

13 PPS と PDFlib Block Plugin

PDFlib Personalization Server (PPS) は、可変データ処理のための、テンプレートをを用いた PDF ワークフローに対応しています。ブロックという概念を用いて、取り込んだページに、外部情報源から引き出した任意量の 1 行ないし複数行のテキスト・画像・PDF ページ・ベクトルグラフィックを入れ込むことができます。これを利用すれば、PDF 文書のカスタマイズを必要とするアプリケーションを容易に実装できます。たとえば：

- ▶ メールへの連結
- ▶ ダイレクトメールの宛名印刷
- ▶ 納品書・明細等の発行
- ▶ 名刺の項目内容を各人ごとに変更

PDFlib Block Plugin を使えば、ブロックを対話的に作成・編集することができ、フォームフィールド変換プラグインを使えば、既存の PDF フォームフィールドを PDFlib ブロックに変換することができます。ブロックへは、PPS を使って流し込みを行うことができます。Block Plugin には、内蔵バージョンの PPS が含まれていますので、PPS によるブロックへの流し込み結果を Acrobat 上でプレビューすることができます。

注 ブロック処理を利用するには PDFlib Personalization Server (PPS) が必要です。PPS はすべての PDFlib 商用パッケージに含まれていますが、PPS に対するライセンスキーをご購入いただく必要があります。PDFlib や PDFlib+PDI のライセンスキーだけではご利用いただけません。PDF テンプレートに対話的に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 可変データとブロックに関するコードサンプルが PDFlib クックブックの blocks カテゴリにあります。

13.1 PDFlib Block Plugin をインストール

Block Plugin は、以下のバージョンの Acrobat で動作します (Adobe Reader では動作しません)：

- ▶ Windows : Acrobat X/XI
- ▶ Windows : Acrobat Standard・Pro 2017/2020
- ▶ Windows : Acrobat Pro DC (32 ビット・64 ビット)
- ▶ macOS : Acrobat Standard・Pro 2017/2020 (Intel)
- ▶ macOS : Acrobat DC (Intel・M1 用ユニバーサルバイナリ)

Windows 版 Acrobat DC には 32 ビット版と 64 ビット版がありますので 2 種類のインストーラがあります。インストールされている Acrobat のバージョンに合致しているインストーラを使うことが重要です。

Windows で PDFlib Block Plugin をインストール PDFlib Block Plugin と PDF フォームフィールド変換プラグインを Acrobat にインストールするには、プラグインのファイルを Acrobat のプラグインフォルダのサブディレクトリに入れる必要があります。これは、プラグインのインストーラによって自動的に実行されますが、手作業でもできます。プラグインのファイル名は **Block.api** と **AcroFormConversion.api** です。

64 ビット Windows 上の Acrobat 32 ビットの場合、プラグインフォルダは典型的には以下のとおりです：

C:\Program Files (x86)\Adobe\Acrobat DC\Acrobat\plug_ins\PDFlib Block Plugin

Acrobat 64 ビットの場合、プラグインフォルダは典型的には以下のとおりです：

C:\Program Files\Adobe\Acrobat DC\Acrobat\plug_ins\PDFlib Block Plugin

macOS で Acrobat DC 用 PDFlib Block Plugin をインストール このプラグインをすべてのユーザーのためにインストールするには以下のように操作します：

- ▶ ディスクイメージをダブルクリックすることによってマウントします。プラグインファイル群が入ったフォルダが現れます。
- ▶ このプラグインフォルダを、システムのライブラリフォルダの中の、以下のパスへ複製します (*Plug-ins* フォルダがまだない場合には作成します)：

/Library/Application Support/Adobe/Acrobat/XXX/Plug-ins

あるいは、以下のように操作することによって、このプラグインを、単独のユーザーのためだけにインストールすることもできます：

- ▶ デスクトップをクリックすることによって Finder 内に確実にいるようにしてから、**Option** キーを押しながら「移動」→「ライブラリ」を選択することによって、そのユーザーのライブラリフォルダを開きます。
- ▶ プラグインフォルダを、そのユーザーのライブラリフォルダの中の、以下のパスへ複製します (*Plug-ins* フォルダがまだない場合には作成します)：

/Users/<ユーザー名>/Library/Application Support/Adobe/Acrobat/XXX/Plug-ins

多言語インタフェース PDFlib Block Plugin は、ユーザーインタフェース内で複数言語に対応しています。Acrobat のアプリケーション言語に従って、Block Plugin はそのインタフェース言語を自動的に選択します。目下、日本語・英語・ドイツ語のインタフェースが利用可能です。Acrobat がこれ以外の言語モードで動作している場合には、Block Plugin は英語インタフェースを使用します。

Windows 版 Acrobat DC のためのサンドボックス保護 Acrobat DC 2020 では、サンドボックス保護という新しいセキュリティモデルが導入されました。これを有効にするには「環境設定」→「セキュリティ(拡張)」→「起動時に保護モードを有効にする(プレビュー)」・「保護されたビュー」を選択します。これが有効になっていると、さまざまな操作が制限され、文書ウィンドウの上端の黄色い帯にセキュリティメッセージが表示されます。サンドボックス保護について詳しくは：

helpx.adobe.com/acrobat/using/whats-new/2020-august.html

www.adobe.com/devnet-docs/acrobatetk/tools/AppSec/sandboxprotections.html

サンドボックスが有効になっていると、PDFlib Block Plugin のプレビュー機能に影響します。保護ビューは、デフォルトでは、Acrobat の *AppData* ディレクトリと、temp ディレクトリなどいくつかのディレクトリへのアクセスを許しますが、任意のユーザーディレクトリへのアクセスを許しません。Block Plugin が読み書きできるディレクトリは、保護ビューのデフォルトディレクトリリストに含まれているか、以下の場所にあるポリシーファイルの中で構成(ホワイトリスト)されているものに限られます(Acrobat の 32 ビット版の場合と 64 ビット版の場合)：

C:\Program Files (x86)\Adobe\Acrobat DC\Acrobat\PDFlibBlockCustomPolicies.txt

C:\Program Files\Adobe\Acrobat DC\Acrobat\PDFlibBlockCustomPolicies.txt

デフォルトではこのポリシーファイルは、以下のディレクトリへのアクセスを許していますが、管理者はこれにディレクトリ名を追加することもできます：

```
; Protected Path Section
FILES_ALLOW_ANY = C:\Users\\*.*
FILES_ALLOW_ANY = C:\Users\Public\*.*
```

もし保護モードか保護ビューが有効にされていて、かつ、使用されているディレクトリがホワイトリストされていない場合には、Block Plugin の機能のうち、プレビューや、ブロックの取り込み／書き出しなどは、失敗する可能性があります。

トラブルシューティング PDFlib Block Plugin が動作しないように見られる場合は、以下をチェックしてください：

- ▶ 「編集」 → 「環境設定」 (→ 「一般 ...」) → 「一般」で「承認されたプラグインのみを使用」チェックボックスがオフになっていることを確認してください。Acrobat が承認済みモードで動作していると、プラグインは読み込まれません。
- ▶ Adobe Designer または Adobe Experience Manager によって作成された PDF フォームは、Block Plugin の適切な動作を妨げることがあります。他の Acrobat のプラグインの動作についても同様です。なぜならこうしたフォームは、Acrobat の内部セキュリティモデルと衝突するためです。ですので、Designer の静的な PDF フォームは利用せずに、動的な PDF フォームだけを Block Plugin への入力として用いることを推奨します。

13.2 ブロックの概念の概要

13.2.1 文書デザインとプログラムコードの分離

PDFlib のデータブロックを利用すると、取り込んだページ上に、可変のテキストや画像や PDF ページやベクトルグラフィックを簡単に配置できます。単純な PDF ページと違って、データブロックを含むページは、後でサーバサイドで行われるべき処理についての情報を内部に持っています。PDFlib ブロックの概念は、以下の 2 種類の作業を完全に分離するものです：

- ▶ デザイナーはページレイアウトを作成し、可変ページ構成要素の位置を指定するとともに、その文字サイズ・色・画像拡張といった関連特性群も指定します。レイアウトは PDF 文書として作成し、その後デザイナーは、Acrobat 用 PDFlib Block Plugin を使って、可変データブロックとそのそれぞれのプロパティを指定します。
- ▶ プログラマーは、取り込まれる PDF ページ上の PDFlib ブロックに含まれる情報を、データベースのフィールドといった動的な情報と紐づけるコードを書きます。プログラマーは、ブロックの詳細については何も知らなくてよく（名前を含むのか ZIP コードを含むのか、ページ上の正確な位置、書式など）、そのため、どのようなレイアウト変更からも独立でいられます。ブロックに関連する詳細についてはすべて、ファイル内のブロックプロパティに基づいて PPS の側で処理します。

言い換えれば、プログラマーによって書かれるコードは「データ非依存」です。すなわちそれは汎用であり、ブロックのいかなる特性にも依存しません。たとえばデザイナーは、手紙の宛先を入れるブロックをページ上の別の場所へ移動させるかもしれませんし、あるいは、文字サイズを変えるかもしれません。一般的なブロック処理コードに変更を加える必要はなく、デザイナーがブロックプロパティを Acrobat プラグインで変更してラストネームのかわりにファーストネームを用いるようにしさえすれば、正しい出力が生成されます。

中間ステップとして、ブロックへの流し込みは Acrobat でプレビューできますので、開発と試験サイクルを迅速化することが可能です。ブロックプレビューには、ブロックの定義内で指定されたデフォルトデータ（文字列や画像ファイル名等）が用いられます。

13.2.2 ブロックのプロパティ

ブロックの動作はブロックのプロパティで制御できます。Block Plugin を使ってブロックにプロパティを割り当てます。

定義済みブロックプロパティ ブロックはページ上の長方形として定義され、名前・種類・その他自由なプロパティ群を割り当てられます。こうしたプロパティは後で PPS によって処理されます。名前は、ブロックを識別する任意の文字列であり、たとえば *firstname*・*lastname*・*zipcode* のように名づけることができます。PPS では、さまざまな種類のブロックを使うことができます：

- ▶ **テキスト行ブロック**は、1 行のテキストデータを持ちます。このデータは、PPS のテキスト行メソッドで処理されます。
- ▶ **テキストフローブロック**は、1 行ないし複数行のテキストデータを持ちます。複数行のテキストは PPS のテキストフローフォーマッタによって組版されます。複数のテキストフローブロックを連結して、前のブロックからあふれたテキストを次のブロックに入れることも可能です（393 ページ「テキストフローブロックを連結」参照）。
- ▶ **画像ブロック**は、ラスタ画像を持ちます。これは、DTP アプリケーションで TIFF や JPEG のファイルを貼り付けるのと似ています。

- ▶ **PDFブロック**は、他のPDF文書のページから取り込んだ任意のPDF内容を持ちます。これは、DTPアプリケーションでPDFページを貼り付けるのと似ています。
- ▶ **グラフィックブロック**は、ベクトルグラフィックを持ちます。これは、レイアウトアプリケーション内でSVGファイルを貼り付けるのに似ています。

ブロックは、その種類によって異なるさまざまなプロパティを持っています。プロパティは、Block Plugin で作成・変更することができます (377 ページ「13.3.2 ブロックプロパティを編集」参照)。ブロックプロパティの全一覧が 396 ページ「13.7 ブロックのプロパティ」にあります。たとえば、テキストブロックではテキストのフォントやサイズを指定することができ、画像ブロックや PDF ブロックでは拡大縮倍率や回転を指定することができます。PPS はブロックの種類ごとに、それを処理するための専用の関数を提供しています (**PDF_fill_textblock()** 等)。こうした関数は、取り込まれた PDF ページの中でブロックを名前前で検索し、そのプロパティを分析して、クライアントの与えたデータ (一行テキスト・複数行テキスト・ラスタ画像・PDF ページ・ベクトルグラフィックのいずれか) を、新しいページ上に、指定されたブロックプロパティに従って配置します。プログラマーは、その照応する、ブロック流し込み関数内のオプションを指定することによって、ブロックプロパティをオーバーライドすることもできます。

デフォルト内容に関するプロパティ 特殊なブロックプロパティを定義して、そのブロックのデフォルト内容を持たせることもできます。すなわち、ブロック流し込み関数に可変データが与えられていないときや、あるいはブロック内容が次の印刷実行時には変わりうるけれども現時点では不変であるようなときに、そのブロックに配置される、テキスト・画像・PDF・グラフィックいずれかの内容です。

デフォルトプロパティは、Block Plugin のプレビュー機能でも用いられます (386 ページ「13.5 ブロックを Acrobat でプレビュー」参照)。

カスタムブロックプロパティ 定義済みブロックプロパティを利用することにより、可変データ処理アプリケーションを手軽に実装することができますが、こうしたプロパティは、PPS の内部的に既知で自動処理可能なものに限られてしまいます。より高い柔軟性を与えるために、デザイナーは、カスタムプロパティをブロックに割り当てることもできます。カスタムプロパティを利用すれば、ブロックの概念を拡張して、より高度な可変データ処理アプリケーションの要請に応えることが可能です。

カスタムプロパティに関してはいかなる規則も存在しません。なぜなら PPS はカスタムプロパティに対してはいかなる処理も行わないからです。PPS はただ、カスタムプロパティをクライアントが利用できるようにするだけです。クライアントコードは、カスタムプロパティを取得し、適切に処理することができます。ブロックのカスタムプロパティに基づいて、アプリケーションがレイアウト関連やデータ抽出関連の決定を行えるようにすることも可能です。たとえば、科学アプリケーションのためのカスタムプロパティであれば、数値出力の桁数を指定することもできるでしょうし、あるいは、データベースのフィールド名をカスタムブロックプロパティとして定義しておいて、そのブロックに照応するデータを取得するために用いることもできるでしょう。

13.2.3 PDF のフォームフィールドを使わない理由

経験ある Acrobat ユーザーならば、なぜ我々は新たにブロックという概念を導入したのか、どうして PDF にすでにあるフォームフィールドのしくみを活用しないのか、疑問を抱かれるかもしれません。そもそもの違いは、PDF のフォームフィールドは対話的に記入されることを主眼として作られているのに対して、PDFlib のブロックは自動的に流し込まれることを目的としている点です。対話的記入と自動流し込みの両方を必要とするアプリケー

ションの場合であれば、フォームフィールド変換プラグインを用いて、PDF フォームと PDFlib ブロックを併用することも可能です (383 ページ「13.4 PDF フォームフィールドを PDFlib ブロックに変換」参照)。

両概念の間には類似点も多くありますが、PDFlib ブロックには PDF フォームフィールドと比較して表 13.1 に示すようないくつかの利点があります。

表 13.1 PDF フォームフィールドと PDFlib ブロックの比較

機能	PDF フォームフィールド	PDFlib ブロック
設計の趣旨	対話的利用	自動流し込み
文字組版機能 (フォント指定・文字サイズ指定よりも高度な)	—	カーニング・単語間隔・文字間隔・下線 / 上線 / 取り消し線
OpenType レイアウト機能	—	何ダースもの OpenType レイアウト機能 (合字・スワッシュ文字・オールドスタイル数字等)
複雑用字系への対応	制約あり	シェーピング・双方向テキスト (アラビア文字・デーヴァナーガリー等)
フォント制御	フォント埋め込み	フォント埋め込み・サブセット化・エンコーディング
テキスト組版制御	左・中央・右揃え	左・中央・右・両端揃え。各種組版アルゴリズム・制御。インラインオプションを用いてテキストの見映えを制御可能
テキストの途中でフォントその他のテキスト属性を変えられる	—	○
追加結果が PDF のページ記述に組み込まれる	—	○
ユーザーが追加フィールドの内容を編集可能	○	×
プロパティの拡張セット	—	○ (カスタムブロックプロパティ)
画像ファイルを流し込める	—	BMP・CCITT・PNG・JPEG・JBIG2・JPEG 2000・TIFF
ベクトルグラフィックを流し込める	—	SVG
カラー対応	RGB	グレースケール・RGB・CMYK・Lab・スポットカラー (HKS・Pantone スポットカラーが Block Plugin に内蔵)・DeviceN
PDF 各種規格	—	PDF/A・PDF/X・PDF/VT・PDF/UA
グラフィックやテキストのプロパティを流し込み時にオーバーライド可能	—	○
透過内容	—	○
テキストブロック群を連結可能 (あるブロックから次のブロックへテキストが流れる)	—	○

13.3 PDFlib Block Plugin でブロックを編集

13.3.1 ブロックを作成

ブロックツールをアクティブにする PDFlib ブロックを作成するための Block Plugin は、Acrobat におけるフォームツールと同様です。ページ上のすべてのブロックは、ブロックツールがアクティブな時に表示されます。Acrobat の他のツールが選択されるとブロックは見えなくなりますが、なくなったわけではありません。ブロックツールをアクティブにするには、以下のいずれかの操作を行います：

- ▶ ブロックアイコン  をクリック。これは以下の場所にあります：
Acrobat DC：「ツール」→「PDF を編集」をクリック。
- ▶ メニュー項目「PDFlib ブロック」→「PDFlib ブロックツール」を選択。

ブロックを作成・変更 ブロックツールをアクティブにしたら、十字ポインタをドラッグすれば、ページ上の希望の位置に希望のサイズのブロックを作成することができます。ブロックは必ず長方形で、その辺はページの辺と平行になります（ブロックの内容をページの辺と平行でなくするには *rotate* プロパティを用います）。ブロックの長方形をドラッグし終わると、「PDFlib ブロックプロパティ」ダイアログが現れるので、ブロックのさまざまなプロパティを編集することができます（377 ページ「13.3.2 ブロックプロパティを編集」参照）。ブロックツールはブロックの名前を自動的に作成しますが、この名前はプロパティダイアログで変更することもできます。ブロック名はページ内では一意である必要がありますが、別のページでは同じ名前も使えます。

ダイアログの上端では、ブロックの種類を「Textline」（テキスト行）・「Textflow」（テキストフロー）・「Image」（画像）・「PDF」・「Graphics」（グラフィック）のいずれかに変更できます。ブロックの種類ごとに異なる色が用いられています（図 13.1 参照）。タブは、どのブロックの種類を選択しているかに応じて、一度に 1 つだけがアクティブになっています。「PDFlib ブロックプロパティ」ダイアログは、ブロックの種類に応じて、プロパティを階層的にいくつかのグループやサブグループにまとめて表示します。

注 PDF にブロックを追加したり、既存のブロックに変更を加えたりした後は、Acrobat の「名前を付けて保存 ...」コマンドを使うほうがファイルサイズが小さくなります（「上書き保存」よりも）。

ブロックを選択 コピー・移動・削除・プロパティ編集といったいくつかのブロック操作は、選択した 1 個ないし複数のブロックに対して動作します。ブロックツールを用いてブロックを選択するには、以下のように操作します：

- ▶ 1 個のブロックを選択するには、単にそれをシングルクリックします。
- ▶ 複数のブロックを選択するには、Shift キーを押しながら 2 個目以降のブロックを選択します。
- ▶ ページ上のすべてのブロックを選択するには、Ctrl+A（Windows の場合）か Cmd+A（macOS の場合）を押すか、または「編集」→「すべて選択」を用います。

コンテキストメニュー 1 個ないし複数のブロックを選択している時には、コンテキストメニューを開けば、ブロック関連のいろいろな機能（「PDFlib ブロック」メニューから利用できる諸機能と同じ）をすばやく実行することができます。コンテキストメニューを開くには、選択した 1 個ないし複数のブロックを、Windows の場合はマウスの右ボタンでクリックし、macOS の場合は Ctrl+ クリックします。たとえば、ブロックを削除するには、

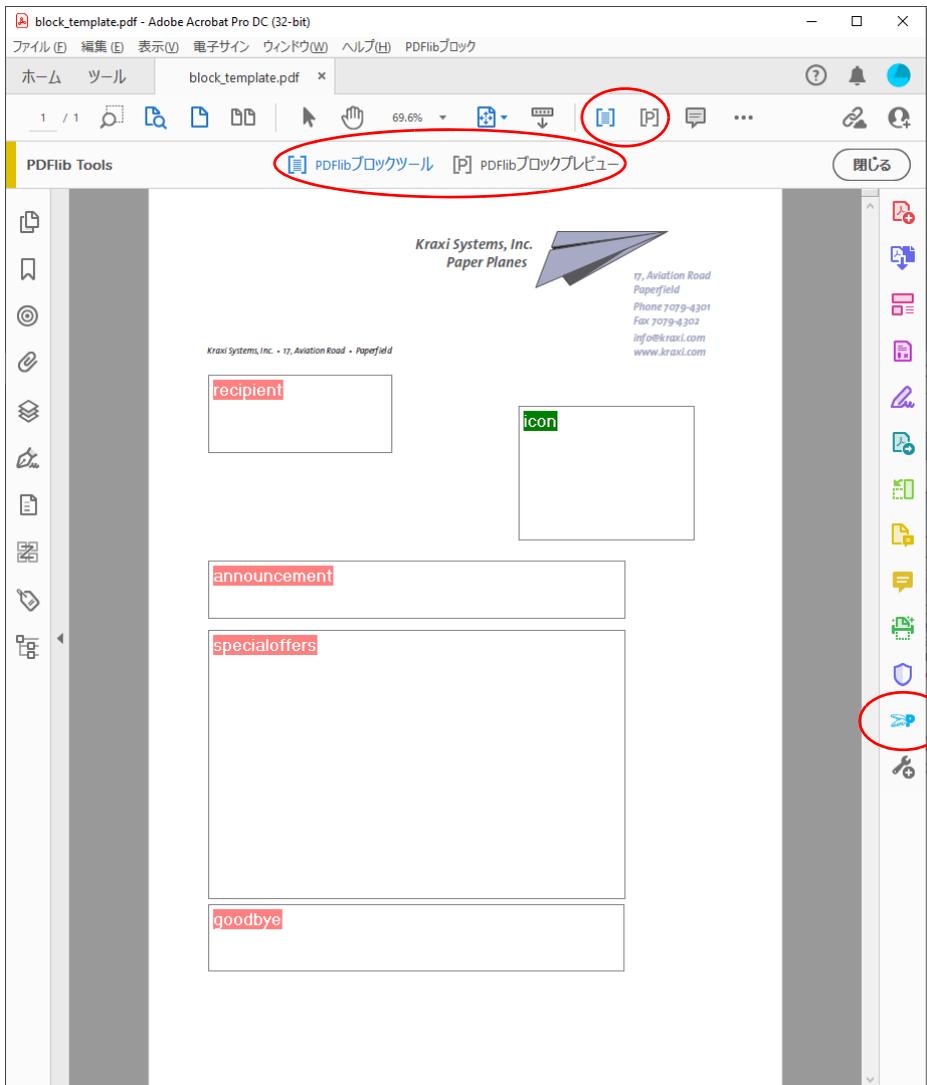


図 13.1
各種ブロック
の視覚化

それをブロックツールで選択したのち、**Delete** キーを押してもよいですし、あるいはコンテキストメニューで「編集」→「削除」を用いることもできます。

ページ上でブロックのない領域を右クリック (macOS では Ctrl+ クリック) すると、そのコンテキストメニューの中には、ブロックプレビューを作成するための項目と、ブロック機能を構成するための項目があります。

ブロックのサイズと位置 ブロックツールを使うと、選択した 1 個ないし複数のブロックを別の位置へ動かすことも可能です。Shift キーを押しながらブロックをドラッグすると、横方向か縦方向にだけ動きます。これはブロックを正確に整列させたいときに便利でしょう。ポインタがブロックの角の近くにある時は、ポインタアイコンが二重矢印に変わり、ブロックのサイズを変えることができます。カーソルがブロック領域内にある間は、ブロックの幅と高さがブロックの後に表示されます。

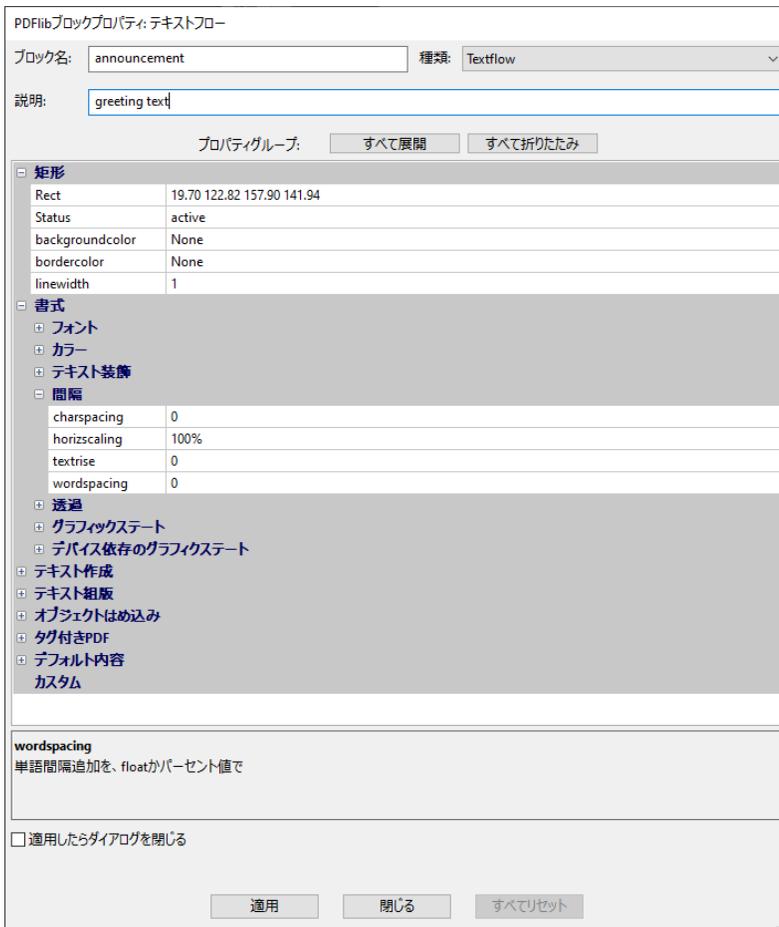


図 13.2
ブロックプロパティダイアログ

複数のブロックの位置やサイズを調整したいときは、複数のブロックを選択して、「PDFlib ブロック」メニューかコンテキストメニューから「整列」・「中央揃え」・「均等配置」・「サイズ」のいずれかのコマンドを選択します。1 個ないし複数のブロックの位置を、矢印キーを使って小刻みに（1 ポイントずつ）変えることもできます。矢印キーを使ってブロックを動かしながら *Shift* キーを押すと刻みは 10 ポイントです。ただし、この刻みはスクリーンピクセル群へ変換されますので、ブロック長方形の座標の実際の変動は 1 または 10 ポイントとはわずかに異なる場合があります。

あるいは、ブロックの座標を数値でプロパティダイアログに入力することもできます。座標系の原点はページの左上隅です。座標は、その時点で Acrobat で選択されている単位で表示されます：

- ▶ Acrobat DC で表示単位を変更するには、以下のように操作します：「編集」→「環境設定」（→「一般 ...」）→「単位とガイド」→「ページと定規の単位」を選択し、インチ・センチメートル・パイカ・ポイント・ミリのいずれかを選ぶ。
- ▶ カーソルの座標を表示するには、「表示」→「表示切り替え」→「カーソル座標」を選択します。

ただしここで選択されている単位は *Rect* プロパティに対してのみ効力を持ち、それ以外の数値プロパティ (*fontsize* 等) に対しては一切効力を持ちません。

グリッドを用いてブロックを位置付け Acrobat のグリッド機能を活用して、ブロックの位置付けやサイズ変更を正確に行うこともできます：

- ▶ グリッドを表示：「表示」→「表示切り替え」→「定規とグリッド」→「グリッド」。
- ▶ グリッドスナップを有効に：「表示」→「表示切り替え」→「定規とグリッド」→「グリッドにスナップ」。
- ▶ グリッドを変更：「編集」→「環境設定」(→「一般...」)→「単位とガイド」を選択します。そこでグリッド線の間隔や位置や色を変えることができます。

「グリッドにスナップ」を有効にしていると、ブロックのサイズと位置は、構成したグリッドに揃います。「グリッドにスナップ」は、新規作成したブロックにも効きますし、既存のブロックをブロックツールで移動したりサイズを変えたりするときにも効きます。

画像やグラフィックを選択することによってブロックを作成 手動でブロックの長方形をドラッグするのではなく、既存のページ内容を使ってブロックのサイズを定義することもできます。まず、メニュー項目「*PDFlib* ブロック」→「オブジェクトをクリックでブロックを定義」を有効にします。これで、ブロックツールを使って、ページ上の画像をクリックして、その画像と同じ位置に同じサイズのブロックを作成することができます。それ以外のグラフィックオブジェクトをクリックすることもでき、その場合、ブロックツールはそのグラフィック (たとえばロゴ) 全体を選択しようとしています。この「オブジェクトをクリック」機能は、ブロック定義作業を補助するために設けてあるものです。できたブロックの位置やサイズを変更したければ、後から何ら制約なく行うことができます。ブロックは、画像やグラフィックに固定されてしまうのではなく、ただそれを位置やサイズの決定の補助として用いるだけです。

この「オブジェクトをクリック」機能は、どのベクトルグラフィックや画像がページ上で論理的要素を形づくっているかを認識しようとしています。いずれかのページ内容がクリックされると、その対象が白かったり非常に大きかったりしない限り、その外接枠 (対象を囲む長方形) が選択されます。その次の段階として、この検知された長方形に一部入り込んでいる他の物が選択領域に追加され、これが繰り返されます。そうして最終的にできた領域に基づいてブロックの長方形が生成されるのです。結局のところ「オブジェクトをクリック」機能は、個々の線ではなくグラフィック全体を選択しようとするようになります。

フォントプロパティを自動検出 Block Plugin は、テキスト行またはテキストフローブロックの位置付けられた場所の背景にあるフォントを分析することができ、ブロックの照応するプロパティを自動的に書き込むことができます：

`fontname · fontsize · fillcolor · charspacing · horzscaling · wordspacing ·
textrendering · textrise`

フォントプロパティの自動検出は望ましくない結果をもたらすこともあるので、背景を無視させたい場合は、「*PDFlib* ブロック」→「背景フォント・色の検出」を用いて機能の有効・無効を切り替えることができます。デフォルトではこの機能は無効になっています。

ブロックをロック ブロックは、うっかり移動したりサイズを変えたり削除したりされないよう、ロックして保護することができます。ブロックツールがアクティブな状態で、ブロックを選択し、そのコンテキストメニューから「ロック」を選びます。ブロックが

ロックされていると、移動させることもサイズを変えることも削除することもできず、そのプロパティダイアログを編集することもできません。

13.3.2 ブロックプロパティを編集

新規ブロックを作成した時や、既存ブロックをダブルクリックした時や、ブロックのコンテキストメニューから「プロパティ」を選択した時には、プロパティダイアログが現れて、その選択したブロックに関するすべての設定を編集することができます（図 13.2 参照）。396 ページ「13.7 ブロックのプロパティ」で詳述するように、プロパティにはブロックの種類に応じて、いくつかのグループがあります。

「適用」ボタンは、ダイアログ内の 1 個ないし複数のプロパティを変更した時にのみ有効になります。プロパティの変更をブロックに適用した後は、ブロックの名前の中にアスタリスクが表示されて、ブロックが変更されたけれどもまだディスクには保存されていないということを示します。「適用」ボタンは、ロックされたブロックについては無効となります。

注 ブロックの種類やプロパティの設定によっては、表示されないプロパティもあります。たとえば、タブ設定を編集するための「hortabmethod=ruler におけるルーラタブ」プロパティサブグループは、「テキスト組版」→「ルーラタブ」グループの hortabmethod プロパティが ruler に設定されているときにのみ表示されます。

注 ブロックプロパティにテキストを入力すると、ストレート引用符がスマート引用符に置換されるなど、キャラクタの置換が起きることがあります。この置き換えはオペレーティングシステムが行なっていることですので、無効にするには「システム環境設定」→「キーボード」→「ユーザ辞書」→「スマート引用符とスマートダッシュを使用」を選択します。

プロパティの値を変更するには、入力したい数や文字列をそのプロパティの入力領域に入力するか（例： *linewidth*）、ドロップダウンリストから値を選ぶか（例： *fitmethod · orientate*）、またはダイアログの右側にある「...」ボタンをクリックしてフォントや色の値やファイル名を選択します（例： *backgroundcolor · defaultimage*）。 *fontname* プロパティの場合は、システムにインストールされているフォントの一覧から選ぶこともできますし、カスタムのフォント名を打ち込むこともできます。フォント名を入力した方式にかかわらず、そのフォントは、PPS によってブロックへ内容が流し込まれるシステム上において利用可能になっていることが必要です。

変更されたプロパティは、ブロックプロパティダイアログ内で太字で表示されます。ブロックのいずれかのプロパティが変更されているときには、その表示されているブロック名の後に接尾辞 (*) が付記されます。プロパティの編集が済んだら、「適用」ボタンをクリックしてブロックを更新します。定義したプロパティは、PDF ファイル内にブロック定義の一部として格納されます。

重なったブロック 重なりあうブロック群は選択しづらいことがあります。その領域をクリックすると必ず最前面のブロックが選ばれてしまうからです。このような場合には、コンテキストメニューの「ブロックの選択」項目を用いれば、ブロックのうちのいずれか 1 個を名前を選択することができます。1 個のブロックを選んだ直後にその領域で行う操作は、その選択した 1 個のブロックに対してのみ効力を持ち、他のブロックに対しては効力を持ちません。たとえば **Enter** を押せば、選択したブロックのプロパティを編集できます。この方法を利用すれば、ブロックの上に他のブロックが部分的ないし完全にかぶさっていても、簡単にそのブロックのプロパティを編集することができます。

ブロックプロパティの値を繰り返し使用・リセット キー入力やクリックの量をいくらか軽減できるよう、ブロックツールは、直前のブロックのプロパティダイアログで入力されたプロパティ値を記憶しています。こうした値は、新規ブロックの作成時に再利用されます。もちろんその値は、いつ別の値でオーバライドしてもかまいません。

プロパティのダイアログで「**全てリセット**」ボタンを押すと、多くのブロックプロパティがそれぞれのデフォルトにリセットされます。以下のアイテムは変更されません：

- ▶ **Name**・**Type**・**Rect**・**Description** プロパティ
- ▶ すべてのカスタムプロパティ

注 定義済みプロパティのデフォルト値は、プレビュー生成時のプレースホルダデータを保持する defaulttext・defaultimage・defaultpdf・defaultgraphics プロパティと混同しないようにする必要があります (386 ページ「**ブロックのデフォルト内容**」参照)。

複数のブロックを一度に編集 複数のブロックのプロパティを一度に編集すれば、大いに時間が節約できます。複数のブロックを選択するには以下のように操作します：

- ▶ メニュー項目「**PDFlib ブロック**」→「**PDFlib ブロックツール**」を選択してブロックツールをアクティブにします。
- ▶ 1 個目のブロックをクリックして選択します。最初に選択したこのブロックがマスターブロックです。他の 1 個ないし複数のブロックを Shift+ クリックして、選択ブロック群に加えます。あるいは、「**編集**」→「**すべてを選択**」をクリックして、現在のページ上のすべてのブロックを選択することもできます。
- ▶ これらのブロックのうちいずれか 1 個をダブルクリックすると、ブロックプロパティダイアログが開きます。この時ダブルクリックしたブロックは、新たにマスターブロックになります。
- ▶ あるいは、1 個のブロックをクリックしてマスターブロックとして指定したうえで、**Enter** キーを押してブロックプロパティダイアログを開くこともできます。

プロパティダイアログには、選択されているすべてのブロックに適用されるプロパティの部分集合だけが表示されます。ダイアログには、マスターブロックから採られたプロパティ値が表示されます。この時、選択されているすべてのブロックに対して、プロパティ群を以下のように適用することができます：

- ▶ チェックボックス「**マスターブロックのすべてのプロパティを適用**」がチェックされていない場合：「**適用**」をクリックすると、ダイアログ内で手変更を加えられたプロパティ群 (黒でハイライトされている) のみが、選択されているすべてのブロックへコピーされます。
- ▶ チェックボックス「**マスターブロックのすべてのプロパティを適用**」がチェックされている場合：「**適用**」を押すと、マスターブロックのすべてのカレントプロパティと、ダイアログ内で手変更を加えられたすべてのプロパティが、選択されているすべてのブロックへコピーされます。これを利用して、ある 1 個のブロックのブロックプロパティを、他の 1 個ないし複数のブロックへコピーすることも可能です。

以下の定義済みプロパティ、およびカスタムプロパティは共用できません。すなわち、これらは複数のブロックに対して一度に編集することはできません：

Name・Description・Subtype・Type・Rect・Status

13.3.3 ブロックフィルタリング

ページにブロックが大量にある場合、ブロック長方形が多いせいで、ひいては画面表示のパフォーマンスに悪影響が出たりして、ブロックの編集が不便になる場合があります。そ



図 13.3
ブロックフィルタ定義

んな時にはブロックフィルタリングを利用すると便利です。これは、一部のブロックだけを画面に表示させるものです。以下の絞り込み条件を、「**ブロックフィルタを定義**」ダイアログで1つないし複数指定できます (図 13.3 参照) :

- ▶ **ブロック名**: 指定したパターンに名前がマッチするブロック群を選択できます。パターン内にワイルドカード「*」(任意のキャラクタ列) と「?」(任意のキャラクタ 1 個) を入れることも可能です。
- ▶ **ブロックの種類**。
- ▶ **ブロックの最小・最大の幅と高さ**。
- ▶ **ブロックの位置**: ページを田型に分割したページクワドラントのうち左上、左下など 1 つを指定します。

フィルタを定義した後は、コンテキストメニューの「**ブロックフィルタを適用**」項目でその有効と無効を切り替えることが可能です。ブロックフィルタリングは画面表示とプレビューに対して効果があります (386 ページ「13.5 ブロックを Acrobat でプレビュー」参照)。フィルタの定義と状態 (有効か無効か) は文書ごとに独立して保持されます。ブロックツールが有効の間は、ページの左上隅に小さな「F」アイコンによってフィルタの状態が表示されています。

Acrobat が閉じられる際には、最後に使われたフィルタ定義が構成ファイルに保存され、次の起動時に復元されます。混乱を避けるために、文書が開かれた時点ではブロックフィルタリングは無効になっています。

13.3.4 ページ間・文書間でブロックをコピー

Block Plugin は、現在のページの中で、または現在の文書の中で、あるいは他の文書へと、ブロックを移動したりコピーしたりするための手段をいくつか提供しています：

- ▶ ブロックをマウスでドラッグして移動・コピー、または他のページや開いている文書へブロックを貼り付け
- ▶ ブロックを、通常のコピー/貼り付け操作を用いて、同一文書内の1個ないし複数のページへ複製
- ▶ ブロックを、新しいファイル（ページが空白の）や、既存の文書（既存のページにブロックを適用）へ書き出し
- ▶ 他の文書からブロックを取り込み

ブロックの定義を維持したままページの内容を更新したい場合には、ブロックを保ったまま背景の1個ないし複数のページを置換することができます。そのためには、「ツール」→「ページを整理」→「置換」を用います。

ブロックをコピー ブロックのコピーを作成するには、1個ないし複数のブロックを選択して、Ctrl キー（Windows の場合）か Alt キー（macOS の場合）を押しながら、新しい位置へドラッグします。この修飾キーは変更可能です（381 ページ「13.3.5 Block Plugin のユーザーインタフェースを XML でカスタマイズ」参照）。キーを押している間は、マウスカーソルが変わります。コピーされたブロックは元のブロックと同じプロパティを持ちますが、ただし名前と位置だけは自動的に変更されます。

また、コピー/貼り付けを使って、ブロックを、同一ページ内の他の場所へ、または同一文書内の他のページへ、あるいは Acrobat でその時点で開いている他の文書へコピーすることもできます：

- ▶ ブロックツールをアクティブにしてから、コピーしたいブロック群を選択します。
- ▶ Ctrl+C（Windows の場合）か Cmd+C（macOS の場合）を、または「編集」→「コピー」を使って、選択したブロックをクリップボードへコピーします。
- ▶ コピー先ページへ移動します（必要なら）。
- ▶ ブロックツールがアクティブであることを確認して、Ctrl+V（Windows の場合）か Cmd+V（macOS の場合）を、または「編集」→「貼り付け」を使って、クリップボードからブロックを現在のページと文書へ貼り付けます。

ブロックを他のページ群へ複製 1個ないし複数のブロックの複製を、現在の文書の中の任意の数のページ上に一度に作成することもできます：

- ▶ ブロックツールをアクティブにしてから、複製したいブロック群を選択します。
- ▶ 「PDFlib ブロック」メニューかコンテキストメニューから「取り込みと書き出し」→「複製...」を選びます。
- ▶ どのブロックを複製するかを選び（「選択されているブロック」または「このページ上の全ブロック」）、この選んだブロック群を複製したい複製先ページの範囲を選びます。

ブロックを書き出す・取り込む ブロックの書き出し/取り込み機能を使うと、ある1つのページ上のブロックの定義や、ある文書内のすべてのブロックの定義を、複数の PDF ファイル間で共用することが可能です。これは、既存のブロック定義を維持したままペー

ジ内容を更新したいときに便利です。ブロック定義を別ファイルとして書き出すには以下のように操作します：

- ▶ ブロックツールをアクティブにしてから、書き出したいブロック群を選択します。
- ▶ 「PDFlib ブロック」メニューかコンテキストメニューから「取り込みと書き出し」→「書き出し ...」を選択します。ページ範囲と、ブロック定義を持たせたい新規 PDF ファイル名を入力します。

ブロック定義を取り込むには「PDFlib ブロック」→「取り込みと書き出し」→「取り込み ...」を選択します。ブロック取り込みの際には、取り込んだブロックを文書内の全ページに適用するか、それともあるページ範囲にのみ適用するかを選ぶことができます。複数のページを選択した場合、ブロック定義は変更されずに各ページへコピーされます。取り込むブロック定義よりも取り込み先範囲のほうがページ数が多い場合には、「テンプレートを繰り返す」チェックボックスを用いることもできます。これをチェックすると、取り込みファイル内のブロックのシーケンスが、現在の文書の中で、文書の終わりに達するまで繰り返されます。

書き出しでブロックを他の文書へコピー ブロックを書き出す際には、そのブロック群を他の文書内のページ群へ直接適用することもできます。結果として、ある文書から別の文書へブロック群を転写することが可能です。そのためには、ブロックの書き出し先として既存の文書を選びます。「既存のブロックを削除」チェックボックスをチェックすると、書き出し先の文書にブロックが存在していてもすべて削除され、その後、新しいブロック群がその文書へコピーされます。

13.3.5 Block Plugin のユーザーインタフェースを XML でカスタマイズ

Block Plugin のユーザーインタフェースのいくつかの設定は、各 Acrobat セッションごとに保管 / 再読み込みされており、XML 構成ファイルを通じて制御することが可能です。サンプル構成ファイル *factory settings.xml* をディストリビューションに同梱しています。構成が変更されたとき、新しい設定は *user settings.xml* に格納されます。変更された構成は、Acrobat が起動されるたびに読み込まれ、Acrobat が閉じられるたびに書き込まれます。この構成ファイルは、以下のような場所に格納されています（システムディレクトリの名前はローカライズされている場合があります）：

Windows : C:\Users\\Documents\PDFlib\Block Plugin 6

macOS : /Users\/Library/Application Support/Adobe/Acrobat/DC/PDFlib/Block Plugin 6

以下の XML エレメントを用いて、構成を手変更することが可能です：

- ▶ エレメント */Block_Plugin/MainDialog/CloseOnApply* : ブロックプロパティダイアログの「適用したらダイアログを閉じる」チェックボックスの初期状態を制御します。このチェックボックスは、ブロックプロパティダイアログを、ブロックを作成した後も、またはブロックプロパティを変更した後も開いたままにしておくかどうかを決定します。
- ▶ エレメント */Block_Plugin/MainDialog/ApplyAllProps* : ブロックプロパティダイアログの「マスターブロックのすべてのプロパティを適用」チェックボックスの初期状態を制御します。このチェックボックスは、マスターブロックのすべてのプロパティが選択された複数のブロックへコピーされるか、それともダイアログ内の変更が加わったプロパティ群のみかを指定します。
- ▶ エレメント */Block_Plugin/FontDialog/ShowBaseFonts* : ブロックプロパティダイアログのフォント一覧（「書式」プロパティグループの *fontname* プロパティ）に、システム

にインストールされているフォント群に加えて、ベース 14 フォントも表示するかどうかを制御します。

- ▶ エlement `/Block_Plugin/Command/ControlByClick`: メニュー項目「PDFlib ブロック」→「オブジェクトをクリックでブロックを定義」の初期状態を制御します。
- ▶ エlement `/Block_Plugin/Command/DetectFonts` は、メニュー項目「PDFlib ブロック」→「背景フォント・色の検出」の初期状態を制御します。
- ▶ エlement `/Block_Plugin/Command/KeyAccelerator`: `control` (Windows では Ctrl キーを、macOS では Command キーを示します)・`shift` (Shift キーを示します)・`control+shift`・`none` のいずれかの値を取り、以下のキーボードショートカットに対するアクセラレータキーを指定します:

A (すべてを選択), C (コピー), I (ブロックプロパティダイアログ), V (貼り付け), X (切り取り)

変更は、Acrobat を次に起動した際に有効になります。なぜならキーボードショートカットは実行時には変更できないからです。このエントリがない場合は、アクセラレータは一切利用できません。デフォルトは `control` です。

- ▶ エlement `/Block_Plugin/Configuration/Preferences/PreviewStatusMessage` は、各プレビュー操作後にステータスメッセージダイアログ（「10 個のブロックが処理されました …」など）が表示されるかどうかを制御します。
- ▶ エlement `/Block_Plugin/Service/MoveAccelerator` は、矢印キーを用いる高速ブロック移動のための修飾キーを指定する値 `shift`・`control` を受け入れます。Element `/Block_Plugin/Service/MoveAcceleratorSize` は、修飾キーが押されている時の刻みを指定します (デフォルト: 10 ポイント)。

13.4 PDF フォームフィールドを PDFlib ブロックに変換

PDFlib ブロックを手動で作成するのではなく、PDF フォームフィールドをブロックへ自動変換させることもできます。これは、複雑な PDF フォームがあって PPS で自動流し込みさせたい場合や、既存の大量の PDF フォームを自動流し込みできるように変換したい場合などに特に便利です。1つのページ上のすべてのフォームフィールドを PDFlib ブロックに変換するには、「PDFlib ブロック」→「フォームフィールドの変換」→「現在のページ」を選択します。文書内のすべてのフォームフィールドを変換したい場合は「全ページ」を選びます。選択したフォームフィールドだけを変換するには（1個または複数のフォームフィールドを選択するには、Acrobat の「ツール」→「リッチメディア」から「オブジェクトを選択」ツールを選びます）、「選択されているフォームフィールド」を選択します。

フォームフィールド変換の詳細 自動フォームフィールド変換では、「PDFlib ブロック」→「フォームフィールドの変換」→「変換オプション ...」ダイアログで選択されている種類のフォームフィールドが、テキスト行ブロックかテキストフローブロックに変換されます。デフォルトでは、すべての種類のフォームフィールドが変換されます。変換されたフィールドの属性は、表 13.3 に従って、その照応するブロックプロパティへ変換されます。

同名の複数のフォームフィールド 同じページ上にある複数のフォームフィールドは、同じ名前を持つことが許されていますが、それに対してブロック名はページ上で一意でなければなりません。このため、フォームフィールドがブロックに変換される際には、生成されるブロックの名前に数の接尾辞が付加され、一意なブロック名が作成されます（383 ページ「フォームフィールドを照応するブロックに関連付け」も参照）。

なお、Acrobat の内部的な問題のため、複数のフォームフィールドが同じ名前を持つ場合のフィールドの属性は正しく報告されません。複数のフィールドが同じ名前を持っていて、しかし属性が異なる場合には、生成されるブロックにはこうした属性の違いは反映されません。変換処理はこの場合、警告メッセージを表示して、関係するフォームフィールド群の名前を示します。この場合は、生成されたブロックのプロパティを注意深くチェックする必要があります。

フォームフィールドを照応するブロックに関連付け 同じ名前のフィールドが複数あった場合（ラジオボタン等）、変換されたフォームフィールドの名前は変更されてしまっていますから、特定のフォームフィールドに照応するブロックを正しく同定することは困難です。このことは特に、FDF または XFDF ファイルをソースとして用いてブロックへの流し込みを行い、その結果をフォームへの記入と同じにしたい場合に問題となります。

この問題を解決するため、AcroFormConversion プラグインは、元のフォームフィールドに関する情報を、その照応するブロックを作成する際に、カスタムプロパティ群として記録します。表 13.2 に、ブロックを正しく同定するために利用できるカスタムプロパティの一覧を示します。プロパティの型はすべて文字列です。

ブロックを照応するフォームフィールドへバインド PDFlib フォームフィールドと生成 PDFlib ブロックを同期させるために、生成されたブロックを、その照応するフォームフィールドにバインドしておくことができます。言い換えれば、ブロックツールが内部的にフォームフィールドとブロックとの紐付けを保持するということです。変換処理が再実行される際、バインドされたブロックは、その照応する PDFlib フォームフィールドの属性を反映して更新されます。ブロックがバインドされていると、作業の二度手間が省け

表 13.2 ブロックに照応する元のフォームフィールドを同定するためのカスタムプロパティ

カスタムプロパティ	意味
<code>PDFlib:field:name</code>	フォームフィールドの完全修飾名。
<code>PDFlib:field:pagenumber</code>	元の文書でフォームフィールドが存在していたページの番号（文字列で）。
<code>PDFlib:field:type</code>	フォームフィールドの種類。pushbutton・checkbox・radiobutton・listbox・combobox・textfield・signature のうちのいずれか。
<code>PDFlib:field:value</code>	(type=checkbox の場合のみ) フォームフィールドの出力値。

表 13.3 PDF フォームフィールドから PDFlib ブロックへの変換

以下の PDF フォームフィールド属性は 以下の PDFlib ブロックプロパティに変換される
全フィールド	
位置	Rect
名前	Name
ツールヒント	Description
一般→一般プロパティ→表示と印刷	Status : 表示 =active 非表示 =ignore 表示 / 印刷しない =ignore 非表示 / 印刷する =active
一版→一般プロパティ→向き	orientate : 0=north、90=west、180=south、270=east
表示方法→テキスト→フォント	fontname
表示方法→テキスト→フォントサイズ	fontsize : 文字サイズ auto は、ブロックの高さの 3 分の 2 の固定文字サイズに変換され、fitmethod は auto に設定されます。複数行のブロック／フィールドにおいては、この組み合わせでは結果として自動的に適切な文字サイズになるので、ブロックの高さの 3 分の 2 という初期値よりも小さくなる場合があります。
表示方法→テキスト→テキストの色	strokecolor・fillcolor
表示方法→境界線と色→境界線の色	bordercolor
表示方法→境界線と色→塗りつぶしの色	backgroundcolor
表示方法→境界線と色→幅	linewidth : 細=1、標準=2、太=3
テキストフィールド	
オプション→整列	position : 左揃え ={left center} 中央 ={center center} 右揃え ={right center}
オプション→デフォルト	defaulttext
オプション→複数行	チェックありならテキストフローブロックを作成 チェックなしならテキスト行ブロックを作成
ラジオボタン・チェックボックス	

表 13.3 PDF フォームフィールドから PDFlib ブロックへの変換

以下の PDF フォームフィールド属性は 以下の PDFlib ブロックプロパティに変換される
「デフォルトでチェック」がオンの場合： オプション→チェックボックススタイル、 オプション→ボタンスタイル	defaulttext : チェックマーク =4 円形 =l 十字形 =8 ひし形 =u 四角形 =n 星形 =H (文字は ZapfDingbats フォントにおける各記号を表します)
リストボックス・コンボボックス	
オプション→選択されている (デフォルト) 項目	defaulttext
ボタン	
オプション→アイコンとラベル→ラベル	defaulttext

て便利です：フォームが対話的利用のために更新された時には、その照応するブロックも自動的に更新されるからです。

ブロック生成後に変換元フォームフィールドを残したくない場合は、「PDFlib ブロック」→「フォームフィールドの変換」→「変換オプション ...」ダイアログの「変換されたフォームフィールドを削除」オプションを選びます。このオプションを選ぶと、フォームフィールドは変換処理後に完全に削除されます。削除されたフィールドに関連づけられていたアクション (JavaScript など) も、すべて文書から削除されます。

バッチ変換 フォームフィールドを PDFlib ブロックに変換したい PDF 文書が多数ある場合には、バッチ変換機能を利用して、任意の数の文書を自動処理することもできます。「PDFlib ブロック」→「フォームフィールドの変換」→「バッチ変換 ...」を選択すれば、バッチ処理ダイアログが現れます：

- ▶ 入力ファイルは個別に選ぶこともできますし、1 個のフォルダの中身をすべてまとめて処理させることもできます。
- ▶ 出力ファイルは、入力ファイルと同じフォルダへ書き出すこともできますし、別のフォルダへ書き出すこともできます。出力ファイルには、入力ファイルと区別するためにプレフィックスを追加することもできます。
- ▶ 大量の文書を処理する際には、ログファイルを指定することを推奨します。変換後、ログファイルには、処理されたすべてのファイルの一覧と、それぞれの変換結果に関する詳細が書き込まれており、エラーが起きた場合にはエラーメッセージも書き込まれます。

変換処理の間、変換される PDF 文書は Acrobat で表示されますが、変換が完了するまで、Acrobat のメニュー機能やツールは使用できません。

13.5 ブロックを Acrobat でプレビュー

注 PDFlib ディストリビューションの中の `block_template.pdf` 文書で、プレビュー機能を試すことができます。必要なリソース（フォント・画像等）も PDFlib ディストリビューションに含まれています。

PDFlib ブロックは PPS によって処理されます。PPS を用いることで、ブロックへの流し込み処理について、そのデータソース（データベース内のテキスト、ディスク上の画像ファイル等）や、生成される文書の書式・対話的性質をカスタマイズすることができます。この処理について詳しくは 391 ページ「13.6 PPS でブロックへ流し込み」で解説します。

しかし Block Plugin には内蔵バージョンの PPS が含まれており、これを用いて、プログラミングを一切必要とせずに Acrobat 上で、流し込まれたブロックのプレビューバージョンを生成することができます。このプレビュー機能は、カスタムプログラミングと同等の柔軟性を提供することはできませんが、ブロックへの流し込み結果を手軽に眺めるには適しています。ブロックプレビューを活用すれば、ブロックの位置やサイズを改善したり、ブロックのプロパティ（フォント名・文字サイズ等）をチェックしたりすることができます。プレビューの表示結果に満足するまで、ブロックを変更し、プレビューを新たに生成することができます。プレビューは、現在のページについても、文書全体についても生成できます。

プレビューはつねに、新しい PDF 文書として表示されます。元の文書（ブロックを含んでいる）は、プレビューを生成しても変更を受けません。プレビュー文書は、必要に応じて保存することも捨てることもできます。

ブロックのデフォルト内容 プラグインでは、サーバサイドのデータソース（データベース等）からブロックのテキスト・画像・ベクトルグラフィック・PDF 内容を入手することは望むべくもありませんので、プレビュー機能ではつねに、ブロックのデフォルト内容が、すなわち `defaulttext`・`defaultimage`・`defaultpdf`・`defaultgraphics` プロパティで指定されている値が用いられます。通常、PPS で使われる実際のブロック内容を代表するようなサンプルデータ集合が、デフォルトデータとして用いられます。デフォルト内容を持たないブロックは、プレビュー生成時には無視されます。`Status=ignoredefault` のブロックについても同様です。

新規ブロックでは、デフォルトプロパティは空です。プレビュー機能を使う前には、「デフォルト内容」プロパティグループの `defaulttext`・`defaultimage`・`defaultpdf`・`defaultgraphics` プロパティ（ブロックの種類による）に書き込むか、あるいは「高度な PPS オプション ...」ダイアログ内の同じ名前のオプションに対して適切な値を与える必要があります。

ブロックプレビューを生成 ブロックプレビューを作成するには、以下のいずれかの方法を用います：

- ▶ PDFlib ブロックプレビューアイコン  をクリック。これは Acrobat DC で以下の場所にあります：「ツール」→「PDF を編集」をクリック。
- ▶ メニュー項目「PDFlib ブロック」→「プレビュー」→「プレビューの生成」で。
- ▶ ブロックツールがアクティブのときは、どのブロックもない所を右クリックすれば、コンテキストメニューに項目「プレビューの生成」と「プレビュー設定 ...」が現れます。

プレビューは、ディスク上の PDF ファイルに基づいて作成されます。Acrobat 上で変更を行っていた場合、ブロック PDF を「ファイル」→「上書き保存」または「ファイル」→

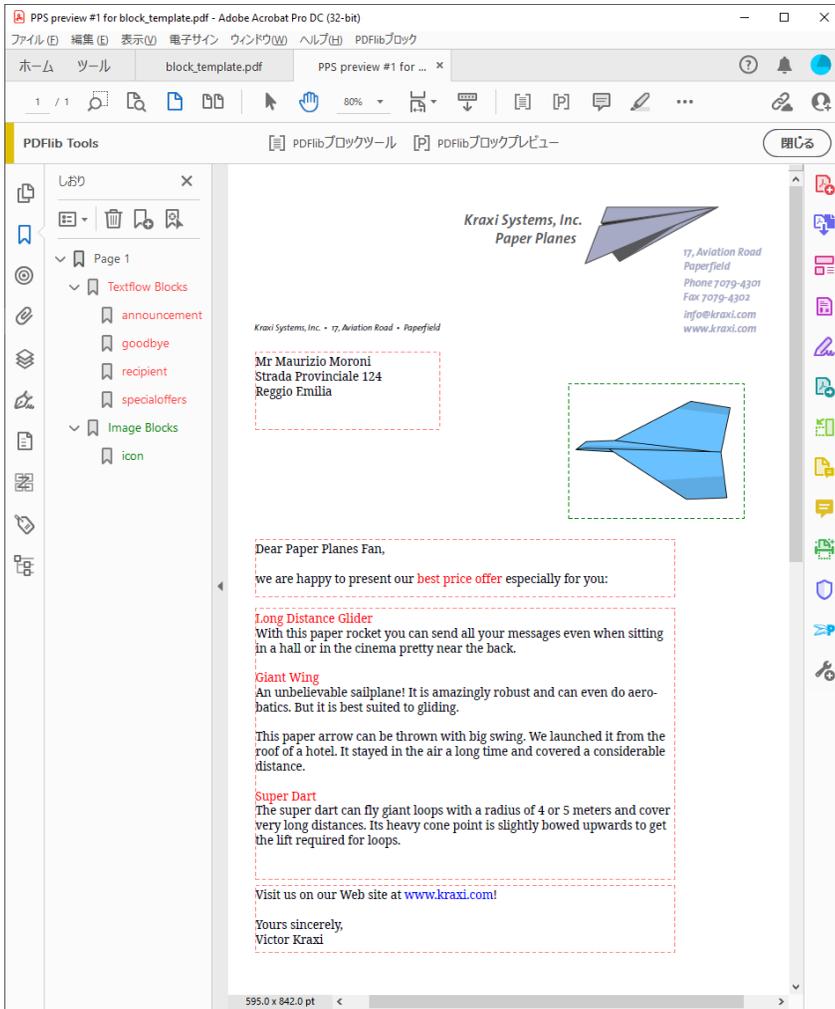


図 13.4
 図 13.1 の文書のプレビュー PDF。ブロック情報レイヤーと注釈を内容として持っています。

「名前を付けて保存 ...」を用いてディスクに保存してはじめて、その変更はプレビューに反映されます。変更を受けたブロックは、ブロックの名前の後にアスタリスクが付いていることで識別できます。プレビュー機能を構成して、プレビュー作成前にブロック PDF が自動保存されるようにすることもできます。そうすれば、対話的に行なった変更が確実に、ただちにプレビュー内で反映されます。

プレビューを構成 ブロックプレビューの作成と、その基礎をなす PPS の動作については、いくつかの性質を、「PDFlib ブロック」→「プレビュー」→「プレビュー設定 ...」で構成することができます：

- ▶ 現在のページをプレビューするか、それとも文書全体をプレビューするか。
- ▶ 生成されるプレビュー文書の出力ディレクトリ。
- ▶ ブロック PDF をプレビュー作成前に自動保存。
- ▶ ブロック情報レイヤーと注釈を追加。
- ▶ 生成される出力へブロック群をコピー。

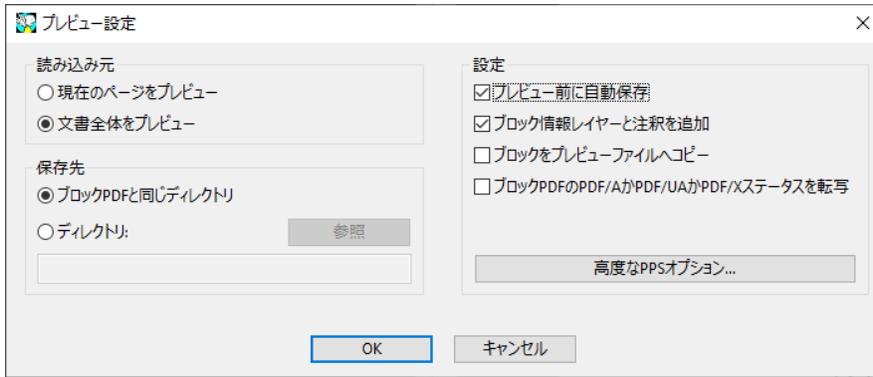


図 13.5 ブロックプレビュー構成

- ▶ 「ブロック PDF の PDF/A か PDF/UA か PDF/X ステータスを転写」: これらの規格では、レイヤーと注釈の使用は制限されていますので、「ブロック情報レイヤーと注釈を追加」オプションとこのオプションは同時には使えません。
- ▶ 「ブロックをプレビューファイルへコピー」を利用すると、流し込みの際に、PDF ブロック群を、生成されたプレビューへコピーすることができます。ブロックへの流し込みが成功したかどうかにかかわらず、すべてのブロックがコピーされます。
- ▶ 「高度な PPS オプション」ダイアログを利用して、PPS API に従って PPS 関数のオプションリストを追加指定することもできます。たとえば、`PDF_set_option()` の `searchpath` オプションを用いて、ブロックへの流し込みに使うフォントや画像が置かれているディレクトリを指定することができます。高度なオプションを指定する際には、PPS コードの開発者との連携のもとに行うことを推奨します。

ブロックの順序 文書が Acrobat の「別名で保存 ...」で保存される際には、そのブロック群は、そのブロック名に従ってアルファベット順に並べ替えられます。ブロック群がプレビューによって処理される順序も、pCOS によって報告される順序も、これと同じです。とはいえ、各種アプリケーションでは通常、ブロックへの流し込みを、その名称に基づいて行いますので（ファイル内での格納順に従ってではなく）、PDF 文書内での順序が問題となることは通常ありません。

ブロックフィルタリングとプレビュー ブロックフィルタを適用すると、そのフィルタは画面表示にもプレビューにも効果を及ぼします。よって、フィルタで除外しているブロック群は、プレビューを生成する際に処理されません。

プレビューで提供される情報 生成されるプレビュー文書には、元のページ内容（背景）と、流し込みが行われたブロックのほかに、さまざまな情報も含まれることがあります。この情報は、ブロックや PPS 構成のチェックや改善に役立つものです。デフォルト内容を持つアクティブなブロックそれぞれについて、以下のアイテムが作成されます：

- ▶ **エラーマーカ**：流し込みが成功しなかったブロックは、打ち消し線の付いた長方形として視覚化されていますので、容易に識別できます。エラーマーカは、ブロックが処理できなかったときにはつねに作成されます。
- ▶ **しおり**：ブロックの処理結果はしおりにまとめられます。このしおりは、ページ番号とブロックの種類に従って、かつエラーが起きたときはエラーにも従って、構造化さ

れています。しおりは「表示」→「表示切り替え」→「ナビゲーションパネル」→「しおり」で表示できます。しおりはつねに作成されます。

- ▶ **注釈**：処理されたブロックごとに、ブロック内容そのものに加えて、ページ上に注釈が作成されます。この注釈の長方形は、元のブロックの枠を視覚化しています（デフォルト内容と流し込みモードによっては、これはブロック内容の枠とは異なる場合があります）。この注釈の中にはブロックの名前が入っており、ブロックへの流し込みができなかったときにはエラーメッセージも入っています。注釈はデフォルトで生成されますが、プレビュー構成で無効化することもできます。PDF/A-1・PDF/X 規格では注釈の使用は制限されていますので、「**ブロック PDF の PDF/A か PDF/UA か PDF/X ステータスを転写**」オプションを有効にすると、注釈は作成されません。
- ▶ **レイヤー**：ページの内容は、分析とデバッグが容易になるよう、複数のレイヤーに分けて配置されます。ページ背景（すなわち元のページの内容）、ブロックの各種類、流し込みができなかったエラーブロック、ブロック情報を持った注釈について、それぞれ別々のレイヤーが作成されます。空のままになるレイヤーについては作成されません（エラーが何も起きなかった場合等）。レイヤー一覧は「表示」→「ナビゲーションパネル」→「レイヤー」で表示できます。デフォルトでは、ページ上のすべてのレイヤーが表示されます。いずれかのレイヤーの内容を見えなくするには、そのレイヤーの名前の左の目のアイコンをクリックします。レイヤーの作成は、プレビュー構成で無効化することも可能です。PDF/A-1・PDF/X-3 規格ではレイヤーの使用は制限されていますので、「**ブロック PDF の PDF/A か PDF/UA か PDF/X ステータスを転写**」オプションを有効にすると、レイヤーは作成されません。

PDF/A か PDF/UA か PDF/X ステータスを転写 「ブロック PDF の PDF/A か PDF/UA か PDF/X ステータスを転写」構成は、これらの規格に従った PDF 出力を作成する必要があるときに有用です。転写モードは、入力以下いずれかの規格に準拠しているときに有効にできます：

PDF/A-1a:2005, PDF/A-1b:2005
PDF/A-2a, PDF/A-2b, PDF/A-2u
PDF/A-3a, PDF/A-3b, PDF/A-3u

PDF/UA-1

PDF/X-4
PDF/X-4p
PDF/X-5n

プレビューが転写モードで作成されるときは、PPS は、ブロック PDF の以下の性質を、生成するプレビューへ複製します：

- ▶ PDF 規格識別。
- ▶ 出力インテント条件。
- ▶ ページ寸法。すべてのページ枠を含みます。
- ▶ タグ付き PDF：構造ツリーと文書言語（もしあれば）。
- ▶ XMP 文書メタデータ。

規格準拠の PDF 文書を転写する際には、すべてのブロック流し込み操作が、それぞれの規格に準拠している必要があります。たとえば、出力インテントがなければ、ICC プロファイルのない RGB 画像は使うことができません。同様に、使用しているフォントはすべて埋め込む必要があります。要請の全一覧は、329 ページ「12.3 PDF/A によるアーカイビング」と、341 ページ「12.4 PDF/X による印刷出力」に示しています。PDF/A または

PDF/X 転写モードでのブロック流し込み操作が、選択されている規格に違反するときには（デフォルト画像が RGB カラー空間を用いているにもかかわらず、文書が適切な出力インテントを含んでいない場合等）、エラーメッセージが表示され、プレビューは生成されません。これにより、ユーザーは作業フローの非常に早い時点で、規格違反の危険を感知することができます。

13.6 PPS でブロックへ流し込み

PPS でブロックへ流し込みを行うには、まずブロックを含むページを、`PDF_fit_pdi_page()` メソッドで出力ページ上に貼り付ける必要があります。ページを貼り付けた後、その上のブロックへ `PDF_fill_*block()` 関数群で流し込みを行うことができます。

簡単な例：可変テキストをテンプレートに追加 PDF テンプレートへの動的テキストの追加は、非常に頻繁に必要となる動作です。以下のコード断片は、入力 PDF 文書（テンプレート、ブロックコンテナ）の中のページを開き、それを出力ページ上に配置し、そして `firstname` というテキストブロックに可変テキストを入れ込みます：

```
doc = p.open_pdi_document(filename, "");
if (doc == -1)
    throw new Exception("エラー： " + p.get_errmsg());

page = p.open_pdi_page(doc, pageno, "");
if (page == -1)
    throw new Exception("エラー： " + p.get_errmsg());

p.begin_page_ext(width, height, "");
/* 取り込んだページを貼り付け */
p.fit_pdi_page(page, 0.0, 0.0, "");

/* 貼り付けたページ上のブロック1個へ流し込み */
p.fill_textblock(page, "firstname", "Serge", "");

p.close_pdi_page(page);
p.end_page_ext("");
p.close_pdi_document(doc);
```

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。

ブロックのプロパティをオーバーライド 場合によってプログラマーは、ブロックの定義が与えているプロパティ群を一部だけ採用し、その他のプロパティをカスタムの値でオーバーライドしたいことがあります。これはさまざまな場合に有用です：

- ▶ 業務上の要請で特定のオーバーライドが必要と判断される場合に対応。
- ▶ 画像・PDF ページの拡張倍率を、ブロック定義から採らずに、アプリケーションで算出。
- ▶ ブロックの座標をプログラムで変える。生成したい請求書のデータ項目数が一定でない場合等。
- ▶ 別々のスポットカラー名を与えることも可能。プリントサービス業務で、顧客ごとの要請に合わせるため。

プロパティをオーバーライドするには、`PDF_fill_*block()` 関数群のオプションリストに、プロパティの名前と、その照応する値を与えます。例：

```
p.fill_textblock(page, "firstname", "Serge", "fontsize=12");
```

これは、ブロックの内部の `fontsize` プロパティを、与えた値 12 でオーバーライドします。ほとんどすべてのプロパティ名を、オプションとして用いることが可能です。

プロパティのオーバーライドは、それぞれのメソッド呼び出しにのみ適用されます。ブロック定義内に保持されるわけではありません。

前のブロック群の内容に応じてブロック群を移動させる ブロックはサイズが固定なので、その内容が変わるとそぐわない場合があります。たとえば大きなブロックにわずかなテキストを流し込むと、2ブロック間にアキが生じてしまうかもしれません。テキストが多すぎる場合には、そのブロックの長方形に収まりきらないこともありえます。そのような場合には、ブロック流し込み処理の結果をクエリして、後続するブロック群の位置を調整することが可能です：

- ▶ デフォルトの *fitmethod* は *auto* ですので、テキストは、ブロックの長方形にむりやりはめ込まれます。内容が多い時にブロックからあふれてもよいことにするには、*fitmethod* を *nofit* に設定する必要があります。これを指定するには、デザイン時にブロックテンプレート内でブロックプロパティで指定するか、*PDF_fill_textblock()* に *fitmethod* オプションを与えます。
- ▶ 整形された内容に合わせて名前付き範囲枠を生成するために、*PDF_fill_*block()* にオプション *matchbox={name=...}* を与えます。この名前は何でもよいのですが、ページ上で一意である必要があります。ブロック名を範囲名へ流用するのが簡便です。
- ▶ 整形されたブロック内容がどこまで達しているかを知るために、*PDF_info_matchbox()* を用いてこの範囲枠の下辺の位置をクエリします。また、このブロックの長方形の下辺の位置を知るために、*PDF_pcos_get_number()* を用いて pCOS パス *pages[...]/blocks/<blockname>/Rect[i]* をクエリします。
- ▶ この2つの値の差をとります。このオフセットが正なら、内容がブロックを満たしきらなかったということです。負なら、内容がブロックからあふれたわけです。いずれの場合も、次のブロックを、この算出したオフセットの分だけ上か下へ動かせばよいのです。これを実現するには、*PDF_fill_*block()* の *refpoint* オプションで *Rect* プロパティをオーバーライドします。このオプションは相対座標をサポートしています。具体的には、接尾辞 *r* をつけることによって相対的オフセットを与えることができます。これを利用すれば、新しい絶対座標を計算しなくて済みます。
- ▶ この方式は、任意の数のブロックに対しても（その種別によらず）、各ブロックのオフセットを累積させることによって、適用してゆくことができます。各ブロックの内容に応じて、その次のブロックを、上か下へ、適切な量だけ動かしていきます。

クックブック 完全なコードサンプルが *starter_block* サンプルにあります。

流し込んだブロックの上に取り込んだページを配置 取り込んだページは、どのブロック流し込み関数を使うよりも前に、出力ページ上に配置しておく必要があります。ということは元ページは通常、ブロック内容よりも下に配置されることになります。しかし場合によっては、流し込みが行われたブロックよりも上に元ページを配置したいこともあるでしょう。これを実現するには、*PDF_fit_pdi_page()* の *blind* オプションを用いてページを一度貼り付けることにより、そのページ上のブロックとその位置を PPS に知らせておき、ブロックへの流し込みが済んだ後にページを再び貼り付けることにより、実際にページ内容を表示させます：

```
/* ブロックを用意するためにページをblindモードで配置してページを見えなくする */
p.fit_pdi_page(page, 0.0, 0.0, "blind");
```

```
/* ブロックへ流し込み */
p.fill_textblock(page, "firstname", "Serge", "");
/* ... いろいろなブロックへ流し込み ... */
```

```
/* ページを再度配置、今度は見えるように */
p.fit_pdi_page(page, 0.0, 0.0, "");
```

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/block_below_contents` トピックにあります。

ブロックへ流し込む際にコンテナページを無視 取り込んだブロックは、そのブロックの背景のページ内容を一切参照せずに、プレースホルダとして使ってもよいでしょう。ブロックを持つコンテナページをブラインドモードで、すなわち `PDF_fit_pdi_page()` で `blind` オプションを指定して、1 個ないし複数のページ上に貼り付けたいので、ブロックへの流し込みを行うという方法です。こうすれば、出力ページ上にコンテナページを貼り付けることなく、ブロックやそのプロパティの利点を活用することができ、また、ブロックを複数のページ上へ（または同一出力ページ上へも）複製することが可能になります。

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

テキストフローブロックを連結 テキストフローブロックは、前のブロックからあふれたテキストが次のブロックに入るよう、連結することが可能です。たとえば、長い可変テキストがあって、別のページへ続かせる必要が想定される場合、2 個のブロックを連結しておけば、1 個目のブロックがいっぱいになっても、残りは 2 個目のブロックへ流し込まれます。

PPS は、`PDF_fill_textblock()` とブロックプロパティに与えられたテキストから、1 個のテキストフローを内部的に作成します。連結されていないブロックの場合は、このテキストフローはそのブロック内に配置され、その照応するテキストフローハンドルは呼び出しが終わった時点で削除され、あふれたテキストは失われます。

連結されたテキストフローの場合は、最初のブロックへ流し込んだ後に余っているあふれテキストを、その次のブロックへ流し込むことができます。最初のテキストフローの余りがブロック内容として使われ、新たなテキストフローは作成されません。テキストフローブロックの連結は以下のように動作します：

- ▶ 連結されたテキストブロックのチェーンの中の最初の `PDF_fill_textblock()` を呼び出す時は、`textflowhandle` オプションに値 `-1` (PHP の場合：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を `PDF_fill_textblock()` が返しますので、アプリケーション側でこれを保持しておく必要があります。
- ▶ `PDF_fill_textblock()` への次の呼び出しでは、前段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（このとき `text` 引数にテキストを与えても無視されるので、空にするべきです）。ブロックへ、テキストフローの余りが流し込まれます。
- ▶ この処理を、さらなるテキストフローブロック群に対して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()` に与えれば、ブロック流し込みの結果を知ることができます。終了状況やテキストの終了位置などがわかります。

なお、`fitmethod` プロパティは `clip` に設定する必要があります（`textflowhandle` を与えているときはどのみちこれがデフォルトです）。テキストフローブロックを連結する基本的なコード断片は以下ようになります：

```
p.fit_pdi_page(page, 0.0, 0.0, "");
tf = -1;

for (i = 0; i < blockcount; i++)
{
    String optlist = "textflowhandle=" + tf;
    int reason;
    tf = p.fill_textblock(page, blocknames[i], text, optlist);
    text = null;
}
```

```

if (tf == -1)
    break;

/* いちばん最近のfit_textflow()呼び出しの結果をチェック */
reason = (int) p.info_textflow(tf, "returnreason");
result = p.get_string(reason, "");

/* テキストが全部配置されたならループを抜ける */
if (result.equals("_stop"))
{
    p.delete_textflow(tf);
    break;
}
}

```

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/linked_textblocks トピックにあります。

ブロックの流し込み順序 ブロック関数群 `PDF_fill_*block()` は、プロパティとブロック内容を、以下の順序で処理します：

- ▶ 背景: `backgroundcolor` プロパティが存在し、かつ `None` 以外のカラースペースキーワードを持っているときは、ブロック領域は指定された色で塗られます。
- ▶ 枠線: `bordercolor` プロパティが存在し、かつ `None` 以外のカラースペースキーワードを持っているときは、ブロックの枠は指定された色と線幅で描線されます。
- ▶ 内容: 与えられたブロック内容と、`bordercolor`・`linewidth` 以外のすべてのプロパティが処理されます。
- ▶ テキスト行・テキストフローブロック: テキストもデフォルトテキストも与えられていないときは、何の出力も行われません。背景色やブロックの枠線もありません。

ネストされたブロック ブロックへ流し込みを行う前には、そのブロックを含むページを出力ページ上にまず貼り付ける必要があります（そうでないと、ページを拡張・回転・平行移動した後のブロックの位置を PPS が知りえないため）。ページをブロックのコンテナとしてのみ使っており、静的内容を新ページへコピーしなくてよい場合には、取り込んだページを、`blind` オプションを用いて貼り付けることができます。

取り込んだページを、どのような方法で出力ページ上に貼り付けても、ブロックへの流し込みは行うことができます：

- ▶ ページは、`PDF_fit_pdi_page()` で直接貼り付けることができます。
- ▶ ページは、テーブルセル内に `PDF_fit_table()` で間接的に貼り付けることができます。
- ▶ ページは、他の PDF ブロックの内容として `PDF_fill_pdfblock()` で貼り付けることができます。

この 3 番目の方法、すなわち PDF ブロックへ、ブロックを含む他のページを流し込むという方法を用いると、ブロックコンテナをネストすることができます。これを活用すると、面白い使い方を簡単に実装できます。たとえば、2 段階のブロック流し込み処理で、組み付けとパーソナライゼーションの両方を実装することができます：

- ▶ 第一層のブロックコンテナページには、いくつかの大きな PDF ブロックを置きます。これらは、印刷する紙の上の主要な領域を表しています。PDF ブロックの配置は、想定している紙の後工程を反映しています（折り・断裁等）。
- ▶ この第一層の PDF ブロックそれぞれへ、第二層のコンテナ PDF ページを流し込みます。この PDF ページには、テキスト・画像・PDF・グラフィックのうちのいずれかのブロッ

クを置いておき、それらへ可変テキストを流し込んでパーソナライゼーションを行います。

この方法で、ブロックコンテナはネストすることができます。ブロックのネストは何重でも可能ですが、三重以上のネストが必要になることはまれでしょう。

この第二層のブロックコンテナ（レターのテンプレートページなど）は、各組み付けページで同じにすることもできますし、別のものにすることもできます。もし同じにした場合は、まずレターテンプレート上のブロック群への流し込みを行ってから、そのレターテンプレート自体を次の第一層ブロック内に貼り付ける必要があります。なぜなら、PPS はつねに、テンプレートページがもっとも最近に配置された位置を用いるからです。

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/nested_blocks` トピックにあります。

ブロックの座標 ブロックの長方形の座標は、PDF のデフォルト座標系を参照しています。ブロックを含んだページを PPS で出力ページに配置するときには、`PDF_fit_pdi_page()` に対していくつかの位置付け・拡張オプションを与えることができます。これらのオプションは、そのブロックが処理される際に考慮されます。これを利用すると、1つのテンプレートページを出力ページ上に何度でも配置して、そのたびにそのブロック群ヘデータを流し込むことができます。たとえば1枚の組み付け紙上に、1つの名刺テンプレートを4回配置するといったことが可能です。ブロック関数群は、座標系の変換を正しく行い、すべてのブロックに対して、それがページ上に配置されるたびに、正しくテキストを配置します。クライアントに求められるのはただ、ページを配置して、そしてその配置したページ上のすべてのブロックを処理することだけです。以後はそのページを、出力ページ上の他の場所に配置したうえで、その新しい場所に対してさらにブロック処理操作を行うことができ、これを繰り返していくことが可能です。

Block Plugin におけるブロック座標の表示のされかたは、PDF ファイル内に格納されているものとは異なっています。プラグインでは Acrobat の方式を用いて、座標の原点をページの左上隅に置いています。内部座標（ブロック内に格納されているもの）では PDF の方式を用いて、座標の原点をページの左下隅に置いているためです。プロパティダイアログの座標表示は、Acrobat で指定されている単位にも従います（374 ページ「ブロックのサイズと位置」参照）。

ブロックプロパティでスポットカラー ブロックプロパティで特色（スポットカラー）を使うには、「...」をクリックすれば、HKS・Pantone スポットカラーの全一覧を表示させることができます。これらのカラー名は PPS に内蔵されており、それ以上の準備なしに使用できます。カスタムスポットカラーに対しては、Block Plugin で代替色を定義することができます。ブロックプロパティで代替色を指定していないときは、PPS アプリケーションで `PDF_makespotcolor()` かなるべきカラーオプションリストを用いてカスタムスポットカラーをあらかじめ定義しておく必要があります。そうでないとブロックへの流し込みは失敗します。

13.7 ブロックのプロパティ

PPS と Block Plugin では、どの種類のブロックに対しても適用することのできる一般プロパティ群が用意されています。そのほかに、ブロックの種類「テキスト行」・「テキストフロー」・「画像」・「PDF」・「グラフィック」にそれぞれ特有のプロパティ群もあります。

プロパティは、ハンドルとアクションリストを除いて、オプションリストと同じデータ型に対応しています。

ブロックプロパティの名前は一般に、`PDF_fit_textline()`・`PDF_fit_image()`といった API メソッドに対するオプションと同じです (`fitmethod`・`charspacing` 等)。その場合、それぞれの動作は、対応するオプションの解説に書いてあるものとまったく同じです。

13.7.1 管理プロパティ

管理プロパティ群は、すべての種類のブロックに適用されます。必須エントリ群は Block Plugin によって自動生成されます。表 13.4 に、管理プロパティの一覧を示します。

表 13.4 管理プロパティ

キーワード	とりうる値・解説
Description	(文字列) ブロックの機能に関する、人が読める説明。エンコーディングは PDFDocEncoding か Unicode (後者の場合は先頭 BOM)。このプロパティは、ユーザーへの情報提供のためだけにあり、PPS には無視されます。
Locked	(論理値) true なら、ブロックとそのプロパティは Block Plugin で編集できません。このプロパティは PPS には無視されます。デフォルト : false
Name	(文字列、必須) ブロックの名前。ブロック名は、ページごとに一意である必要がありますが、文書内では一意でなくてもかまいません。3 種のキャラクタ [] / は、ブロック名には使えません。ブロック名は最長 125 文字です。
Subtype	(キーワード、必須) ブロックの種類によって、Text・Image・PDF・Graphics のいずれか。テキスト行ブロックとテキストフローブロックは、ともに Subtype が Text になることに留意してください。両者は textflow プロパティによって識別されます。
textflow	(論理値) 一行処理か複数行処理かを制御します。このプロパティは、Block Plugin のユーザーインタフェースでは表示されず、それぞれテキスト行ブロックとテキストフローブロックへマップされます (デフォルト : false) : false テキスト行ブロック : テキストが一行で生まれ、 <code>PDF_fit_textline()</code> で処理されます。 true テキストフローブロック : テキストが複数行にわたる場合があり、 <code>PDF_fit_textflow()</code> で処理されます。標準テキストプロパティ群に加え、テキストフロー関連のプロパティ群も指定できます (表 13.9 参照)。
Type	(キーワード、必須) つねに Block

13.7.2 長方形プロパティ

長方形プロパティ群は、すべての種類のブロックに適用されます。これらは、ブロックの長方形本体の書式を記述します。必須エントリ群は Block Plugin によって自動生成されま
す。表 13.5 に、長方形プロパティの一覧を示します。

表 13.5 長方形プロパティ

キーワード	とりうる値・解説
background-color	(色) このプロパティが存在し、かつ None 以外の色空間キーワードを持つならば、長方形が描かれ、与えられた色で塗られます。既存のページ内容をおおい隠したいときに有用です。デフォルト : None
bordercolor	(色) このプロパティが存在し、かつ None 以外の色空間キーワードを持つならば、長方形が描かれ、与えられた色で描線されます。デフォルト : None
linewidth	(float。正の値でなければならない) ブロックの長方形を描くのに用いる線の描線幅。bordercolor 設定時のみ有効です。デフォルト : 1
Rect	(長方形、必須) ブロックの座標。座標原点はページ左下隅。ただし Block Plugin では、座標は Acrobat の方式で、すなわちページ左上隅を原点として表されます。座標の単位は、Acrobat でその時点で選択されている単位で表示されますが、PDF ファイル内部ではつねにポイント単位で格納されています。
Status	(キーワード) PPS とブロック機能がブロックを処理する方法を記述します (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	defaulttext/image/pdf/graphics プロパティ・オプションが無視される、すなわち可変内容がないとき (特にプレビューで) ブロックが空のままになる点を除き、active と同じです。これは特に、ブロックがプレビュー生成のためのデフォルト内容を持っているかもしれないけれども、サーバサイドでブロックへ流し込みが行われる際にはそのブロックのデフォルト内容が用いられないようにしたいときに有用です。また、ブロックをプレビューする際にも、ブロックプロパティからデフォルト内容を削除することなくデフォルト内容を無効化するために用いることができます。
static	可変内容が配置されません。ブロックにデフォルトのテキスト・画像・PDF・グラフィック内容があれば、それが用いられます。

13.7.3 書式プロパティ

書式プロパティ群は、組版の詳細を指定します：

- ▶ 表 13.6 に、透過書式プロパティの一覧を示します。これらは、すべての種類のブロックに適用されます。
- ▶ グラフィックステートプロパティ群 (PDFlib API リファレンス内の `PDF_create_gstate()` のオプション群を参照)。
- ▶ 表 13.7 に、テキスト書式プロパティの一覧を示します。これらは、テキスト行ブロックとテキストフローブロックに適用されます。

表 13.6 すべてのブロック種別に対する透過書式プロパティ

キーワード	とりうる値・解説
<i>blendmode</i>	(キーワードリスト。PDF/A-1 モードで用いられるときは値 <code>Normal</code> を持つ必要があります) ブレンドモードの名前 : <code>None</code> · <code>Color</code> · <code>ColorDodge</code> · <code>ColorBurn</code> · <code>Darken</code> · <code>Difference</code> · <code>Exclusion</code> · <code>HardLight</code> · <code>Hue</code> · <code>Lighten</code> · <code>Luminosity</code> · <code>Multiply</code> · <code>None</code> · <code>Normal</code> · <code>Overlay</code> · <code>Saturation</code> · <code>Screen</code> · <code>SoftLight</code> 。デフォルト : <code>None</code>
<i>opacityfill</i>	(float。PDF/A モードで用いられるときは値 <code>1</code> を持つ必要があります) 塗り操作の不透明度を範囲 <code>0</code> ~ <code>1</code> で表したものの。値 <code>0</code> は完全透過を意味し、 <code>1</code> は完全不透過を意味します。
<i>opacitystroke</i>	(float。PDF/A モードで用いられるときは値 <code>1</code> を持つ必要があります) 描線操作の不透明度を範囲 <code>0</code> ~ <code>1</code> で表したものの。値 <code>0</code> は完全透過を意味し、 <code>1</code> は完全不透過を意味します。

表 13.7 テキスト行・テキストフローブロックに対するテキスト書式プロパティ

キーワード	とりうる値・解説		
<i>charspacing</i>	(float またはパーセント値) 文字間隔。パーセント値は <i>fontsize</i> に対する割合。デフォルト : 0		
<i>decoration-above</i>	(論理値) true の場合、 <i>underline</i> ・ <i>strikeout</i> ・ <i>overline</i> オプションで有効にされたテキスト装飾がテキストの前面に描かれ、そうでなければテキストの背面に描かれます。描画順序を変えると、装飾線の書式に影響を与えます。デフォルト : false		
<i>fillcolor</i>	(色) テキストの塗り色。デフォルト : gray 0 (= 黒)		
<i>fontname</i> ¹	(文字列) フォントの名前。 <i>PDF_load_font()</i> が求めるものと同じです。Block Plugin では、システムで利用可能なフォントの一覧が表示されます。ただしこうしたフォント名は、macOS・Windows・Unix システム間で互換とは限りません。 <i>fontname</i> の先頭が「@」キャラクタである場合は、そのフォントは縦書きモードで適用されます。		
<i>fontsize</i> ¹	(float) 文字のサイズをポイント単位で指定します		
<i>horizscaling</i>	(float またはパーセント値) テキストの横伸縮。デフォルト : 100%		
<i>italicangle</i>	(float) テキストの斜体角度を度単位で表したものです。デフォルト : 0		
<i>kerning</i>	(論理値) カーニングの動作。デフォルト : false		
<i>overline</i>	(論理値) 上線のモード。デフォルト : false		
<i>shadow</i>	(複合) 影付き効果を生み出します (デフォルト : 影なし)。以下のサブプロパティが使用可能です : <i>fillcolor</i> (色) 影の色。デフォルト : {gray 0.8} <i>offset</i> (float 2 個かパーセント値 2 個のリスト) テキストの参照点からの影のオフセットを、ユーザー座標で、または文字サイズに対するパーセント値で表したものです。デフォルト : {5% -5%}		
<i>strikeout</i>	(論理値) 取り消し線のモード。デフォルト : false		
<i>strokecolor</i>	(色) テキストの描線色。デフォルト : gray 0 (= 黒)		
<i>strokewidth</i>	(float・パーセント値・キーワードのいずれか。 <i>textrendering</i> が袋文字に設定されているときのみ意味を持ちます) 袋文字の線幅 (ユーザー座標で、または <i>percentage</i> に対するパーセント値で)。キーワード <i>auto</i> 、またはそれと等価な値 0 は、内蔵のデフォルトを用います。デフォルト : <i>auto</i>		
<i>textrendering</i>	(整数) テキスト表現モード。Type 3 フォントでは値 3 のみ意味を持ちます (デフォルト : 0) : <table border="0" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 50%; vertical-align: top;"> <p>0  テキストを塗る</p> <p>1  テキストを描線 (袋文字)</p> <p>2  テキストを塗って描線</p> <p>3 不可視テキスト</p> </td> <td style="width: 50%; vertical-align: top;"> <p>4  テキストを塗り、クリッピングパスに追加</p> <p>5  テキストを描線し、クリッピングパスに追加</p> <p>6  テキストを塗って描線し、クリッピングパスに追加</p> <p>7  テキストをクリッピングパスに追加 (ブロックでは不可)</p> </td> </tr> </table>	<p>0  テキストを塗る</p> <p>1  テキストを描線 (袋文字)</p> <p>2  テキストを塗って描線</p> <p>3 不可視テキスト</p>	<p>4  テキストを塗り、クリッピングパスに追加</p> <p>5  テキストを描線し、クリッピングパスに追加</p> <p>6  テキストを塗って描線し、クリッピングパスに追加</p> <p>7  テキストをクリッピングパスに追加 (ブロックでは不可)</p>
<p>0  テキストを塗る</p> <p>1  テキストを描線 (袋文字)</p> <p>2  テキストを塗って描線</p> <p>3 不可視テキスト</p>	<p>4  テキストを塗り、クリッピングパスに追加</p> <p>5  テキストを描線し、クリッピングパスに追加</p> <p>6  テキストを塗って描線し、クリッピングパスに追加</p> <p>7  テキストをクリッピングパスに追加 (ブロックでは不可)</p>		
<i>textrise</i>	(float またはパーセント値) テキストの縦方向のオフセット。パーセント値は <i>fontsize</i> に対する割合。デフォルト : 0		
<i>underline</i>	(論理値) 下線のモード。デフォルト : false		
<i>underline-position</i>	(float・パーセント値・キーワードのいずれか) 下線テキストのベースラインに対する描線の相対位置。パーセント値は <i>fontsize</i> に対する割合。デフォルト : <i>auto</i>		

表 13.7 テキスト行・テキストフローブロックに対するテキスト書式プロパティ

キーワード **とりうる値・解説**

underline-width (float・パーセント値・キーワードのいずれか) 下線テキストの線幅。パーセント値は `font-size` に対する割合。デフォルト : `auto`

wordspacing (float またはパーセント値) 単語間隔。パーセント値は `font-size` に対する割合。デフォルト : `0`

1. このプロパティは、テキスト行・テキストフローブロックでは必須です。Block Plugin はこれを自動生成します。

13.7.4 テキスト作成プロパティ

テキスト作成プロパティ群は、テキスト行・テキストフローブロックの前処理工程を指定します。表 13.8 に、テキスト行・テキストフローブロックに適用されるテキスト作成プロパティの一覧を示します。

表 13.8 テキスト行・テキストフローブロックに対するテキスト作成プロパティ

キーワード	とりうる値・解説
<i>charref</i>	(論理値) true なら、数値参照・文字参照・グリフ名参照の置き換えを行います。デフォルト：グローバルな <i>charref</i> オプション
<i>escape-sequence</i>	(論理値) true なら、内容文字列・ハイパーテキスト文字列・名前文字列内のエスケープシーケンスの置き換えを行います。デフォルト：グローバルな <i>escapesequences</i> オプション
<i>features</i>	(キーワードのリスト) <i>script</i> ・ <i>language</i> オプションに従って、OpenType フォントのどのタイプグラフィ機能をテキストに適用するかを指定します。フォントにない機能は無視されます： <i>_none</i> フォント内のどの機能も適用しません。 <名前> 4文字のOpenType名を与えてその機能を有効にします。よく用いられる機能名は <i>aalt</i> ・ <i>liga</i> ・ <i>ital</i> ・ <i>tnum</i> ・ <i>smcp</i> ・ <i>swsh</i> ・ <i>zero</i> です。利用できる機能の名前と説明の一覧を、167ページ「7.3 OpenTypeレイアウト機能」に示しています。 <i>no<名前></i> 機能名の前に接尾辞 <i>no</i> をつけると (<i>noliga</i> 等) その機能が無効化されます。 以下の機能はデフォルトで有効化されます： <i>calt</i> ・ <i>ccmp</i> ・ <i>clig</i> ・ <i>liga</i> ・ <i>locl</i> 。 <i>vert2</i> ・ <i>vert</i> 機能は、縦書きモードのフォントに対しては自動的に有効化されます。
<i>language</i>	(キーワード。 <i>script</i> が与えられているときのみ意味を持ちます) 指定された言語に従ってテキストが処理されます。これは <i>features</i> ・ <i>shaping</i> オプションに対して意味を持ちます。キーワードの全一覧は 176 ページ「7.4.2 用字系と言語」に示しています。例：ARA (アラビア語)・JAN (日本語)・HIN (ヒンディー語)。デフォルト： <i>_none</i> (言語未定義)
<i>script</i>	(キーワード。 <i>shaping=true</i> のときは必須) 指定された用字系に従ってテキストが処理されます。これは <i>features</i> ・ <i>shaping</i> ・ <i>advancedlinebreak</i> オプションに対して意味を持ちます (デフォルト： <i>_none</i>)： <i>_auto</i> (シェーピングに対してのみ意味を持ちます) 大多数のテキストが属する用字系を選択。ただし <i>latn</i> ・ <i>_none</i> は無視されます。 <i>_none</i> 未定義の用字系。 以下の用字系を指定すると複雑用字系シェーピングが可能です： <i>arab</i> ・ <i>hebr</i> 中東用字系群 <i>sinh</i> ・ <i>tibt</i> 中央アジア用字系群 <i>khmr</i> ・ <i>thai</i> 東南アジア用字系群 <i>hang</i> ・ <i>hani</i> 東アジア用字系群 <i>beng</i> ・ <i>deva</i> ・ <i>gujr</i> ・ <i>knda</i> ・ <i>mlym</i> ・ <i>orya</i> ・ <i>guru</i> ・ <i>taml</i> ・ <i>telu</i> インド用字系群 このほか、OpenType 仕様に従った <i>latn</i> ・ <i>cyrl</i> ・ <i>grek</i> など用字系タグを用いて OpenType 機能処理を制御できます。全キーワード一覧が 176 ページ「7.5.2 シェーピング処理を制御」にあります。デフォルト： <i>_none</i>
<i>shaping</i>	(論理値) true なら、 <i>script</i> ・ <i>language</i> オプションに従った複雑用字系シェーピングと双方向再配列がテキストに適用されます。 <i>script</i> オプションが <i>_none</i> 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。シェーピングは、テキストフロー内の右書きテキストには使えません (テキスト行内のみ)。デフォルト： <i>false</i>

13.7.5 テキスト組版プロパティ

表 13.9 に、テキストフローブロックに対してのみ用いることができるテキスト組版プロパティの一覧を示します。ただし *stamp* プロパティだけは、テキスト行ブロックに対しても用いることができます。これらは、テキストフローを処理するための初期オプションリストを構築するために用いられます (*PDF_create_textflow()* の *optlist* 引数に照応しています)。テキストフローで用いるインラインオプションリストは、プラグインでは指定することができず、サーバ上で *PDF_fill_textblock()* によるブロックへの流し込みの際に、またはブロックの *defaulttext* プロパティ内で、テキスト内容の一部として与えることができます。

表 13.9 テキスト組版プロパティ (主にテキストフローブロック用)

キーワード	とりうる値・解説
<i>adjust-method</i>	(キーワード) <i>minspacing</i> ・ <i>maxspacing</i> オプションで指定された制限内で単語間隔を詰めたり拡げたりしてもテキストの一部が行内に収まらない時、行の調整に用いる方式 (デフォルト: <i>auto</i>): <i>auto</i> 次の方式を順に適用します: <i>shrink</i> ・ <i>spread</i> ・ <i>nofit</i> ・ <i>split</i> . <i>clip</i> はめ込み枠の右端 (<i>rightindent</i> オプションを考慮) からはみ出した部分を切り落とす点を除いて、 <i>nofit</i> と同じ。 <i>nofit</i> 最後の単語を次行へ送ります。ただし、残される (短い) 行が、 <i>nofitlimit</i> オプションで指定されたパーセント値よりも短くならない場合に限りです。均等配置の段落であっても、若干がたつて見えることがあります。 <i>shrink</i> 単語が行内に収まらないとき、テキストを <i>shrinklimit</i> の制限内で圧縮します。それでも収まらなければ <i>nofit</i> 方式を適用します。 <i>split</i> 最後の単語を次行へ送らず、強制的にハイフネーションします。テキストフォントならハイフンキャラクタを挿入しますが、記号フォントなら挿入しません。 <i>spread</i> 最後の単語を次行へ送り、残された (短い) 行を均等配置するよう単語内の文字間の間隔を <i>spreadlimit</i> の制限内で拡げます。それで均等配置できなければ <i>nofit</i> 方式を適用します。
<i>advanced-linebreak</i>	(論理値) 複雑用字系で必要とされる高度な改行アルゴリズムを適用します。これはタイ語等、単語間の区切りを空白キャラクタを用いて示さない用字系での改行に必須です。locale・script オプションに従います。デフォルト: <i>false</i>
<i>alignment</i>	(キーワード) 段落内の行の書式を指定。デフォルト: <i>left</i> . <i>left</i> 左揃え (<i>leftindent</i> を始点として)。 <i>center</i> 中央揃え (<i>leftindent</i> から <i>rightindent</i> までの間で)。 <i>right</i> 右揃え (<i>rightindent</i> を終点として)。 <i>justify</i> 両端揃え。
<i>avoid-emptybegin</i>	(論理値) <i>true</i> なら、はめ込み枠先頭の空行が削除されます。デフォルト: <i>false</i>
<i>fixedleading</i>	(論理値) <i>true</i> なら、各行内で最初に見つかった行送り値を用います。そうでないなら、行内のすべての行送り値のうちの最大値を用います。デフォルト: <i>false</i>
<i>hortab-method</i>	(キーワード) テキスト内の水平タブの扱い。算出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: <i>relative</i>): <i>relative</i> <i>hortabsize</i> で指定された分、位置を進めます。 <i>typewriter</i> <i>hortabsize</i> の次の倍数まで位置を進めます。 <i>ruler</i> <i>ruler</i> オプション内の <i>n</i> 番目のタブ値まで位置を進めます。ここで <i>n</i> は、その行内でそれまでに見つかったタブの数です。n がタブ位置の数を超える分については、 <i>relative</i> 方式を適用します。

表 13.9 テキスト組版プロパティ (主にテキストフローブロック用)

キーワード	とりうる値・解説
horthabsize	(float またはパーセント値) 水平タブの幅 ¹ 。その解釈は horthabmethod オプションに依存します。デフォルト : 7.5%
lastalignment	(キーワード) 段落内の最終行の書式。alignment オプションのすべてのキーワードを用いることができるほか、以下のキーワードも用いることができます (デフォルト : auto) : auto alignment オプションの値を用います。ただしそれが justify のときは left を用います。
leading	(float またはパーセント値) テキストのベースライン間の間隔。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 : 100%
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 1 個ないし複数の構成要素から成り、オプション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。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。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。例 : DE (ドイツ)・CH (スイス)・GB (イギリス)。 ロケールを指定することは、高度な改行のために必須ではありません : キーワード <code>_none</code> は、ロケール独自の処理が行われないことを指定します。デフォルト : <code>_none</code> 。 例 : <code>de_DE</code> ・ <code>en_US</code> ・ <code>en_GB</code>
maxspacing minspacing	(float またはパーセント値) 単語間の最大間隔・最小間隔 (ユーザー座標で表すか、空白キャラクタの幅に対するパーセント値で指定)。算出される単語間隔が、与えられた値までに制限されます (ただし、wordspacing オプションはなお加算されます)。デフォルト : <code>minspacing=50%</code> 、 <code>maxspacing=500%</code>
minlinecount	(整数) はめ込み枠の最終段落の最小行数。これより行が少ないときは、次のはめ込み枠内に配置されます。値 2 を用いると、段落の中の 1 行だけがはめ込み枠末尾に配置される (「オーファン」) を避けられます。デフォルト : 1
nofitlimit	(float またはパーセント値) nofit 方式における行の長さの下限 (ユーザー座標で表すか、はめ込み枠の幅に対するパーセント値で指定)。デフォルト : 75%
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 。leftindent にこの値が加算されます。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 leftindent	(float またはパーセント値) 全テキスト行の右インデント・左インデント ¹ 。leftindent が行内で指定された場合、決定された位置がカレントテキスト位置より左のときは、このオプションはカレント行については無視されます。デフォルト : 0
ruler²	(float のリスト、またはパーセント値のリスト) horthabmethod=ruler に対する絶対タブ位置のリスト ¹ 。このリストは、非負エントリを昇順で最大 32 個持てます。デフォルト : horthabsize の整数倍
shrinklimit	(パーセント値) shrink 方式におけるテキスト長体の下限。算出される縮小率が、与えられた値までに制限されます。ただし、horizscaling オプションの値が乗算されます。デフォルト : 85%
spreadlimit	(float またはパーセント値) spread 方式における 2 文字間の間隔の上限 (ユーザー座標で表すか、文字サイズに対するパーセント値で指定)。算出された文字間隔が、charspacing オプションの値に加算されます。デフォルト : 0

表 13.9 テキスト組版プロパティ（主にテキストフローブロック用）

キーワード	とりうる値・解説
<i>stamp</i>	（キーワード。テキスト行・テキストフローブロック）このオプションを使うと、ブロック長方形内の対角線上にスタンプを作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拡大されて印字されます。スタンプのテキストを枠内に配置する際には、 <code>position</code> ・ <code>fitmethod</code> ・ <code>orientate</code> （ <code>north</code> ・ <code>south</code> のみ）オプションに従います。デフォルト： <code>none</code> 。 <i>llzur</i> 左下隅から右上隅へ向かう対角線上にスタンプが配置されます。 <i>ulzlr</i> 左上隅から右下隅へ向かう対角線上にスタンプが配置されます。 <i>none</i> スタンプは作成されません。
<i>tabalignchar</i>	（整数）タブの小数点揃えの整列位置にしたいキャラクタの Unicode 値。デフォルト：キャラクタ「。」（U+002E）
<i>tabalignment</i> ²	（キーワードのリスト）タブ位置の整列方式。このリスト内の各エントリはそれぞれ、 <code>ruler</code> オプション内で、その照応するエントリの整列方式を定義します（デフォルト： <code>left</code> ）： <i>center</i> テキストはタブ位置で中央揃えされます。 <i>decimal</i> 最初に現れる <i>tabalignchar</i> をタブ位置で左揃えされます。 <i>tabalignchar</i> が見つからないときは右揃えが適用されます。 <i>left</i> テキストはタブ位置で左揃えされます。 <i>right</i> テキストはタブ位置で右揃えされます。

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. タブ設定は、ブロックプロパティダイアログの「テキスト組版」グループの「hortabmethod=ruler におけるルーラタブ」サブグループで編集することができます。

13.7.6 オブジェクトはめ込みプロパティ

はめ込みプロパティ群は、すべての種類のブロックで利用できますが、いくつかのプロパティは、特定の種類のブロックでのみ利用できます。これらは、ブロック内に内容が配置される方法を制御します：

- ▶ 表 13.10 に、テキスト行・画像・PDF・グラフィックブロックで利用できるはめ込みプロパティの一覧を示します。
- ▶ 表 13.11 に、テキストフローブロックで利用できるはめ込みプロパティの一覧を示します（主に縦方向のはめ込みに関するものです）。

オブジェクトはめ込みアルゴリズムは、ブロック長方形をはめ込み枠として用います。*fitmethod=clip* の場合を除き、切り落としは行われません。ブロック内容がブロック長方形からはみ出さないようにしたいときは、*fitmethod=nofit* を避けてください。

表 13.10 テキスト行・画像・PDF・グラフィックブロックに対するはめ込みプロパティ

キーワード	とりうる値・解説
<i>alignchar</i>	(Unichar またはキーワード。テキスト行ブロックにのみ適用) 指定されたキャラクタをテキスト内に見つけたとき、その左下隅を、ブロック長方形の左下隅に整列させます。横書きテキストで orientate=north か south の場合には、position オプションの 1 番目の値で位置を決定します。縦書きテキストで orientate=west か east の場合には、position オプションの 2 番目の値で位置を決定します。指定された位置整列キャラクタがテキスト内にないときは、このオプションは無視されます。値 0 かキーワード none ならば、位置整列キャラクタを無効にします。fitmethod は指定されれば適用しますが、alignchar で強制的に位置付けられるので、はめ込み枠内にテキスト配置することはできません。デフォルト : none
<i>dpi</i>	(float のリスト。画像ブロックにのみ適用) 縦方向・横方向の画像解像度を表す 1 個または 2 個の値。単位は pixels per inch。値 0 の場合、画像内部に解像度が格納されていればそれを用い、なければ 72 dpi とします。fitmethod プロパティが指定されていて、そのキーワードが auto・meet・slice・entire のいずれかならば、本プロパティは無視されます。デフォルト : 0
<i>fitmethod</i>	(キーワード) 与えられた内容がブロック長方形に収まりきらないときの解決方式 : auto・clip・entire・meet・nofit・slice。(デフォルト : meet)。
<i>margin</i>	(float のリスト。テキスト行ブロックにのみ適用) テキスト枠の縦・横方向の長さ差し引きを記述する 1 個または 2 個の値。デフォルト : 0
<i>minfontsize</i>	(float またはパーセント値。テキスト行にのみ適用) fitmethod=auto で shrinklimit を超えたとき、テキストがブロック長方形に収まるよう縮小される際に、許される最小の文字サイズ。下限値を、ユーザー座標で、またはブロックの高さに対するパーセント値で指定します。下限に達したときは、テキストは指定された minfontsize を文字サイズとして作成されます。デフォルト : 0.1%
<i>orientate</i>	(キーワード) 内容を配置する向きを指定します。とりうる値は north・east・south・west。デフォルト : north
<i>position</i>	(float のリスト) 内容の中における参照点の位置を指定する、1 個または 2 個の値。ブロック内でのパーセント値として位置を指定します。テキスト行ブロックのみ : 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（欧文等）には left を意味します。 デフォルト : [0 0]、すなわち左下隅
<i>rotate</i>	(float) 回転角を度単位で表したものの。処理が始まる前にブロックが反時計回りに回転されます。参照点が回転の中心となります。デフォルト : 0

表 13.10 テキスト行・画像・PDF・グラフィックブロックに対するはめ込みプロパティ

キーワード	とりうる値・解説
<i>scale</i>	(float のリスト。画像・PDF・グラフィックブロックのみ) 縦方向・横方向の、求める伸縮倍率を表す 1 個または 2 個の値。fitmethod プロパティが指定されていて、かつそのキーワードが auto・meet・slice・entire のいずれかならば、本プロパティは無視されます。デフォルト: 1
<i>shrinklimit</i>	(float またはパーセント値。テキスト行ブロックにのみ適用) fitmethod=auto でテキストを収める際に適用される縮小倍率の下限。デフォルト: 0.75

表 13.11 テキストフローブロックに対するはめ込みプロパティ

キーワード	とりうる値・解説
<i>firstlinedist</i>	(float・パーセント値・キーワードのいずれか) ブロック長方形上端とテキスト先頭行ベースラインとの間隔を、ユーザー座標で表すか、その文字サイズ (fixedleading=true なら行の先頭の文字サイズ、そうでないなら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表すか、キーワードで表したもの (デフォルト: leading)。 <i>leading</i> 先頭行について決定された行送り値。A のような、読み分け記号付きの文字は普通、はめ込み枠上端に接するでしょう。 <i>ascender</i> 先頭行について決定されたアセンダ値。d や h のような、大きなアセンダを持つ文字は普通、はめ込み枠上端に接するでしょう。 <i>capheight</i> 先頭行について決定されたキャップハイト値。H のような大文字は普通、はめ込み枠上端に接するでしょう。 <i>xheight</i> 先頭行について決定された x ハイト値。x のような小文字は普通、はめ込み枠上端に接するでしょう。 fixedleading=false なら、先頭行内で見出されたすべての leading・ascender・xheight・capheight 値のうちの最大値が用いられます。
<i>fitmethod</i>	(キーワード) ブロックがテキストフローに対して小さすぎるときの解決方式: <i>auto</i> テキストが収まるまで、fontsize と leading を縮めます。 <i>clip</i> テキストをブロックの端で切り落としします (テキストフローブロックを連結する場合に有用です)。 <i>nofit</i> テキストをブロック下端からはみ出させます (ブロックを移動させる場合に有用です)。 デフォルト: textflowhandle オプションが与えられているなら clip、そうでないなら auto
<i>lastlinedist</i>	(float・パーセント値・キーワードのいずれか。fitmethod=nofit の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との間隔を、ユーザー座標で表すか、文字サイズ (fixedleading=true なら行の先頭の文字サイズ、そうでないなら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表すか、キーワードで表したもの。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用い、ディセンダは普通、はめ込み枠の下へはみ出すでしょう。 <i>descender</i> 最終行について決定されたディセンダ値。g や j のような、ディセンダを持つ文字は普通、はめ込み枠下端に接するでしょう。 fixedleading=false なら、最終行内で見出されたすべてのディセンダ値のうちの最大値が用いられます。
<i>linespread-limit</i>	(float またはパーセント値。verticalalign=justify の場合のみ) 上下合わせの場合に行送りを増やす際の最大値を、ユーザー座標で表すか、行送りに対するパーセント値で表したもの。デフォルト: 200%
<i>maxlines</i>	(整数またはキーワード) はめ込み枠内の最大行数。あるいはキーワード auto を指定して、できるだけ多くの行をはめ込み枠内に入れさせることもできます。最大行数が入ったとき、PDF_fit_textflow() は文字列 _boxfull を返します。

表 13.11 テキストフローブロックに対するはめ込みプロパティ

キーワード	とりうる値・解説
<i>minfontsize</i>	(float またはパーセント値) 特に fitmethod=auto のとき、はめ込み枠に収まるようテキストが縮小される際に許される最小文字サイズ。下限値を、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。下限に達してもなおテキストが収まりきらないときは、文字列 <code>_boxfull</code> が返されます。デフォルト : 0.1%
<i>orientate</i>	(キーワード) テキストを配置する向きを指定します。とりうる値は north・east・south・west。デフォルト : north
<i>rotate</i>	(float) 回転角を度単位で表したもの。はめ込み枠の左下隅を中心として、座標系を回転させます。これによって、枠とテキストが回転されます。テキストが配置された時点で回転はリセットされます。デフォルト : 0
<i>verticalalign</i>	(キーワード) はめ込み枠内のテキストの縦揃え (デフォルト : top) :
<i>top</i>	先頭行から下へ順に組版。テキストがはめ込み枠に満たないときは、テキストの下に余白があきます。
<i>center</i>	はめ込み枠内の縦方向の中央にテキストを配置。テキストがはめ込み枠に満たないときは、テキストの上下に余白があきます。
<i>bottom</i>	最終行から上へ順に組版。テキストがはめ込み枠に満たないときは、テキストの上に余白があきます。
<i>justify</i>	はめ込み枠の上端と下端にテキストを合わせます。それを実現するために、行送りを増やします。ただし、linespreadlimit で指定された限界までしか増やしません。先頭行の高さは、firstlinedist=leading の場合のみ増やします。

13.7.7 タグ付き PDF プロパティ

タグ付き PDF プロパティ群は、すべての種類のブロックで利用できます。表 13.12 に、タグ付き PDF プロパティの一覧を示します。

表 13.12 タグ付き PDF プロパティ (PPS インタフェース 11)

キーワード	とりうる値・解説
<i>tag</i>	(オプションリスト) 表 14.2 に従った構造エレメントタグ付けのためのオプション群。構造エレメント名がプロパティとして与えられている場合には、テキストフロー内のインラインタグは無視されます。

13.7.8 デフォルト内容のためのプロパティ

デフォルト内容に関するプロパティ群は、内容が特に与えられなかったときのブロックへの流し込みの方法を指定します。これらはとりわけプレビュー機能で有用です。なぜならプレビューではブロックにへそのデフォルト内容を流し込むからです。表 13.13 に、デフォルト内容に関するプロパティの一覧を示します。

表 13.13 デフォルト内容のためのプロパティ

キーワード	とりうる値・解説
<i>default-graphics</i>	(文字列。グラフィックブロックにのみ適用) グラフィックがクライアントアプリケーションから与えられなかったときに用いられるグラフィックファイルのパス名。 ¹
<i>defaultimage</i>	(文字列。画像ブロックにのみ適用) 置き換え画像がクライアントアプリケーションから与えられなかったときに用いられる画像のパス名。 ¹
<i>defaultpdf</i>	(文字列。PDF ブロックにのみ適用) 置き換え PDF がクライアントアプリケーションから与えられなかったときに用いられる PDF 文書のパス名。 ¹
<i>default-pdfpage</i>	(整数。PDF ブロックにのみ適用) デフォルト PDF 文書内のページのページ番号。デフォルト : 1
<i>defaulttext</i>	(文字列。テキスト行・テキストフローブロックにのみ適用) 可変テキストがクライアントアプリケーションから与えられなかったときに用いられるテキスト ²

1. ファイル名には絶対パスを付けず、SearchPath 機能を利用するよう、PPS クライアントアプリケーションを作っておくほうがよいでしょう。そうすればブロック処理を、プラットフォームやファイルシステムの細かい違いから切り離すことができます。
2. テキストは winansi エンコーディングか Unicode で解釈されます。

13.7.9 カスタムプロパティ

カスタムプロパティは、すべての種類のブロックに適用されます。PPS とプレビュー機能からは無視されます。表 13.14 に、カスタムプロパティの命名規則を示します。

表 13.14 すべてのブロック種別に対するカスタムブロックプロパティ

キーワード	とりうる値・解説
3 種類のキャラクタ [] / を含まないあらゆる名前	(文字列・名前・float のいずれか、または float のリスト) 各カスタムプロパティの値をどう解釈するかは、全くクライアントアプリケーションの領分です。PPS からは無視されます。

13.8 pCOS でブロック名とプロパティをクエリ

PPSによる自動ブロック処理に加えて、内蔵のpCOS機能を使うと、ブロック名を評価したり、標準・カスタムプロパティをクエリしたりすることができます。

クックブック 取り込んだPDFの中に含まれているブロックのプロパティをクエリするための完全なコードサンプルがクックブックの `blocks/query_block_properties` トピックにあります。

ブロックの数と名前を知る クライアントコード側では、取り込んだページ上のブロックの名前も数も知らなくてかまいません。なぜならクエリすることもできるからです。以下のステートメントは、ページ番号 *pagenum* のページ上のブロックの数を返します：

```
blockcount = (int) p.pcos_get_number(doc, "length:pages[" + pagenum + "]/blocks");
```

以下のステートメントは、ページ *pagenum* 上の *blocknum* 番目のブロックの名前を返します（ブロックとページの番号は0から始まります）：

```
blockname = p.pcos_get_string(doc,
    "pages[" + pagenum + "]/blocks[" + blocknum + "]/Name");
```

返されたブロック名はその後、ブロックのプロパティをクエリしたり、ブロックヘテキスト・画像・PDF・グラフィック内容を流し込んだりするために利用することができます。指定されたブロックが存在しないときは、例外が発生します。これを避けるには、*length* 接頭辞を用いて、ブロックの数を知り、ひいては *blocks* 配列の最大添字を知ることができます（配列の添字が0から始まるため、ブロックの数は最大可能添字より1大きいことに留意してください）。

ブロックが存在するかどうかをチェック クライアントアプリケーションコードにさらに柔軟性を加えるために、ブロックに流し込みを行う前に、そのブロックが存在するかどうかをチェックすることもできます。こうしておけば、デザイナーが別のページへブロックを移動させても、そのブロックへ流し込みを行うアプリケーションを破壊せずにすみません。

以下のコードは、*foo* という名前のブロックがページ上に存在するかどうかをチェックします：

```
/* pCOSオブジェクト種別「辞書」はそのブロックが存在することを意味します */
if (pcos_get_string(doc, "type:pages[" + pagenum + "]/blocks/" + "foo").equals("dict"))
{
    /* ブロック「foo」はそのページ上に存在 */
}
```

ブロックを番号か名前で特定

ブロックプロパティを特定するパス文法において、以下の表現は等価です。ここで、番号6のブロックが、その *Name* プロパティを *foo* に設定されているとします：

```
pages[...]/blocks[6]
pages[...]/blocks/foo
```

ブロックの座標をクエリ 名前 *foo* のブロックの左下隅と右上隅を記述する2個の座標ペア (*llx*, *lly*) および (*urx*, *ury*) は、以下のようにクエリできます：

```
llx = p.pcos_get_number(doc, "pages[" + pagenum + "]/blocks/foo/rect[0]");
lly = p.pcos_get_number(doc, "pages[" + pagenum + "]/blocks/foo/rect[1]");
```

```
urx = p.pcos_get_number(doc, "pages[" + pagenum + "]/blocks/foo/rect[2]");
ury = p.pcos_get_number(doc, "pages[" + pagenum + "]/blocks/foo/rect[3]");
```

上記の座標はデフォルト座標系で与えられていることに注意してください（左下隅が原点。ただし、そのページの CropBox によって変更されている可能性もあります）。一方 Block Plugin は、ページ左上隅に原点を持つ Acrobat のユーザーインタフェース座標系に従って座標を表示します。

pCOS 擬似オブジェクト **rect**（すべて小文字）を用いてクエリされる値は、関連するいかなる CropBox/MediaBox・Rotate エントリをも考慮に入れ、かつ座標の順序を正規化します。これに対し、ネイティブ PDF キー **Rect** を用いてクエリされる値は、CropBox が存在する場合には新しい座標としてそのまま受け渡すことはできません。

topdown オプションはブロック座標をクエリする際には考慮されないことに留意してください。

カスタムプロパティをクエリ カスタムプロパティは、以下の例のようにクエリすることができます。ここでは、ページ *pagenum* 上の *b1* というブロックからプロパティ *zipcode* をクエリしています：

```
zip = p.pcos_get_string(doc, "pages[" + pagenum + "]/blocks/b1/Custom/zipcode");
```

ブロック内に具体的に何というカスタムプロパティがあるかわからなければ、実行時にその名前を得ることもできます。*b1* というブロックの最初のカスタムプロパティの名前を得るには、以下のようにします：

```
propname = p.pcos_get_string(doc, "pages[" + pagenum + "]/blocks/b1/Custom[0].key");
```

番号を 0 のかわりに 1 つずつ増やしていけば、すべてのカスタムプロパティの名前を得ることができます。**length** 接頭辞を用いれば、カスタムプロパティの数を知ることができます。

存在しないブロックプロパティとデフォルト値 ブロックまたはプロパティが実在するかどうかを知るには、**type** 接頭辞を用います。パスに対する型が 0 か **null** ならば、そのオブジェクトは PDF 文書内に存在していません。なお、定義済みプロパティの場合、これはプロパティのデフォルト値が用いられることを意味します。

カスタムプロパティの名前空間 ささまざまなソースからの PDF 文書をやり取りする際に混乱が生じることを避けるため、カスタムプロパティ名をつけるときには必ず、インターネットドメイン名を企業固有の接頭辞として使い、その後にはコロン「:」とプロパティ名本体を続けることを推奨します。たとえば、ACME 社であれば以下のようなプロパティ名を使用するのです：

```
acme.com:digits
acme.com:refnumber
```

標準プロパティとカスタムプロパティはブロック内で異なる格納のされ方をしているので、標準 PPS プロパティ名（396 ページ「13.7 ブロックのプロパティ」で定義されているもの）がカスタムプロパティ名と衝突することは決してありません。

13.9 ブロックをプログラマ的に作成・取り込む

13.9.1 POCA で PDFlib ブロックを作成

PDFlib ブロックは、PPS に内蔵されている POCA インタフェースでプログラマ的に作成することも可能です。POCA を用いると、ブロックのために必要な PDF データ構造を作成したうえで、`PDF_begin/end_page_ext()` の `blocks` オプションに与えることができます。ブロック定義を作成する際には、413 ページ「13.10 PDFlib ブロックの仕様」の要請に従う必要があります。ブロックプロパティは、396 ページ「13.7 ブロックのプロパティ」に挙げたデータ型に従って作成する必要があります。

クックブック PDFlib ブロックを PPS で作成するためのコードサンプルが PDFlib クックブックの `blocks` カテゴリにあります。

PDFlib ブロックの仕様には、1 つのブロックの名前が 2 回記録されているという残念な冗長性があります：ページのメイン `Blocks` 辞書内に 1 回と、特定のブロック辞書内の `Name` エントリ内にもう 1 回です。この 2 個の名前は、そのブロックに PPS で流し込みを行う際や、そのブロックを Block Plugin でプレビューする際に問題が起こることを避けるために、同一でなければなりません。`PDF_begin/end_page_ext()` はそのため、`blocks` オプションで与えられた辞書がこの「同一ブロック名」規則に違反するブロック定義を含んでいる場合には、例外を発生させます。以下のコードサンプルでは、その照応するペアを青色で示しています。

以下のコード断片では、413 ページ「ブロック辞書のキー」で示すブロック定義を POCA 関数群を用いて作成する様子を演示しています：

```
/* ブロック辞書を作成 */
blockdict = p.poca_new("containertype=dict usage=blocks");

/* -----
 * テキストブロックを作成
 * -----
 */
textblock = p.poca_new("containertype=dict usage=blocks type=name key=Type value=Block");

container1 = p.poca_new("containertype=array usage=blocks " +
    "type=integer values={70 640 300 700}");

p.poca_insert(textblock, "type=array key=Rect value=" + container1);
p.poca_insert(textblock, "type=name key=Name value=job_title");
p.poca_insert(textblock, "type=name key=Subtype value=Text");
p.poca_insert(textblock, "type=name key=fitmethod value=auto");
p.poca_insert(textblock, "type=name key=fontname value=Helvetica");
p.poca_insert(textblock, "type=float key=fontsize value=12");

/* このブロックをページのブロック辞書内に挿入 */
p.poca_insert(blockdict, "type=dict key=job_title direct=false value=" + textblock);

/* -----
 * 画像ブロックを作成
 * -----
 */
imageblock = p.poca_new("containertype=dict usage=blocks " +
    "type=name key=Type value=Block");
```

```

container2 = p.poca_new("containertype=array usage=blocks " +
    "type=integer values={70 440 300 600}");

p.poca_insert(imageblock, "type=array key=Rect value=" + container2);
p.poca_insert(imageblock, "type=name key=Name value=logo");
p.poca_insert(imageblock, "type=name key=Subtype value=Image");
p.poca_insert(imageblock, "type=name key=fitmethod value=auto");

/* このブロックをページのブロック辞書内に挿入 */
p.poca_insert(blockdict, "type=dict key=logo direct=false value=" + imageblock);

/* -----
 * PDFブロックを作成
 * -----
 */
pdfblock = p.poca_new("containertype=dict usage=blocks " +
    "type=name key=Type value=Block");

container3 = p.poca_new("containertype=array usage=blocks " +
    "type=integer values={70 240 300 400}");

p.poca_insert(pdfblock, "type=array key=Rect value=" + container3);
p.poca_insert(pdfblock, "type=name key=Name value=pdflogo");
p.poca_insert(pdfblock, "type=name key=Subtype value=PDF");
p.poca_insert(pdfblock, "type=name key=fitmethod value=meet");

/* このブロックをページのブロック辞書内に挿入 */
p.poca_insert(blockdict, "type=dict key=pdflogo direct=false " + "value=" + pdfblock);

/* -----
 * このブロック辞書をカレントページ内に挿入
 * -----
 */
p.end_page_ext("blocks=" + blockdict);

/* クリーンナップ */
p.poca_delete(blockdict, "recursive");

```

13.9.2 PDFlib ブロックを取り込む

入力文書から1個ないし複数のPDFlibブロックを、*PDF_process_pdi()*と*action=copyallblocks*か*action=copyblock*で以下のようにカレント出力ページへコピーすることも可能です：

```

if (p.process_pdi(p, doc, 0, "action=copyallblocks block={pagenumber=1}") != 1)
{
    /* エラー */
}

```

このようにすると、マルチレベルのブロック流し込みワークフローを実装することができます。ただし、各ページ上でブロック名は一意でなければならないことに留意してください。すなわち、同じ名前を持つ複数のブロックを同じページ上へ取り込むことはできません。ブロックをコピー時に名称変更するには*outputblockname*サブオプションを用います。

13.10 PDFlib ブロックの仕様

ブロックの文法は、PDF ページを構成するデータ構造にアプリケーションが独自データを追加格納できるようにする拡張のしくみを定めた PDF Reference に準拠しています。ここでは PDFlib ブロックの文法を説明します。Block Plugin か PDFlib でブロックを作成するユーザーにはこの情報は必要ではありません。

PDFlib ブロックの PDF オブジェクト構造 ページ辞書は *PieceInfo* エントリを含んでおり、このエントリは値として別の辞書を持っています。ページ辞書は、ブロック構造の作成または最終更新のタイムスタンプを内容とするキー *LastModified* をも含む必要があります。この辞書はキー *PDFlib* を含んでおり、このキーはアプリケーションデータ辞書を値として持っています。このアプリケーションデータ辞書は、表 13.15 に挙げる 2 個の標準キーを含んでいます。

表 13.15 PDFlib アプリケーションデータ辞書内のエントリ

キー	値
<i>LastModified</i>	(日付文字列、必須) ページ上のブロックが作成された、または最近更新された日時。このエントリは、POCA インタフェースでブロックを作成した場合には PDFlib が作成します。
<i>Private</i>	(辞書、必須) ブロックリスト (表 13.16 参照)

ブロックリストとは、ブロック処理に関する一般的な情報に加えて、ページ上のすべてのブロックの一覧をも含んだ辞書です。表 13.16 に、ブロックリスト辞書の中のキーの一覧を示します。

表 13.16 ブロックリスト辞書内のエントリ

キー	値
<i>Blocks</i>	(辞書、必須) キーはそれぞれ、ブロック名の名前オブジェクト。その照応する値は、そのブロックに対するブロック辞書です (表 13.18 参照)。ブロック辞書内の Name キーは、この辞書内のブロック名と同じでなければなりません。
<i>BlockProducer</i> ¹	(文字列) ブロックをプログラマ的に作成するのに用いられたソフトウェアの名前。このエントリは、POCA インタフェースでブロックを作成した際に PDFlib によって作成されます。
<i>PluginVersion</i> ¹	(文字列) ブロックの作成に使われた Block Plugin のバージョン識別を表す文字列。
<i>pdfmark</i> ¹	(論理値) ブロックリストが pdfmark を用いて生成されている場合は true でなければなりません。
<i>Version</i>	(数、必須) ファイルが準拠するブロック仕様のバージョン番号。本文書では、バージョン 11 のブロック仕様を解説しています。

1. キー *BlockProducer*・*PluginVersion*・*pdfmark* のうちのいずれか 1 つが、かつ 1 つのみが存在する必要があります。

ブロックプロパティのデータ型 プロパティはオプションリストと同じデータ型に対応しています。ただしハンドルと、アクションリストのような特化されたリストには対応していません。表 13.17 に、これらの型が PDF データ型へどのようにマップされているかを示します。

ブロック辞書のキー ブロック辞書は、表 13.18 に挙げるキーを含むことができます。

表 13.17 ブロックプロパティに対するデータ型

データ型	PDF での型および注釈
論理値	(論理値)
文字列	(文字列)
キーワード (名前)	(名前) そのプロパティが対応するキーワードのリストにないキーワードを与えるとエラー。
<i>float</i> ・整数	(数値) オプションリストでは点もカンマも小数点として対応しているのに対し、PDF の数値では点が必要とします。
パーセント 値	(要素 2 個の配列) 配列の 1 番目の要素は数、2 番目の要素はパーセントキャラクタを持った文字列。
リスト	(配列)
色	(要素 2 個か 3 個の配列) 配列の 1 番目の要素は色空間を指定し、2 番目の要素はカラー値を指定。色なしを指定するには、各プロパティを省略する必要があります。配列の 1 番目の要素に対しては以下のエントリを指定できます： <i>/DeviceGray</i> 2 番目の要素はグレー値 1 個。 <i>/DeviceRGB</i> 2 番目の要素は RGB 値 3 個の配列。 <i>/DeviceCMYK</i> 2 番目の要素は CMYK 値 4 個の配列。 <i>[/Separation/ スポットカラー名]</i> 1 番目の要素は、キーワード Separation とスポットカラー名 1 個を持った配列。2 番目の要素は濃度値。 オプションとして 3 番目の要素で、スポットカラーの代替色を指定します。代替色はそれ自身が 1 個の色配列であり、DeviceGray・DeviceRGB・DeviceCMYK・Lab のいずれかの色空間で表されます。代替色を指定しないときは、スポットカラー名は、PPS が内部的に知っている色か、またはアプリケーションで動作時に定義しておいた色でなければなりません。 <i>[/Lab]</i> 1 番目の要素はキーワード Lab を持った配列。2 番目の要素は Lab 値 3 個の配列。
<i>unichar</i>	(テキスト文字列) UTF-16 BOM U+FEFF で始まる utf16be 形式の Unicode 文字列

表 13.18 ブロック辞書内のエントリ

プロパティグループ	値
管理プロパティ群	(いくつかのキーは必須) 表 13.4 に従った管理プロパティ群
長方形プロパティ群	(いくつかのキーは必須) 表 13.5 に従った長方形プロパティ群
書式プロパティ群	(いくつかのキーは必須) 表 13.6 に従った、すべての種類のブロックに適用される書式プロパティ群と、表 13.7 に従った、テキスト行・テキストフローブロックに適用されるテキスト書式プロパティ群
テキスト作成プロパティ群	(オプション) 表 13.8 に従った、テキスト行・テキストフローブロックに適用されるテキスト作成プロパティ群
テキスト組版プロパティ群	(オプション) 表 13.9 に従った、テキスト行・テキストフローブロックに適用されるテキスト組版プロパティ群
オブジェクトはめ込みプロパティ群	(オプション) 表 13.10 に従った、テキスト行・画像・PDF・グラフィックブロックに適用されるオブジェクトはめ込みプロパティ群と、表 13.11 に従った、テキストフローブロックに適用されるはめ込みプロパティ群

表 13.18 ブロック辞書内のエントリ

プロパティグループ	値
デフォルト内容に関するプロパティ群	(オプション) 表 13.13 に従った、デフォルト内容に関するプロパティ群
カスタム	(辞書、オプション) 表 13.14 に従った、カスタムプロパティに対するキー / 値ペアを含んだ辞書

