13 PPS と PDFlib Block Plugin

PDFlib Personalization Server（PPS）は、可変データを処理する、テンプレートを用いるPDF ワークフローをサポートするものです。ブロックという概念を用い、外部の情報源から取得した、任意の量の、1 行ないし複数行のテキスト、または画像・PDF ページ・ベクトルグラフィックを、取り込んだページに入れ込むことができます。これを利用して、PDF文書のカスタマイズを要するアプリケーションを容易に実装が可能です。たとえば：

- メールを連結
- ダイレクトメールの宛名印刷
- 納品書・明細等を発行
- 名刺の項目内容を各人ごとに変更

ブロックは、PDFlib Block Plugin を使って対話的に作成・編集もできますし、フォームフィールド変換プラグインを使って既存の PDF フォームフィールドを PDFlib ブロックに変換もできます。PPS を使ってブロックに流し込みができます。Block Plugin は内蔵版 PPS を含んでいますので、PPS でブロックに流し込んだ結果を Acrobat でプレビューできます。

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、PDFlib や PDFlib+PDI のライセンスキーだけではご利用になれません。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 が必要となります。**

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 が動作する Acrobat のバージョンは以下のとおりです（Adobe Reader では動作しません）：

- Windows：Acrobat 8/9/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ー）

Windows 版 Acrobat DC には 32 ビット版と 64 ビット版がありますので、インストーラーが 2 種類あります。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ー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ーのサブディレクトリーに入れる必要があります。プラグインのインストーラーはこれを自動的に実行しますが、手作業も可能です。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 32 ビット Acrobat の場合はプラグインフォルダーはたいてい以下のとおりです：

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

64 ビット Acrobat の場合はプラグインフォルダーはたいてい以下のとおりです：

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

macOS で 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操作を行います：

- 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。プラグインファイル群の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**フォルダ内の以下のパスへ複製（*Plug-ins* フォルダがまだない場合には作成）：

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

あるいは以下の操作を行うと、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（*Plug-ins* フォルダがまだない場合には作成）：

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

多言語インターフェイス　PDFlib Block Plugin は、ユーザーインターフェイス内で複数言語に対応しています。Block Plugin は、Acrobat のアプリケーション言語に従ってインターフェイス言語を自動的に選択します。目下、日本語・英語・ドイツ語のインターフェイスが利用可能です。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 は英語インターフェイスを使用します。

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**という新たなセキュリティー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があります。保護ビューはデフォルトでは、Acrobat の *AppData* ディレクトリーと、temp ディレクトリーなどいくつかのディレクトリーへのアクセスは許しますが、任意のユーザーディレクトリーへのアクセスは許しません。Block Pluginが読み書きできるディレクトリーは、保護ビューのデフォルトディレクトリー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ー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ー名を追加することもできます：

```
; Protected Path Section
FILES_ALLOW_ANY = C:\Users\<username>\*.*
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ー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には、以下を確認してください：

- **「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 がこの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ーモデルと衝突するからです。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、取り込んだページに、可変のテキスト・画像・PDF ページ・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、後でサーバー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、可変な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・画像拡縮といった関連特性も指定します。レイアウトを PDF 文書として作成し、その後にデザイナーが、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など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（内容が名前なのか ZIP コードなのかとか、ページ上の具体的な位置や書式）については何も知る必要がないので、レイアウトがどう変更されても影響されずにいられます。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 の側で処理されます。

言いかえれば、プログラマーが書くコードが「データ非依存」になります。すなわち汎用であり、ブロックのいかなる特性にも依存しないのです。たとえば手紙の宛先を印字するブロックがあったとして、かりにデザイナーがそれをページ上で別の所へ動かそうが、文字サイズを変えようが、汎用であるブロック処理コードに変更を加える必要はありません。デザイナーがこのブロックにラストネームでなくファーストネームを印字したいと思ったら、ブロックのプロパティを Acrobat プラグインで変更するだけで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、開発とテストのサイクルを高速化できます。ブロックのプレビューには、ブロックの定義で指定しているデフォルトデータ（文字列や画像ファイル名）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、名前や種類、その他自由なプロパティを割り当てることができます。これらプロパティは後で 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ー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、ラスター画像を持ちます。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コンテンツを持ちます。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定義済みプロパティを持っています。プロパティは、Block Plugin で作成・変更できます（378 ページ「13.3.2 ブロックプロパティを編集」を参照）。定義済み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ならテキストのフォントやサイズを指定できますし、画像ブロックや PDF ブロックなら拡縮倍率や回転を指定できます。PPS はブロックの種類ごとに、それを処理する専用の関数を提供しています（*PDF_fill_textblock()* 等）。これら関数は、取り込まれた PDF ページにおいてブロックを名前で検索し、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（1 行テキスト・複数行テキスト・ラスター画像・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。プログラマーがブロックのプロパティをオーバーライドすることも可能です。そのためにはブロック流し込み関数でそれに対応するオプションを指定します。

デフォルト内容に関するプロパティ　ブロックのデフォルト内容を持たせる特殊なブロックプロパティを指定することも可能です。ブロック流し込み関数に可変データが与えられなかったときや、ブロック内容が次回の印刷実行時には変わるかもしれないが現時点では固定なときにそのブロックに配置されるテキスト・画像・PDF・グラフィック内容を指定できます。

この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　定義済みのブロックプロパティを利用すれば可変データ処理アプリケーションを手軽に実装できますが、これらのプロパティについて PPS は内部的に知っていて自動的に処理できるわけですが、ブロックにもっと高い柔軟性を与えるためにデザイナーがカスタムのプロパティを割り当てることも可能です。カスタムプロパティを利用してブロックの概念を拡張することで、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られるでしょう。

カスタムプロパティには何のルールもありません。PPS はカスタムプロパティを何ら処理しないからです。PPS はクライアントがカスタムプロパティを利用できるようにするだけです。クライアントのコードでカスタムプロパティを取得してよしなに処理できます。レイアウトやデータ抽出についてブロックのカスタムプロパティを見て決めるようなアプリケーションが可能です。科学的なアプリケーションでは数値出力の桁数を指定するカスタムプロパティを設けてもよいでしょうし、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にしてそこからブロック用データを取得してもよいでしょう。

13.2.3 なぜ PDF のフォームフィールドを使わないの

経験ある Acrobat ユーザーなら疑問に思うでしょう。なぜ我々は新たにブロックという概念を導入したのか、なぜ 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と。根本的な違いは、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を両方とも要するアプリケーションの場合には、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）。

両者は似た点も多いですが、PDFlib ブロックには、PDF フォームフィールドに比べて、表 13.1 に示す利点があります。

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などテキスト属性を変更可能	―	○
書き込んだ内容が PDF のページ記述に取り込まれる	―	○
追加フィールドの内容をユーザーが編集可能	○	×
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・GIF・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
流し込みの際にグラフィックやテキストのプロパティをオーバーライド可能	―	○
透過内容	―	○
テキストブロックを連結可能	―	○

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 Block Plugin は、Acrobat におけるフォームツールに似ています。ブロックツールをアクティブにするとページ上のブロックが全部表示されます。Acrobat の他のツールを選択す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操作のどちらかを行います：

- ブロックアイコン  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」→「*PDFlib* ブロックツール」**。

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、十字ポインターをドラッグすれば、ページ上の希望の位置に希望の大きさのブロックを作れます。ブロックはつねに長方形で、かつ辺がページの辺と平行になります（***rotate*** プロパティを用いるとブロックの内容をページの辺に平行でなくすることができます）。ブロックの長方形をドラッグし終えると、**「*PDFlib* ブロックプロパティ」**ダイアログが現れ、ブロックの各種プロパティを編集できます（378 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生成しますが、この名前はプロパティダイアログで変更可能です。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、別のページで同じ名前を使うのはかまいません。

ダイアログの一番上で、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）から選べます。ブロックの種類ごとに色が変えてあります（図 13.1 参照）。タブは、その時選んでいるブロック種別のものだけがアクティブになります。**「*PDFlib* ブロックプロパティ」**ダイアログは、ブロックの種類によって、プロパティをグループやサブグループにまとめて階層的に表示します。

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

ブロックを選択　コピー・移動・削除・プロパティ編集などのブロック操作は、選択した 1 個ないし複数のブロックに対して働きます。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とおり操作します：

- ブロック 1 個を選択するには単にそれをシングルクリック。
- 複数のブロックを選択したい場合は、2 個目以降のブロックを Shift キーを押しながら選択。
- 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」**。

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コンテキストメニューを開くと、ブロックに関するさまざまな機能（**「*PDFlib* ブロック」**メニューから利用できる各種機能と同じ）をさっと実行できます。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 ではマウスの右ボタンでクリック、macOS では Ctrl+ クリックします。たとえばブロックを削除するには、ブロックツールで選択し

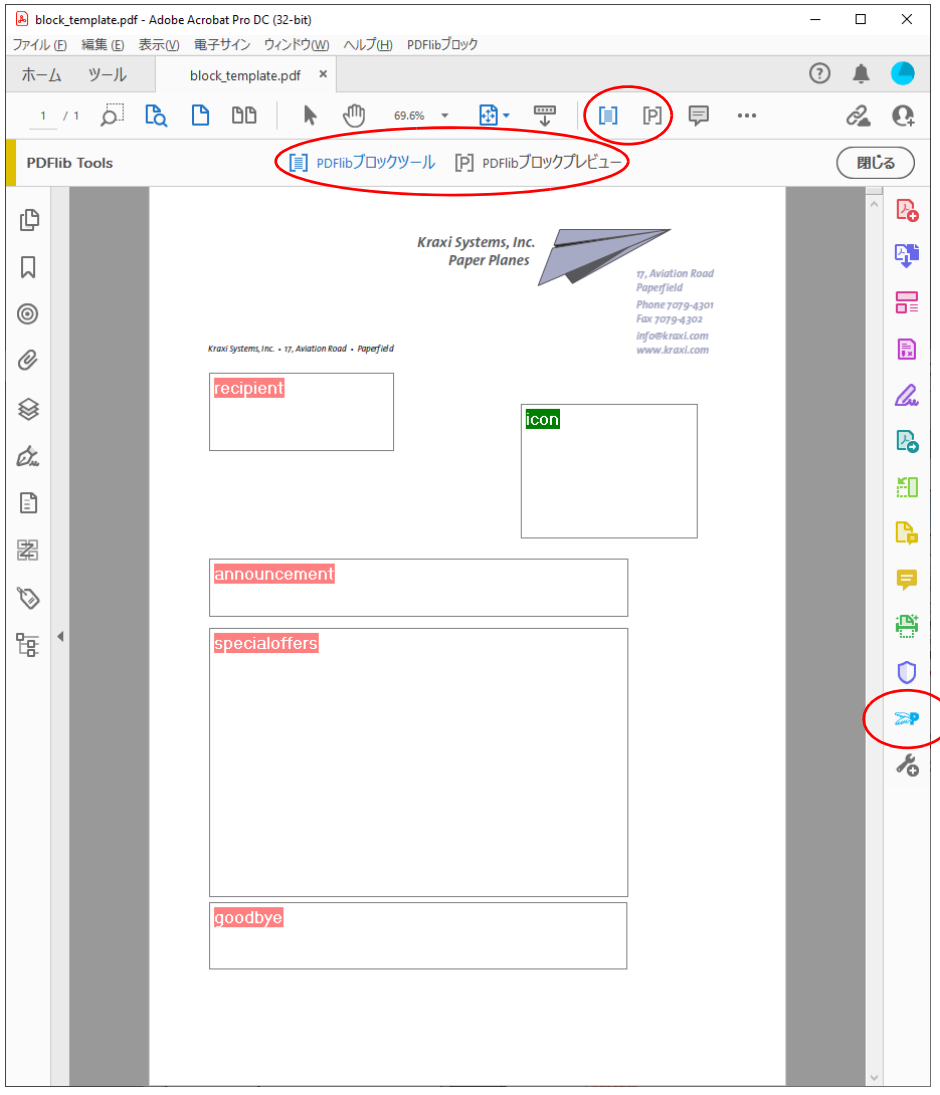

図 13.1
各種ブロックの視覚化

てから *Delete* キーを押してもいいのですが、コンテキストメニューで「**編集**」→「**削除**」という方法もあるのです。

ページ上でブロックのない所を右クリック（macOS では Ctrl+ クリック）した場合のコンテキストメニューには、ブロックプレビューを作成する項目と、ブロック機能を構成する項目があります。

ブロックのサイズと位置　ブロックツールを使って選択した 1 個ないし複数のブロックを別の位置へ動かすこともできます。Shift キーを押しながらブロックをドラッグすると真横か縦へだけ動きます。ブロックをきっちり揃えるのに便利でしょう。ポインターをブロックの角付近へ持って行くと、ポインター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。

複数のブロックの位置やサイズを揃えたいときは、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**

図 13.2
ブロックプロパティダイアログ

イズ」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動かすこともできます。

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という方法もあります。この座標系の原点はページの左上隅です。座標の表示は、その時点で Acrobat で選んでいる単位によります：

- Acrobat DC で表示単位を変更するには、**「編集」→「環境設定」**（→**「一般 ...」**）→**「単位とガイド」→「ページと定規の単位」**でインチ・センチメートル・パイカ・ポイント・ミリのいずれかを選択。
- 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」**。

ただしここで選んだ単位は、*Rect* プロパティにのみ効力を持ち、他の数値プロパティ（*fontsize* 等）には効力を持ちません。

グリッドでブロックの位置合わせ　Acrobat のグリッド機能を活用すると、ブロックの位置合わせやサイズ変更が正確にできます：

- 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化：**「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」**でグリッドの線の間隔や位置や色を変更できます。

「グリッドにスナップ」を有効にしていると、構成したグリッドにブロックのサイズと位置が揃います。この**「グリッドにスナップ」**は、新規作成する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・サイズ変更する際にも効きます。

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　ブロックの長方形を手作業でドラッグせずに、既存のページ内容からブロックのサイズを定義することも可能です。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」**を有効化します。そのうえでブロックツールを使ってページ上の画像をクリックすると、その画像と同じ位置に同じサイズのブロックができます。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ます。この**「オブジェクトクリック」**機能の意図は、ブロックを定義する作業を支援することです。できたブロックの位置やサイズは後から自由に変更できます。画像やグラフィックにブロックが固定されてしまうわけではなく位置・サイズ決めの補助として使うだけです。

この「オブジェクトクリック」機能は、ページでどのベクトルグラフィックや画像が論理的要素を形成しているかを認識しようと試みます。ページ内容をクリックすると、それが白い場合と極端に大きい場合でなければ、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他の物の一部が入り込んでいればそれが選択領域に追加され、これが繰り返されます。こうして最終的に得られ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ことになります。

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置かれた場所の背景にあるフォントを分析し、対応するブロックプロパティに自動的に書き込むこともできます：

`fontname`・`fontsize`・`fillcolor`・`charspacing`・`horizscaling`・`wordspacing`・`textrendering`・`textrise`

このフォントプロパティの自動検出は結果が変になることもあるため、背景を無視させたいときには**「*PDFlib* ブロック」→「背景フォント・色の検出」**で機能の有効・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　ブロックがうっかり移動・サイズ変更・削除されぬようロックして保護することもできます。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てコンテキストメニューから**「ロック」**を選択します。ブロックがロックされていると、移動・サイズ変更・削除できず、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と、既存ブロックをダブルクリックした時と、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、その選んだブロックのあらゆる設定を編集できます（図 13.2 参照）。プロパティにはブロック種別によってグループがいくつかあり、詳しくは 396 ページ「13.7 ブロックのプロパティ」で説明します。

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だけ**「適用」**ボタンが有効になります。プロパティの変更をブロックに適用した後には、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が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。ロックされたブロックでは**「適用」**ボタンは無効となります。

注 ブロック種別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 プロパティを ruler に設定し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キャラクターの置換が起こることがあります。たとえばストレート引用符がスマート引用符へ置き換えられたりします。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、無効化するには「システム環境設定」→「キーボード」→「ユーザー辞書」→「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プロパティの入力領域に入力するか（*linewidth* など）、ドロップダウンリストから値を選ぶか（*fitmethod*・*orientate* など）、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カラー値やファイル名を選びます（*backgroundcolor*・*defaultimage* など）。*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可能です。どの方式でフォント名を入力したにせよ、PPS がブロックへ内容を流し込むシステムでそのフォントが利用可能でなければなりません。

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。そんなとき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 1 個を名前で選択できます。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、その選択したブロックのプロパティを編集できます。この方法を駆使すれば、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。これらの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でも別の値でオーバーライドしてかまいません。

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、下記以外のブロックプロパティがデフォルトにリセットされます：

- *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ーデータを保持する defaulttext・defaultimage・defaultpdf・defaultgraphics プロパティとは別物ですので注意してください（386 ページ「ブロックのデフォルト内容」参照）。

複数のブロックをまとめて編集　複数のブロックのプロパティをまとめて編集すれば大いに時間を節約できます。複数のブロックを選択するには以下のとおり操作します：

- メニュー項目「*PDFlib* ブロック」→「*PDFlib* ブロックツール」を選択することによって、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。この最初に選んだブロックがマスターブロックになります。他の 1 個ないし複数のブロックを Shift+ クリックすることによっ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」**をクリックすると、現在のページの全ブロックを選択できます。
- これらのいずれかのブロック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。
- あるいはいずれかのブロックをクリックしてマスターブロックにしたうえで***Enter***キーを押すことによっ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。

プロパティのうち、選択しているブロックすべてに該当する部分集合だけがプロパティダイアログに表示されます。プロパティ値はマスターブロックから採られてダイアログに表示されます。この時、選択しているブロックすべてにプロパティ群を以下のように適用できます：

- 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**」をチェックしていないとき：「**適用**」をクリックすると、選択しているすべてのブロックへ、ダイアログで手変更されたプロパティ（黒でハイライトされている）だけがコピーされます。
- 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**」をチェックしているとき：**「適用」**を押すと、選択しているすべてのブロックへ、マスターブロックのすべてのカレントプロパティと、ダイアログで手変更されたすべてのプロパティがコピーされます。これを利用して、ある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できるわけです。

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、複数のブロックでまとめて編集できません：

`Name・Description・Subtype・Type・Rect・Status`

13.3.3 ページ間・文書間でブロックをコピー

Block Plugin は、ブロックを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移動・コピーする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することによって、移動・コピーするか、あるいは他のページか開いている文書へブロックを貼り付ける
- 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つ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）か既存の文書（既存のページにブロックを適用）へ書き出す
- 他の文書からブロックを取り込む

ブロックの定義は維持したままページ内容を差し替えたいときは、ブロックを温存しながら背景の 1 つ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。

ブロックを移動・コピー　ブロックの位置を変えたりコピーを作成したりするには、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。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、名称と位置が自動的に変更される以外は、元のブロックと同じプロパティを持ちます。

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、あるいはその時に Acrobat 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、コピーしたいブロック群を選択。
- Ctrl+C（Windows の場合）か Cmd+C（macOS の場合）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。
- コピー先ページへ移動（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）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・文書へ貼り付け。

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気に作ることもできます：

- ブロックツールをアクティブにして、複製したいブロック群を選択。
- **「*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 ...」**を選択。
- どういう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択。

ブロックを書き出し・取り込み　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温存したままページ内容を差し替えときに便利です。ブロック定義を別ファイルとして書き出すには以下のとおり操作します：

- ブロックツールをアクティブにして、書き出したいブロック群を選択。
- **「*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。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。

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。ブロックを取り込む際には、取り込んだブロックを文書の全ページに適用するのか、それとも特定のページ範囲にだけ適用するかを選べます。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できるわけです。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てから、新たにブロック群がその文書へコピーされます。

13.3.4 Block Plugin のユーザーインターフェイスを XML でカスタマイズ

Block Plugin のユーザーインターフェイスの設定の一部は、Acrobat セッションごとに保管 / 再読み込みされており、XML 構成ファイルで制御可能です。構成ファイルのサンプルとして *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ーの名前はローカライズされている場合があります。必要に応じて *DC* を他の Acrobat トラック名へ置き換えてください）：

```
Windows：C:\Users\<user>\AppData\Local\Adobe\Acrobat\DC\PDFlib\Block Plugin 5
macOS：/Users\<user>/Library/Application Support/Adobe/Acrobat/DC/PDFlib/Block Plugin 5
```

以下の XML エレメントを用いて構成を手動で変更できます：

- 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。このチェックボックスは、ブロック作成後またはブロックプロパティ変更後にもブロックプロパティダイアログが開いたままになるかどうかを決めるものです。
- 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。このチェックボックスは、選択している複数のブロックへコピーされるのはマスターブロックのすべてのプロパティなのか、それともダイアログで変更したプロパティ群のみかを指定するものです。
- 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だけでなくベースフォント 14 種も表示するかどうかを制御。
- エレメント */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。
- エレメント */Block_Plugin/Command/DetectFonts*：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。
- エレメント */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* のいずれかの値をとり、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ーキーを指定：

  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。キーボードショートカットは実行時には変更できないためです。このエントリーがない場合、アクセラレータは利用できません。デフォルトは *control* です。
- エレメント *Configuration/Preferences/PreviewStatusMessage*：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。

13.4 PDF フォームフィールドを PDFlib ブロックに変換

PDFlib ブロックを手作業で作らず、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可能です。これが特に役立つのは、複雑な PDF フォームに PPS で自動流し込みできるようにしたい場合や、既存の大量の PDF フォームを自動流し込みできるよう変換したい場合などです。1 つのページ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。文書のすべてのフォームフィールドを変換したければ**「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したければ（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」**ツールを選択）、**「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においては、**「*PDFlib* ブロック」→「フォームフィールドの変換」→「変換オプション ...」**ダイアログで選んで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 のとおり、それぞれ対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　フォームフィールドの場合には同じページに同じ名前のものが複数あっても許されるのですが、ブロックの場合には名称がページ上で一意でなければなりません。ですので、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の末尾に数を付加して一意なブロック名が生成されます（383 ページ「フォームフィールドを対応するブロックに紐付け」も参照）。

なお、Acrobat の内部的な問題により、同名のフォームフィールドが複数あるとフィールドの属性が正しく報告されません。同名の複数のフィールドで属性が異なると、生成されるブロックにはこの属性の違いが反映されません。変換処理はこの場合には警告メッセージを表示し、関連するフォームフィールド群の名前を示します。その場合には、生成されたブロックのプロパティをよくチェックする必要があります。

フォームフィールドを対応するブロックに紐付け　同名のフィールドが複数あった場合（ラジオボタン等）、変換先の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、どのフォームフィールドがどのブロックになったのかを特定しづらくなります。これは特に、FDF ファイルか XFDF ファイルをソースとしてブロックへ流し込んでその結果をフォーム記入と同じにしたいときに問題となります。

この問題を解決するため、AcroFormConversion プラグインでは、元のフォームフィールドに関する情報を、それに対応するブロックを生成する際に、カスタムプロパティ群として記録します。表 13.2 に、ブロックを正しく特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。

ブロックを対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 ブロックとを同期させるために、生成されたブロックを、その対応するフォームフィールドにバインドさせておくことができます。言い換えれば、プラグイン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、その対応する 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て便

表 13.2 ブロックに対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
PDFlib:field:name	フォームフィールドの完全修飾名。
PDFlib:field:pagenumber	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）。
PDFlib:field:type	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。
PDFlib:field:value	（type=checkbox の場合にのみ可）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ことがあります。
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

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

ブロックを生成した後に変換元のフォームフィールドを残したくなければ、**「*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、変換処理後にフォームフィールドが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可能です。**「*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 個のフォルダー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ーへ書き出すこともできますし、別のフォルダー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せます。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や、ディスク上の画像ファイルなど）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 でブロックに流し込み」で説明します。

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、プレビューを生成しなおしことができます。プレビューは、カレントページについても、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規 PDF 文書として表示されます。元の文書（ブロックを有している）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できますし、捨てることもできます。

ブロックのデフォルト内容　プラグインの場合には、ブロックのテキスト・画像・ベクトルグラフィック・PDF 内容をサーバーサイドのデータソース（データベースなど）から取得するのは無理ですので、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用いられます。具体的には ***defaulttext*・*defaultimage*・*defaultpdf*・*defaultgraphics*** プロパティで指定した値が用いられます。PPS で使う現実のブロック内容を代表するようなサンプルデータ集合をデフォルトデータとして使うのが普通です。デフォルト内容のないブロックはプレビュー生成の際に無視されます。***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、「**高度な *PPS* オプション ...**」ダイアログの同名オプションに適切な値を与える必要があります。

注 記号フォントのデフォルトテキストを入力するのは少しトリッキーな場合があります。詳しくは 390 ページ「デフォルトテキストに記号フォントを使用」を参照してください。

ブロックプレビューを生成　ブロックプレビューを生成するには、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 [P] 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」**。
- ブロックツールがアクティブの時は、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 上で変更を行なっていた場合には、ブロック PDF を**「ファイル」→「上書き保存」**または**「ファイ**

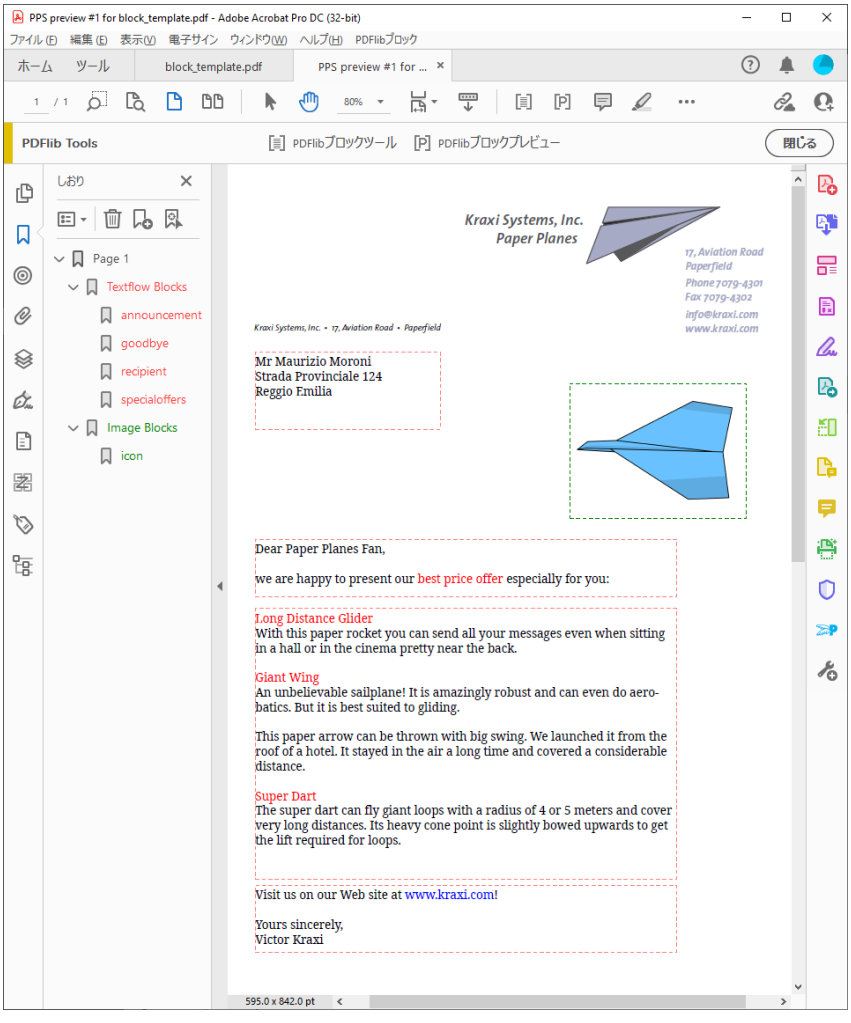

図 13.3
図 13.1 の文書のプレビュー PDF。ブロック情報レイヤーと注釈を内容として持っています。

ル」→「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、ブロック名の後にアスタリスクが付いていることで識別できます。プレビュー作成前にブロック PDF が自動保存されるようプレビュー機能を構成することも可能です。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生成と、その基礎をなす PPS の動作について、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ー。
- 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。

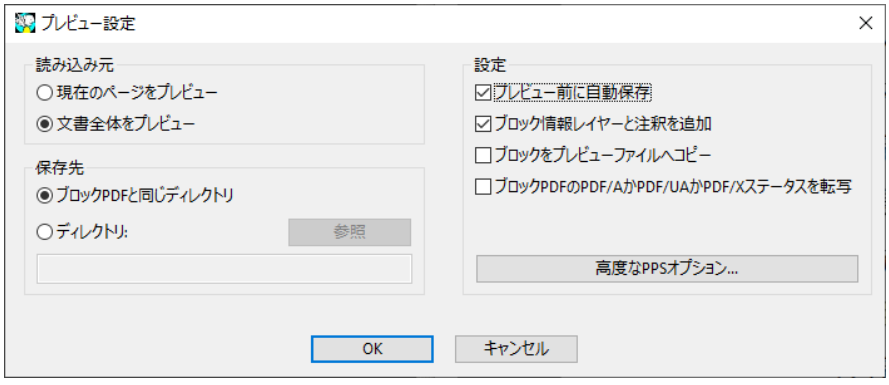

図 13.4 ブロックのプレビューを構成

- **「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」**を使うと、流し込みの際、生成されるプレビューへ PDF ブロック群をコピーできます。ブロックへの流し込みが成功したか否かにかかわらず、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」**ダイアログを使うと、PPS の関数のオプションリストを PPS の API に従って追加指定できます。たとえば、ブロックへの流し込みに使うフォントや画像が位置しているディレクトリーを *PDF_set_option()* の *searchpath* オプションを用いて指定できます。この高度なオプションを指定する際には、PPS のコードの開発者と連携して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されます。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よってではなく）、PDF 文書内での順序が問題となることは通常ありません。

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、以下のアイテムが生成されます：

- **エラーマーカー**：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。エラーマーカーは、ブロックが処理できなかったときには必ず作成されます。
- **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」**（Acrobat X/XI/DC）で表示できます。しおりは必ず作成されます。
- **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、プレビューの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、注釈は生成されません。

- **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)、ブロックの種類毎、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。空のレイヤーは生成されません（エラーが何も起きなかった場合など)。レイヤーの一覧を**「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、レイヤー名の左にある眼のアイコンをクリックします。レイヤーの生成は、プレビューの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、レイヤーは生成されません。

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 出力を生成する必要があるときに有用です。転写モードは、入力が以下の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-3:2003
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：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 画像は使えません。同様に、使用フォントはすべて埋め込む必要があります。すべての要件の一覧を、331 ページ「12.3 PDF/A によるアーカイビング」と、343 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 色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など)、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かなり早い時点で規格違反の危険に気づけます。

デフォルトテキストに記号フォントを使用　記号フォントを用いているブロックにデフォルトテキストを与える方法は 2 通りあります：

- 8 ビットレガシーコードを使用する。このコードはたとえば Windows の文字コード表アプリケーションで表示されます。この 8 ビットコードを ***defaulttext*** に与えます。そのためには具体的には、対応する 8 ビットキャラクターを直接的に入力するか（たとえば Windows の文字コード表からコピー・ペーストすることによって）、あるいは数値のエスケープシーケンスを用います。この場合には、「**テキスト作成**」プロパティグループの ***charref*** プロパティをデフォルト値 ***false*** のままにしておく必要があり、文字参照は使用できません。たとえば、以下のデフォルトテキストは、***charref=false*** であれば、記号フォント Wingdings の「スマイリー」グリフを生成します：

  ```
  J
  \x4A
  \112
  ```

- Unicode 値か、フォント内で用いられているグリフ名を使用する。「**テキスト作成**」プロパティグループの ***charref*** プロパティを ***true*** に設定し、記号の文字参照かグリフ名参照を与えます（121 ページ「5.6.2 文字参照」を参照）。たとえば、以下のデフォルトテキストは、***charref=true*** であれば、記号フォント Wingdings の「スマイリー」グリフを生成します：

  ```
  &#xF04A;
  &.smileface;
  ```

どちらのやり方でも、ブロックプロパティダイアログでは、実際の記号グリフではなく、代替表現が表示されることに留意してください。

13.6 PPS でブロックに流し込み

PPS でブロックに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( )* 関数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んでいます：

```
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", "encoding=winansi");

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　場合によっては、プログラマーが、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 ページの拡縮倍率を、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など。
- 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、それぞれの関数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。

テキストフローブロックを流し込みと同時に移動させる　テキストフローブロックはサイズが固定なので、そのテキスト内容が変わるとそぐわない場合があります。テキストがわずかしかないときには、2 つのブロックの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、テキストフローの流し込みの結果をクエリーして次のブロック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、テキストがブロックの長方形にむりやりはめ込まれます。テキスト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。
- *PDF_fill_textblock()* にダミーオプション *textflowhandle=-1*（PHP では *textflowhandle=0*）を与えると、このメソッドは、ブロックの内容のテキストフローハンドルを返します。
- 返されたテキストフローハンドルを *PDF_info_textflow()* に与えることによって、キーワード *textendy* を用いてテキストの終点をクエリーします。
- *PDF_pcos_get_number()* と pCOS パス *pages[..]/blocks/<blockname>/Rect[1]* を用いてブロックの下辺の位置をクエリーします。
- この 2 つの値の差をとります。このオフセットが正なら、テキストフロー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、テキストフロー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_textblock()* 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。このオプションでは絶対座標が必要ですから、ブロックの縦位置をクエリーする必要があり（前の手順を参照）、そして元の位置とオフセットの和を *refpoint* オプションに与える必要があります。
- この方式は、任意の数のテキストフローブロックに対しても、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層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層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、ページをいったん *PDF_fit_pdi_page()* の *blind* オプションを用いて貼り付けることにより、そのページ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", "encoding=winansi");
/* ... 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、プレースホルダー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うえで、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内部的に１つのテキストフローを生成します。連結されていないブロックの場合には、このテキストフローはそのブロックの中に配置され、その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には、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生成されません。この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チェーンのうちの最初の *PDF_fill_textblock()* を呼び出す際に、*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。*PDF_fill_textblock()* は内部的に生成されたテキストフローハンドルを返しますので、これをアプリケーション側で保持しておく必要があります。
- 次に *PDF_fill_textblock()* を呼び出す際に、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（この時 *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を *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* を与えている場合には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のようになります：

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

for (i = 0; i < blockcount; i++)
{
        String optlist = "encoding=winansi 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、*None* 以外の色空間キーワードを持っているときは、指定された色でブロック領域が塗られます。
- 枠線：*bordercolor* プロパティが存在し、*None* 以外の色空間キーワードを持っているときは、指定された色と線幅でブロックの枠が描線されます。
- 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、何も出力されません。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、ページを拡縮・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：

- ページを *PDF_fit_pdi_page()* で直接貼り付け。
- ページをテーブルセル内に *PDF_fit_table()* で間接的に貼り付け。
- ページを他の PDF ブロックの内容として *PDF_fill_pdfblock()* で貼り付け。

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をネストできます。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、それぞれの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には、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なってから、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・拡縮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へ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方は、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が、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（376 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、すべての HKS・Pantone スポットカラーの一覧を表示できます。これらのカラー名は PPS に内蔵されており、ことさら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可能です。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()*** か然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 には、すべての種類のブロックに適用できる一般プロパティ群と、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が用意されています。

プロパティは、ハンドルとアクションリスト以外、オプションリストと同じデータ型をサポートしています。

ブロックプロパティの名前は一般に、***PDF_fit_textline()*** や ***PDF_fit_image()*** など API 関数のオプションと同じです（たとえば ***fitmethod*** や ***charspacing***）。その場合には動作は、それぞれ対応するオプションの説明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。必須エントリー群は Block Plugin によって自動生成されます。管理プロパティの一覧を表 13.4 に示します。

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 からは無視されます。
Locked	（論理値）true にすると、ブロックとそのプロパティを Block Plugin で編集できなくなります。このプロパティは PPS からは無視されます。デフォルト：false
Name	（文字列、必須）ブロックの名称。このブロック名は、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 種のキャラクター[ ] / 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。なお、テキスト行ブロックとテキストフローブロックはどちらも Subtype が Text になり、両者は textflow プロパティで区別されます。
textflow	（論理値）一行処理か複数行処理かを制御。このプロパティは、Block Plugin のユーザーインターフェイスでは表示されず、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：テキストが一行で組まれ、PDF_fit_textline( ) 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ことができ、PDF_fit_textflow( ) で処理されます。標準テキストプロパティ群のほかにテキストフロー関連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、ブロックの長方形本体の書式を記述します。必須エントリー群は Block Plugin によって自動生成されます。長方形プロパティの一覧を表 13.5 に示します。

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）このプロパティを指定して None 以外の色空間キーワードを与えると、長方形が描かれ、与えた色で塗られます。既存のページ内容を覆い隠したいときに有用です。デフォルト：None
bordercolor	（色）このプロパティを指定して None 以外の色空間キーワードを与えると、長方形が描かれ、与え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）ブロックの長方形の描画に使用する線の描線幅。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 では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式を記述（デフォルト：active）： ***active*** ブロックをそのプロパティに従って完全に処理。 ***ignore*** ブロックを無視。 ***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。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、サーバー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*** 可変内容を配置しない。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：

- 透過書式プロパティの一覧を表 13.6 に示します。これらはすべての種類のブロックに適用されます。
- テキスト書式プロパティの一覧を表 13.7 に示します。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
blendmode	（キーワードリスト。PDF/A-1 モードで用いられるときは値 Normal を持つ必要があります）ブレンドモードの名前：None・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/A モードで用いられるときは値 1 を持つ必要があります）塗り操作の不透明度を範囲 0 ～ 1 で表したもの。値 0 は完全透過を意味し、1 は完全不透過を意味します。
opacitystroke	（float。PDF/A モードで用いられるときは値 1 を持つ必要があります）描線操作の不透明度を範囲 0 ～ 1 で表したもの。値 0 は完全透過を意味し、1 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
charspacing	（float またはパーセント値）文字間隔。パーセント値は fontsize に対する割合。デフォルト：0
decoration-above	（論理値）true の場合、underline・strikeout・overline 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
fillcolor	（色）テキストの塗り色。デフォルト：gray 0（= 黒）
fontname[1]	（文字列）フォントの名前。PDF_load_font( ) 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。fontname の先頭が「@」キャラクターである場合は、そのフォントは縦書きモードで適用されます。ブロックに流し込みを行う際には、PDF_fill_textblock( ) に、font オプションを与えていない限り、テキストのエンコーディングをオプションとして与える必要があります。
fontsize[1]	（float）文字のサイズをポイント単位で指定
horizscaling	（float またはパーセント値）テキストの横伸縮。デフォルト：100%
italicangle	（float）テキストの斜体角度を度単位で指定。デフォルト：0
kerning	（論理値）カーニングの動作。デフォルト：false
overline	（論理値）上線のモード。デフォルト：false
shadow	（複合）影付き効果を生成（デフォルト：影なし）。以下のサブプロパティを使えます： ***fillcolor*** （色）影の色。デフォルト：{gray 0.8} ***offset*** （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。デフォルト：{5% -5%}
strikeout	（論理値）取り消し線のモード。デフォルト：false
strokecolor	（色）テキストの描線色。デフォルト：gray 0（= 黒）
strokewidth	（float・パーセント値・キーワードのいずれか。textrendering 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 percentage に対するパーセント値で）。キーワード auto、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：auto
textrendering	（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）： ***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（ブロックでは不可）
textrise	（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 fontsize に対する割合。デフォルト：0
underline	（論理値）下線のモード。デフォルト：false

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-position	（float・パーセント値・キーワードのいずれか）下線テキストのベースラインに対する描線の相対位置。パーセント値は fontsize に対する割合。デフォルト：auto
underline-width	（float・パーセント値・キーワードのいずれか）下線テキストの線幅。パーセント値は fontsize 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 fontsize 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。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表 13.8 に示します。

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
charref	（論理値）true にすると、数値参照・文字参照・グリフ名参照の置き換えが行われます。デフォルト：グローバルな charref オプション
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・名前文字列内のエスケープシーケンスの置き換えが行われます。デフォルト：グローバルな escapesequence オプション
features	（キーワードのリスト）script・language オプションに従って、OpenType フォントのどのタイポグラフィー機能をテキストに適用するかを指定。フォントにない機能のキーワードは警告なく無視されます。以下のキーワードを使えます： ***_none*** フォント内のどの機能も適用しない。例外として、vert 機能は novert キーワードを用いて明示的に無効化する必要があります。 **<名前>** 4文字のOpenTypeタグ名を与えるとその機能が有効化されます。よく用いられる機能名はliga・ital・tnum・smcp・swsh・zeroなどです。利用できる機能の名前と説明の全一覧を、167ページ「7.3.1 対応しているOpenTypeレイアウト機能」に示しています。 ***no*<名前>** 機能名の前に接尾辞noをつけると（noliga等）その機能が無効化されます。 デフォルト：横書きモードでは _none。縦書きモードでは vert が自動的に適用されます。 OpenType 機能対応を利用するには PDF_load_font( ) の readfeatures オプションが必要です。
language	（キーワード。script を与えられている場合にのみ意味を持ちます）指定した言語に従ってテキストを処理。これは features・shaping オプションに対して意味を持ちます。すべてのキーワードの一覧を 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：_none（言語未定義）
script	（キーワード。shaping=true の場合には必須）指定した用字系に従ってテキストを処理。これは features・shaping・advancedlinebreaking オプションに対して意味を持ちます。最もよく使われる用字系のキーワードは以下の通りです：_none（未定義の用字系）・latn・grek・cyrl・armn・hebr・arab・deva・beng・guru・gujr・orya・taml・thai・laoo・tibt・hang・kana・han。キーワード _auto は、テキスト内の大多数のキャラクターが属する用字系を選択しますが、ただし latn・_none は無視されます。キーワードの全一覧が 176 ページ「7.4.2 用字系と言語」にあります。デフォルト：_none
shaping	（論理値）true にすると、テキストが script・language オプションに従って整形（シェーピング）されます。script オプションの値を _none 以外にする必要があり、かつ、必要なシェーピングテーブルがフォント内で得られる必要があります。デフォルト：false

13.7.5 テキスト組版プロパティ

テキストフローブロックにだけ使用できるテキスト組版プロパティの一覧を表 13.9 に示します。ただし *stamp* プロパティだけはテキスト行ブロックにも使用できます。これらを用いて、テキストフロー処理のための初期オプションリストを構築できます（*PDF_create_textflow( )* の *optlist* 引数に対応）。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できず、サーバー上で *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
adjust-method	（キーワード）minspacing・maxspacing オプションで指定した制限の範囲内で単語間隔を詰めたり拡げたりしてもテキストが行内に収まりきらない時に行の調整に用いる方式（デフォルト：auto）： ***auto*** 以下の方式を順に適用：shrink・spread・nofit・split。 ***clip*** 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す点を除いて、nofit と同じ。 ***nofit*** 最後の単語を次行へ送る。ただし、残される（短い）行が、nofitlimit 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ます。均等配置の段落であっても、若干がたついて見えることがあります。 ***shrink*** 単語が行内に収まらないとき、shrinklimit の制限内でテキストを圧縮。それでも収まらなければ nofit 方式を適用。 ***split*** 最後の単語を次行へ送らずハイフネーションを強制。テキストフォントならハイフンキャラクターを挿入しますが、記号フォントなら挿入しません。 ***spread*** 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 spreadlimit の制限内で拡張。それで均等配置できなければ nofit 方式を適用。
advanced-linebreak	（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。これはタイ語など、単語間の区切りを空白キャラクター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：false
alignment	（キーワード）段落内の行の書式を指定。デフォルト：left。 ***left*** 左揃え（leftindent を始点として）。 ***center*** 中央揃え（leftindent から rightindent までの間で）。 ***right*** 右揃え（rightindent を終点として）。 ***justify*** 両端揃え。
avoid-emptybegin	（論理値）true にすると、はめ込み枠先頭の空行が削除されます。デフォルト：false
fixedleading	（論理値）true にすると、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：false
hortab-method	（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：relative）： ***relative*** hortabsize で指定された分、位置を進めます。 ***typewriter*** hortabsize の次の倍数まで位置を進めます。 ***ruler*** ruler オプション内の n 番目のタブ値まで位置を進めます。ここで n 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、relative 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
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）段落内の最終行の書式。alignment オプションの全キーワードと以下のキーワードを使えます（デフォルト：auto）： ***auto*** alignment オプションの値を使用、ただしそれが justify なら left を使用。
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。デフォルト：100%
locale	（キーワード）advancedlinebreak=true のときに用字系特有の改行方式で用いられるロケール。キーワードは、以下の 1 個ないし複数の構成要素から成り、オプショナルな構成要素は下線キャラクター「_」で区切られます（その文法は、NLS/POSIX のロケール ID とは若干異なっています）： ▸（必須）ISO 639-2（www.loc.gov/standards/iso639-2 を参照）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 _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。デフォルト：_none。 例：de_DE・en_US・en_GB
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、空白キャラクター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：minspacing=50%、maxspacing=500%
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に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」）のを避けられます。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
ruler[2]	（float のリスト、またはパーセント値のリスト）hortabmethod=ruler に対する絶対タブ位置のリスト[1]。このリストは、非負エントリーを昇順で最大 32 個持てます。デフォルト：hortabsize 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
stamp	（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、position・fitmethod・orientate（north・south のみ）オプションに従います。デフォルト：none。 ***ll2ur*** 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。 ***ul2lr*** 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。 ***none*** スタンプは作成されません。
tabalignchar	（整数）タブの小数点揃えの整列位置にしたいキャラクターの Unicode 値。デフォルト：キャラクター「.」（U+002E）
tabalignment[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ーはそれぞれ、ruler オプション内で、その照応するエントリーの整列方式を定義します（デフォルト：left）： ***center*** テキストはタブ位置で中央揃えされます。 ***decimal*** 最初に現れる tabalignchar をタブ位置で左揃えされます。tabalignchar が見つからないときは右揃えが適用されます。 ***left*** テキストはタブ位置で左揃えされます。 ***right*** 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、ただし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：

- 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表 13.10 に示します。
- 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表 13.11 に示します（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
alignchar	（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ー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ー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ーを無効にします。 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
dpi	（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
fitmethod	（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）。
margin	（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
minfontsize	（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%
orientate	（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
position	（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
rotate	（float）回転角を度単位で表したもの。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
scale	(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
shrinklimit	(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
firstlinedist	(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、そこの文字サイズ (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)。 ***leading*** 先頭行について決定された行送り値。À 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。 ***ascender*** 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。 ***capheight*** 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。 ***xheight*** 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。
fitmethod	(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： ***auto*** テキストが収まるまで、fontsize と leading を縮めます。 ***clip*** テキストをブロックの端で切り落とします (テキストフローブロックを連結する場合に有用です)。 ***nofit*** 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
lastlinedist	(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。 ***descender*** 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。
linespread-limit	(float またはパーセント値。verticalalign=justify の場合にのみ可) 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%
maxlines	(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
minfontsize	（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 _boxfull が返されます。デフォルト：0.1%
orientate	（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
rotate	（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
verticalalign	（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）： ***top*** 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。 ***center*** 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。 ***bottom*** 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。 ***justify*** 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 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そのデフォルト内容を流し込むからです。デフォルト内容に関するプロパティの一覧を表 13.12 に示します。

表 13.12 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
default-graphics	（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]
defaultimage	（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]
defaultpdf	（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]
default-pdfpage	（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
defaulttext	（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。

13.7.8 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。カスタムプロパティの命名規則を表 13.13 に示します。

表 13.13 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 種類のキャラクター [ ] / 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ー

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ー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ーする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。なぜならクエリー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ーしたり、ブロックへテキスト・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できます。指定されたブロックが存在しないときには例外が発生します。これを避けるには、*length* 接頭辞を用いれば、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（配列の添字は 0 から始まるので、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す。

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、以下の表現は互いに等価です。ここでは、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ー 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ーできます：

```
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]");
```

なお、上記の座標はデフォルト座標系で与えられます（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 のユーザーインターフェイ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***（すべて小文字の）を用いてクエリーされる値は、関連するいかなる CropBox/MediaBox・Rotate エントリーをも考慮に入れ、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*** を用いてクエリーされる値は、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ー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ー　カスタムプロパティは、以下の例のようにクエリー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*** をクエリーしています：

```
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い場合には、実行時にその名前を取得することも可能です。***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 でなく 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用い、その後にコロン「:」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ム的に作成・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 インターフェイスでプログラム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 データ構造を生成して、*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。すなわちページのメイン *Blocks* 辞書内に 1 回と、特定のブロック辞書内の *Name* エントリー内にもう 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、その対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* エントリーを含んでおり、このエントリー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 13.14 に挙げる 2 個の標準キーを含んでいます。

表 13.14 PDFlib アプリケーションデータ辞書内のエントリー

キー	値
LastModified	（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。このエントリーは、POCA インターフェイスでブロックを作成した場合には PDFlib が作成します。
Private	（辞書、必須）ブロックリスト（表 13.15 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。ブロックリスト辞書の中のキーの一覧を表 13.15 に示します。

表 13.15 ブロックリスト辞書内のエントリー

キー	値
Blocks	（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 13.17 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。
BlockProducer[1]	（文字列）ブロックをプログラム的に作成するのに用いられたソフトウェアの名前。このエントリーは、POCA インターフェイスでブロックを作成した際に PDFlib によって作成されます。
PluginVersion[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。
pdfmark[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。
Version	（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 10 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。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表 13.16 に示します。

ブロック辞書のキー　ブロック辞書は、表 13.17 に挙げるキーを含むことができます。

表 13.16 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。
***float*・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 の数値では点を必要とします。
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ーを持った文字列。
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ーを指定できます： ***/DeviceGray*** 2 番目の要素はグレー値 1 個。 ***/DeviceRGB*** 2 番目の要素は RGB 値 3 個の配列。 ***/DeviceCMYK*** 2 番目の要素は CMYK 値 4 個の配列。 ***[/Separation/ スポットカラー名]*** 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。代替色はそれ自体が 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。 ***[/Lab]*** 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。
unichar	（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 13.17 ブロック辞書内のエントリー

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 13.17 ブロック辞書内のエントリー

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 13.12 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 13.13 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