )・PDF_mc_point( ) によるタグのユーザー定義プロパティオプション

オプション	説明
properties	（オプションリストのリスト。tag=Plib・tag=Plib_custom のみ）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 ***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内容シーケンスを終了させます。

詳細 マーク付き内容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()* と必ず対にして、同じ**ページ・パターン・テンプレート・グリフ**スコープ内で呼び出す必要があります。

C++ Java	***void mc_point(String tag, String optlist)***
Perl PHP	***mc_point(string tag, string optlist)***
C	***void PDF_mc_point(PDF *p, const char *tag, const char *optlist)***

マーク付き内容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。

tag マーク付き内容ポイントの名前。以下のタグが使えます：

- 表 14.5 のすべてのインラインレベル・擬似タグ群。
- カスタム項目に対してはタグ *Plib_custom* を用いることができます。
- タグ *Plib* は予約済みです。

optlist 以下のオプションが使えます：

- 表 14.6 に従ったマーク付き内容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 14.6 のオプションも使えます。

詳細 指定したタグとプロパティ群を持ったマーク付き内容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内容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

A 全関数一覧

この付章ではすべての API 関数を挙げます。関数名をクリックするとその説明へ飛べます。

一般

get_value
set_value
get_parameter
set_parameter
set_option
new
delete
begin_document
begin_document_callback
end_document
get_buffer

begin_page_ext
end_page_ext
suspend_page
resume_page
define_layer
set_layer_dependency
begin_layer
end_layer

create_pvf
delete_pvf
get_errnum
get_errmsg
get_apiname
get_opaque

フォント

load_font
close_font
setfont
info_font
begin_font
end_font
begin_glyph
end_glyph
encoding_set_char

テキスト出力

set_text_pos
show
xshow
show_xy
continue_text
stringwidth

Unicode 変換

utf16_to_utf8
utf8_to_utf16
utf32_to_utf16
utf8_to_utf32
utf32_to_utf8
utf16_to_utf32

テキスト組版

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
makespotcolor
load_iccprofile
begin_pattern
end_pattern
shading_pattern
shfill
shading

画像

load_image
close_image
fit_image
info_image
begin_template_ext
end_template_ext
add_thumbnail

グラフィック状態

setdash
setdashpattern
setflat
setlinejoin
setlinecap
setmiterlimit
setlinewidth
initgraphics
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
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
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

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

文書交換

set_info
begin_item
end_item
activate_item
begin_mc
end_mc
mc_point

B 全パラメター一覧

PDF_get/set_parameter( ) と *PDF_get/set_value( )* のすべてのキーワードをこの付章に挙げます。キーワードをクリックするとその説明へ飛べます。

セットアップ	**ログ記録**	**フォント処理**	**単純テキスト出力**
PDF_get/set_parameter( ):	PDF_get/set_parameter( ):	PDF_get/set_parameter( ):	PDF_get/set_parameter( ):
errorpolicy	logging[1]	Encoding	autospace
filenamehandling	logmsg[1]	FontAFM	charref
license[1]		FontPFM	decorationabove
licensefile	**バージョン管理**	FontOutline	escapesequence
nodemostamp	PDF_get/set_parameter( ):	HostFont	fakebold
resourcefile	version[1]	PDF_get/set_value( ):	glyphcheck
scope	PDF_get/set_value( ):	font[2]	kerning
SearchPath[1]	major	fontsize[2]	underline・overline・strikeout
string[1]	minor		textformat
asciifile	revision[2]	**画像**	PDF_get/set_value( ):
		PDF_get/set_parameter( ):	charspacing
PDF_get/set_value( ):	**ページ**	honoriccprofile	horizscaling
compress	PDF_get/set_parameter( ):	renderingintent	italicangle
maxfilehandles	topdown		leading
	PDF_get/set_value( ):		strokewidth
グラフィック	pagewidth		textrendering
PDF_get/set_parameter( ):	pageheight		textrise
fillrule			textx[2]・texty[2]
PDF_get/set_value( ):	**ICC プロファイル**		underlineposition
currentx[2]・currenty[2]	PDF_get/set_parameter( ):		underlinewidth
ctm_a[2]・ctm_b[2]・ctm_c[2]	ICCProfile	**インタラクティブ**	wordspacing
ctm_d[2]・ctm_e[2]・ctm_f[2]	StandardOutputIntent	PDF_get/set_parameter( ):	
	PDF_get/set_value( ):	usercoordinates	**PDI**
色	icccomponents[2]	hypertextencoding	PDF_get/set_parameter( ):
PDF_get/set_parameter( ):	setcolor:iccprofilegray	hypertextformat	pdi[1]
preserveoldpantonenames	setcolor:iccprofilergb	usehypertextencoding	
spotcolorlookup	setcolor:iccprofilecmyk		

1. PDF_get_parameter( ) のみ
2. PDF_get_value( ) のみ

C 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こへ飛びます。

B

C

D

E

F

G

H

I

K

L

M

N

O

P

R

S

T

U

V

W

X

Y

Z

D 改訂履歴

日付	更新点
2011 年 7 月 11 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
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

日付	更新点
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

なお、パラメタとオプションは別途付章に示してあります。

R

PDFlib GmbH

PDFlib GmbH
Franziska-Bilek-Weg 9
80339 München, Germany
www.pdflib.com
電話 +49・89・452 33 84-0
FAX +49・89・452 33 84-99

ご質問があるときは、PDFlib メーリングリストと
tech.groups.yahoo.com/group/pdflib にあるアーカイブをチェックしてください

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）