PDFlib GmbH

pCOS パスリファレンス

PDF 情報取得ツール

pCOS インタフェースバージョン 9

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

疑問がおありの際は、*PDFlib* メーリングリストと、*groups.yahoo.com/neo/groups/pdflib/info* にあるアーカイブをチェックしてください。

ライセンスご希望の際の連絡先：*sales@pdflib.com*
商用 *PDFlib* ライセンス保持者向けサポート：*support@pdflib.com*（お使いのライセンス番号をお書きください）

この出版物およびここに含まれた情報はありのままに供給されるものであり、通知なく変更されることがあり、また、*PDFlib GmbH* による誓約として解釈されるべきものではありません。*PDFlib GmbH* はいかなる誤りや不正確に対しても責任や負担を全く負わず、この出版物に関するいかなる類の（明示的・暗示的または法定に関わらず）保証をも行わず、そして、いかなるそしてすべての売買可能性の保証と、特定の目的に対する適合性と、サードパーティの権利の侵害とを明白に否認します。

PDFlib と *PDFlib* ロゴは *PDFlib GmbH* の登録商標です。*PDFlib* ライセンス保持者は *PDFlib* の名称とロゴを彼らの製品の文書内で用いる権利を与えられます。ただし、これは必須ではありません。

Adobe・*Acrobat*・*PostScript*・*XMP* は *Adobe Systems Inc.* の商標です。

目次

1 はじめに

1.1 pCOS とは

pCOS（*PDFlib Comprehensive Object Syntax*）インタフェースは、PDF 文書のページ寸法・メタデータ・インタラクティブ要素など、ページ内容を記述しないあらゆる部分から技術的情報を取得できるシンプルで洗練された機能を提供します。pCOS を利用するには、内部 PDF 構造と辞書キーの基礎知識がいくらか前提になりますが、PDF 文法と解析の細部を扱う必要はありません。pCOS 利用の際には、*PDF Reference* を入手することを強く推奨します。PDF 1.7 が 2008 年に標準化されて以来、PDF Reference は ISO 32000-1 として入手可能です。この規格文書は www.iso.org から購入できます。この公式版を購入したくない場合は、同じ内容の無料版をダウンロードすることもできます：

Document Management － Portable Document Format － Part 1: PDF 1.7, First Edition
www.adobe.com/devnet/pdf/pdf_reference.html から PDF でダウンロード可能です。

1.2 文書とサンプルの案内

pCOS を使いこなす助けとなるよう、以下の資料を用意しています。

全言語バインディングのためのミニサンプル　*dumper* ミニサンプルは、すべてのパッケージ内で、すべての言語バインディングに対して利用可能です。これは、pCOS を利用するうえでの最小のサンプルコードを提供します。このミニサンプルは、自分の pCOS インストールを試験したり、pCOS アプリケーションの書き方をざっと学んだりするために有用です。

pCOS パスリファレンス　*pCOS* **パスリファレンス**（このマニュアル）は、pCOS インタフェースの核となっている pCOS パス文法の例と簡明な説明を内容としています。pCOS インタフェースは、他のさまざまな PDFlib GmbH 製品に内蔵されていますので、pCOS パスリファレンスは、pCOS を内蔵するすべての製品で使用できます。

おのおのの製品マニュアル　pCOS インタフェースは、スタンドアロン製品としても、他の PDFlib GmbH 製品に内蔵された一部分としても利用可能です。各製品には、それぞれのプログラミングライブラリ（たとえば pCOS や TET）と、それに対応するコマンドラインツール（あれば）の使用方法を記述した製品個別のマニュアルが 1 つないし複数付いています。製品マニュアルは、製品が対応しているさまざまなプログラミング言語を網羅し、その API を具体的に説明しています。

pCOS クックブック　*pCOS* **クックブック**は、pCOS インタフェースのためのコードを集めたものです。下記 URL で入手可能です：

```
www.pdflib.com/pcos-cookbook/
```

pCOS クックブックでは、さまざまなアプリケーションのための pCOS の使い方を具体的に示しています。これは、有用な pCOS プログラミングイディオムのレポジトリとして利用できますので、高く推奨されます。

1.3 pCOS インタフェースの利用可能性

pCOS インタフェースは、PDFlib pCOS という独立した製品として利用可能です。また、他のさまざまな PDFlib GmbH 製品に内蔵された機能としても提供されています。インタフェースが拡張されたり、新しい PDF 入力バージョンへの対応が追加されたりするのに伴い、pCOS インタフェース番号は大きくなっていきます。表 1.1 に、さまざまな製品バージョンに実装されている pCOS インタフェース番号を具体的に示します。

表 1.1 PDFlib GmbH 製品に実装されている pCOS インタフェースバージョン一覧

pCOS インタフェース	対応している PDF 入力バージョン / それにあたる Acrobat バージョン	PDFlib GmbH 製品名とバージョン
1	PDF 1.6 / Acrobat 7	TET 2.0・2.1
2	PDF 1.6 / Acrobat 7	pCOS 1.0
3	PDF 1.7 / Acrobat 8 ISO 32000-1 と同等	PDFlib+PDI 7・PPS 7・ TET 2.2・pCOS 2.0・PLOP 3.0・TET 2.3
4	PDF 1.7 拡張レベル 3 / Acrobat 9 ただし AES-256 暗号化を除く	PLOP 4.0・TET 3.0・TET PDF IFilter 3.0
5	PDF 1.7 拡張レベル 3 / Acrobat 9	PDFlib+PDI 8・PPS 8
6	PDF 1.7 拡張レベル 3 / Acrobat 9	TET 4.0・TET PDF IFilter 4.0
7	PDF 1.7 拡張レベル 8 / Acrobat X ISO 32000-2 と同じ文法と暗号化方式。 PDF 2.0 ともいいます	pCOS 3.0・PLOP 4.1・ PDFlib+PDI 8.1・PPS 8.1
8	PDF 1.7 拡張レベル 8 / Acrobat X/XI ISO 32000-2 と同じ文法と暗号化方式。 PDF 2.0 ともいいます	TET 4.1+・TET PDF IFilter 4.1+ PDFlib+PDI 9.0・PPS 9.0 pCOS 4.0
9	PDF 1.7 拡張レベル 8 / Acrobat X/XI ISO 32000-2 と同じ文法と暗号化方式。 PDF 2.0 ともいいます	PLOP 5.0・PLOP DS 5.0

pCOS インタフェースのいくつかの側面は、TET 製品内でのみ利用可能で、他の PDFlib GmbH 製品にはありません。これらの機能はこのマニュアルでその旨明示しています。

2 pCOS のさまざまな例

この章では、PDF 文書からおのおの値を取得するために利用できる、pCOS パスのさまざまな例を提供します。さらなるプログラムロジックを要するようなより複雑な利用例は、PDFlib Web サイト上の pCOS クックブックで入手可能です。

すべてのプログラミング例は、特記ないかぎり Java 言語で示されています。ただし、わかりきった変更（多くは文法的性質の）を加えることで、pCOS が対応しているどのプログラミング言語でもこれらの例は使うことができます。

あらゆる例をこの章で網羅して示しているわけではありません。他のさまざまな PDF オブジェクトを使うことで、これ以外にも多くの pCOS アプリケーションが可能です。

2.1 pCOS 関数

基本的な pCOS 関数呼び出し　以下の関数が、pCOS で PDF 文書をクエリする際に常用されます：

- *pcos_get_number()*：数値・論理値のいずれかの型のオブジェクトを取得します。
- *pcos_get_string()*：名前・数値・文字列・論理値のいずれかの型のオブジェクトを取得します。
- *pcos_get_stream()*：stream・fstream・文字列のいずれかの型のオブジェクトを取得します。

これらの関数を使えば、pCOS 文法を用いて PDF 文書から情報を取得することができます。pCOS アプリケーションの基本的な構造は以下のようになります：

```
/* PDF文書を開く */
int doc = p.open_document(filename, "");
if (doc == -1)
        throw new Exception("エラー: " + p.get_errmsg());

/* pCOS擬似オブジェクトの値を取得 */
System.out.println(" PDFバージョン: " + p.pcos_get_string(doc, "pdfversionstring"));

p.close_document(doc);
```

pCOS 関数に対する引数は、どの製品でも同じです。それらは、それぞれの製品リファレンスマニュアルに示されています。pCOS 関数プロトライブの早見表が付章 A「pCOS 関数リファレンス」にあります。

プログラムロジックを加える　多くの pCOS オブジェクトは、何個かの要素を持つ配列から成っています。要素数は、*length:* 接頭辞で取得することができます。このときこの配列は、範囲 0 から *length-1* までの整数値を添字にとることができます。以下のコードは、文書内のフォントの数を取得して、各フォントの種類を名前を出力します：

```
count = (int) p.pcos_get_number(doc, "length:fonts");

for (i = 0; i < count; i++) {
    String fonts;

    System.out.print(p.pcos_get_string(doc, "fonts[" + i + "]/type") + " フォント ");
```

```
    System.out.println(p.pcos_get_string(doc, fonts[" + i + "]/name));
}
```

C の整形プレースホルダ　C 言語バインディングでは、pCOS パス内でパラメタの使用を実現できる便利な機能を提供しています。*printf( )* 関数ファミリの整形パラメタと同様に、*%s* と *%d* プレースホルダをそれぞれ文字列と整数のパラメタに対して用いることができます。これらのパラメタの値を、pCOS パスの後に追加の関数引数として加える必要があります。pCOS がプレースホルダを実際の値へ置き換えます。この機能はとくに、配列添字を含むパスに対して有用です。

たとえば、上記の全フォント一覧を作る Java プログラムは、C では以下のように書けます：

```
count = (int) PDF_pcos_get_number(p, doc, "length:fonts");

for (i = 0; i < count; i++)
{
    printf("%s フォント ", PDF_pcos_get_string(p, doc, "fonts[%d]/type", i));
    printf("%s\n", PDF_pcos_get_string(p, doc, "fonts[%d]/name", i));
}
```

最近のプログラミング言語では、より洗練された文字列処理関数群が提供されていますので、この機能は C 言語バインディングでのみ利用可能で、他の言語バインディングにはありません。

2.2 文書

表 2.1 に、一般・文書関連オブジェクトに対する pCOS パスを挙げます。

表 2.1 文書関連項目に対する pCOS パス一覧

pCOS パス	型	説明
pcosmode	数値	文書の pCOS モード、すなわちその暗号化ステータス（37 ページの 4.9「保護された PDF 文書と pCOS モード」を参照）
pdfversionstring	文字列	文書の PDF バージョン番号を表す文字列
/Info/Title	文字列	文書情報フィールド Title。以下のフィールド名が PDF 内で定義済みであり、同様に用いることができます： Title・Author・Subject・Keywords・Creator・Producer・CreationDate・ModDate・Trapped
/Info/ArticleNumber	文字列	カスタム文書情報フィールド ArticleNumber（文書情報項目は任意の名前を用いることができます）
/Root/Metadata	stream	文書のメタデータを持つ XMP ストリーム a
pdfa・pdfe・pdfua・pdfvt・pdfx	文字列	PDF/A・PDF/E・PDF/UA・PDF/VT・PDF/X のいずれかの規格準拠ステータス

暗号化ステータスと pCOS モード *pcosmode* 擬似オブジェクトをクエリすると、その文書の pCOS モードを知ることができます。これは、アクセスが許可されていない（たとえばその文書が暗号化されていて、かつ正しいパスワードが与えられていないために）情報をのちに取得しようとして例外が発生することを防ぐために重要です。あらゆる pCOS アプリケーションについて、*pcosmode* の値に基づいた、以下の一般的な構造が推奨されます：

```
/* PDF文書を開く */
int doc = p.open_document(filename, "requiredmode=minimum");
if (doc == -1)
        throw new Exception("エラー：" + p.get_errmsg());

int pcosmode = (int) p.pcos_get_number(doc, "pcosmode");
boolean plainmetadata = p.pcos_get_number(doc, "encrypt/plainmetadata") != 0;

// つねに得られるユニバーサルな擬似オブジェクトを取得
System.out.println(" PDFバージョン: " + p.pcos_get_string(doc, "pdfversionstring"));
System.out.println("  暗号化: " + p.pcos_get_string(doc, "encrypt/description"));

// 暗号化された文書なのに、パスワードが与えられていない
if (pcosmode == 0)
{
    System.out.println("最小モード: 得られる情報はもうありません\n");
    p.delete();
    return;
}

// そうでないなら他の情報も取得
System.out.println("PDF/Aステータス: " + p.pcos_get_string(doc, "pdfa"));

// マスターパスワードが与えられていないので、メタデータは取得できない
```

```
if (pcosmode == 1 && !plainmetadata && p.pcos_get_number(doc, "encrypt/nocopy") != 0)
{
    System.out.print("限定モード: 得られる情報はもうありません");
    p.delete();
    return;
}

// そうでないなら文書情報フィールドとXMPメタデータも取得できる
...

p.close_document(doc);
```

PDF バージョン　下記のコードは、文書の PDF バージョン番号を出力します：

```
System.out.println(" PDFバージョン: " + p.pcos_get_string(doc, "pdfversionstring"));
```

文書情報フィールド　文書情報フィールドは、以下のコードで取得できます。オブジェクトがその PDF 文書内に実在すること、かつ前提している型を持っていることを確かめるために、まずその型をチェックしましょう。そのオブジェクトが存在し、かつ**文字列**型を持っているならば、これを取得することができます：

```
objtype = p.pcos_get_string(doc, "type:/Info/Title");
if (objtype.equals("string"))
{
        /* 文書情報キーを見つけた */
        title = p.pcos_get_string(doc, "/Info/Title");
}
```

XMP メタデータ　XMP メタデータを内容とするストリームは、以下のコードで取得できます：

```
objtype = p.pcos_get_string(doc, "type:/Root/Metadata");
if (objtype.equals("stream"))
{
        /* XMPメタデータを見つけた */
        metadata = p.pcos_get_stream(doc, "", "/Root/Metadata");
}
```

さまざまな PDF 規格　PDF/A・PDF/E・PDF/UA・PDF/VT・PDF/X のいずれかの規格準拠ステータスは、以下のようなシンプルな pCOS 擬似オブジェクトで取得できます：

```
System.out.println("PDF/Aステータス: "  + p.pcos_get_string(doc, "pdfa"));
System.out.println("PDF/Eステータス: "  + p.pcos_get_string(doc, "pdfe"));
System.out.println("PDF/UAステータス: " + p.pcos_get_string(doc, "pdfua"));
System.out.println("PDF/VTステータス: " + p.pcos_get_string(doc, "pdfvt"));
System.out.println("PDF/Xステータス: "  + p.pcos_get_string(doc, "pdfx"));
```

2.3 ページ

表 2.2 に、ページ関連オブジェクトに対する pCOS パスを挙げます。

表 2.2 ページ関連項目に対する pCOS パス一覧

pCOS パス	型	説明
length:pages	数値	文書内のページの数
pages[...]/width *pages[...]/height*	数値	配列の指定添字のページの幅と高さ（なお、配列添字は 0 から数えます）ステータス

ページ数　文書のページ総数は、以下のように取得できます：

```
pagecount = p.pcos_get_number(doc, "length:pages");
```

ページサイズ　ページの *MediaBox*・*CropBox*・*Rotate* 項目を pCOS で直接得ることは可能ですが、ページの実際のサイズを知るにはそれらを組み合わせて評価する必要があります。*pages* 擬似オブジェクトの *width*・*height* キーを用いれば、はるかに容易にページサイズを知ることができます。以下のコードは、3 ページ目の幅と高さを取得しています（*pages* 擬似オブジェクトの添字は 0 から数えることに留意してください）：

```
pagenum = 2;        // 3ページ目（0から数えて）
width  = p.pcos_get_number(doc, "pages[" + pagenum + "]/width");
height = p.pcos_get_number(doc, "pages[" + pagenum + "]/height");
```

透過　ページの透過が、印刷などの処理の際に影響する場合があります。透過要素を含むページを特定するには、*pages* 擬似オブジェクトの *usespagetransparency* キーを用います：

```
pagenum = 0;        // ページ1（0が先頭）
if (p.pcos_get_number(doc, "pages[" + pagenum + "]/usespagetransparency"))
{
        ...ページは透過要素群を含んでいる...
}
```

2.4 フォント

表 2.3 に、フォント関連オブジェクトに対する pCOS パスを挙げます。

表 2.3 フォント関連特性に対する pCOS パス一覧

pCOS パス	型	説明
length:fonts	数値	文書内のフォントの数
fonts[...]/name	文字列	フォントの名前
fonts[...]/vertical	論理値	フォントが縦書き用かどうかをチェック
fonts[...]/embedded	論理値	フォントの埋め込みステータス
fonts[...]/ascender *fonts[...]/descender*	数値	フォントのアセンダ / ディセンダ値（つねに得られるわけではありません。後出のコードサンプルを参照してください）

すべてのフォントをリスト　以下のコードは、文書内のすべてのフォントのリストを、その埋め込みステータスとともに生成します：

```
count = p.pcos_get_number(doc, "length:fonts");
for (i=0; i < count; i++)
{
        fontname = p.pcos_get_string(doc, "fonts[" + i + "]/name");
        embedded = p.pcos_get_number(doc, "fonts[" + i + "]/embedded");
        /* ... */
}
```

縦書き　以下のコードは、フォントが縦書き用かどうかをチェックしています。フォントはその ID で、すなわち *fonts* 配列の添字によって特定されます。この ID は、とりうる添字値をすべてなめることで得られます：

```
count = p.pcos_get_number(doc, "length:fonts");
for (i=0; i < count; i++)
{
        if (p.pcos_get_number(doc, "fonts[" + i + "]/vertical"))
        {
            /* フォントは縦書き用 */
            vertical = true;
        }
}
```

TET　TET 製品は、フォント ID を *get_char_info()* 関数でも提供します。

フォントメトリック　PDF 内のフォントは、メトリック値群やその他そのフォントに関する情報を持った 1 個のフォント記述子辞書を含んでいることがあります：

```
count = p.pcos_get_number(doc, "length:fonts");
for (i=0; i < count; i++)
{
        ascender  = p.pcos_get_number(doc, "fonts[" + i + "]/ascender");
        descender = p.pcos_get_number(doc, "fonts[" + i + "]/descender");
        /* ... */
}
```

2.5 ラスタ画像

表 2.4 に、ラスタ画像関連オブジェクトに対する pCOS パスを挙げます。

表 2.4 画像関連特性に対する pCOS パス一覧

pCOS パス	型	説明
length:images	数値	文書内のラスタ画像の数
images[...]/Width	数値	画像の幅をピクセル単位で
images[...]/Height	数値	画像の高さをピクセル単位で

すべての画像をリスト　フォントリストと同様に、文書内のすべての画像のリストを生成することもできます :

```
count = p.pcos_get_number(doc, "length:images");
for (i=0; i < count; i++)
{
        width  = p.pcos_get_string(doc, "images[" + i + "]/Width");
        height = p.pcos_get_number(doc, "images[" + i + "]/Height");
        bpc    = p.pcos_get_number(doc, "images[" + i + "]/bpc");
}
```

2.6 インタラクティブ要素

表 2.5 に、インタラクティブ要素関連オブジェクトに対する pCOS パスを挙げます。

表 2.5 さまざまな PDF オブジェクトに対する pCOS パス一覧

pCOS パス	型	説明
length:bookmarks	数値	文書内のしおりの数
bookmarks[...]/Title	文字列	しおりテキスト
bookmarks[...]/destpage	数値	しおりが動作した時の移動先ページの番号、あるいは文書内のページへ飛ぶしおりでない場合は -1
pages[...]/annots[...]/A/URI	文字列	すべてのページ上の Web リンクの移動先 URL
length:fields	数値	文書内のフォームフィールドの数

しおり 以下のコードは、文書内のしおり群をクエリしています。各しおりについて、その入れ子レベル・移動先ページ・タイトルが表示されます :

```
int count = (int) p.get_number(doc, "length:bookmarks");

for (int i = 0; i < count; ++i) {
    int level    = (int) p.get_number(doc, "bookmarks[" + i + "]/level");
    int destpage = (int) p.get_number(doc, "bookmarks[" + i + "]/destpage");

    for (int j = 0; j < level * 4; j += 1) {
        System.out.print(" ");
    }

    System.out.print(p.get_string(doc, "bookmarks[" + i + "]/Title"));

    if (destpage != -1) {
        System.out.print(": ページ " + destpage);
    }
}
```

3 さまざまな pCOS データ型

3.1 基本 PDF データ型

pCOS は、3 つの関数 *pcos_get_number()*・*pcos_get_string()*・*pcos_get_stream()* を提供しています。これらを使えば、PDF 文書内に現れうるすべての基本データ型を取得することができます。PDF 内の特定のオブジェクトのデータ型を知るには、PDF Reference を参照してください。

数値　整数・実数型のオブジェクトは、*pcos_get_number()* でクエリできます。pCOS では、整数と浮動小数点数とを一切区別していません。例：

```
/* 文書内のページの数を得る */
int n_pages = (int) p.pcos_get_number(doc, "length:pages");
```

名前・文字列　名前・文字列型のオブジェクトは、*pcos_get_string()* でクエリできます。例：

```
string title = p.pcos_get_string(doc, "/Info/Title");
```

PDF 内の名前オブジェクトは、非 ASCII キャラクタと、ある種の特殊キャラクタを入れ込むための *#xx* 文法（16 進値に接頭辞）を含んでいることがあります。pCOS は PDF 名前を以下のように処理します：

- 名前オブジェクトは、修飾除去されて（すなわち、*#xx* 文法が解決されて）から返されます。
- 名前オブジェクトは、多くの言語バインディングでは Unicode 文字列として返されます。ただし、C 言語バインディングでは BOM なし UTF-8 値群として返されます。

PDF 内の文字列の大半はテキスト文字列ですので、*pcos_get_string()* はそれらをそのように扱います。しかし、まれな場合において、PDF 内の文字列はバイナリ情報の伝達に用いられています。この場合には文字列は、バイナリ文字列を温存してその内容に対していかなる変更も加えない *pcos_get_stream()* 関数で取得するべきです。例：

```
byte[] signature = p.pcos_get_stream(doc, "", "fields[0]/V/Contents");
```

論理値　論理値型のオブジェクトは、*pcos_get_number()* でクエリすることができ、1（真）または 0（偽）として返されます。例：

```
int linearized_i = p.pcos_get_number(doc, "linearized");
```

pcos_get_string() を用いて論理値オブジェクトをクエリすることもできます。この場合には、それは文字列 *true*・*false* のいずれかで返されます。例：

```
string linearized_s = p.pcos_get_string(doc, "linearized");
```

ストリーム　ストリーム型のオブジェクトは *pcos_get_stream()* でクエリできます。例：

```
byte[] contents = p.pcos_get_stream(doc, "", "/Root/Metadata");
```

PDF 内のストリームデータは、1 個ないし複数の圧縮フィルタで前処理させることもできます。その pCOS データ型（*stream* または *fstream*）に応じて、その内容は圧縮または解

凍されます。*pcos_get_stream()* の *keepfilter* オプションを用いると、クライアントは *stream* 型に対しても圧縮データを取得することができます。

ストリームに存在するフィルタのリストはストリーム辞書からクエリできます。画像の場合には、この情報は、その画像の *filterinfo* 辞書内でのほうがはるかに容易に利用できます。ストリームのフィルタチェーンが対応フィルタのみを含んでいる場合には、その型は *stream* になります。*stream* オブジェクトの内容を取得する際には、*pcos_get_stream()* はすべてのフィルタを除去し、その結果のフィルタなしデータを返します。

注記 pCOS は、JBIG2・JPX ストリームフィルタには対応していません。

ストリームのフィルタチェーン内に 1 個でも非対応フィルタがあるときは、そのオブジェクト型は *fstream*（フィルタされたストリームの意）として報告されます。*fstream* オブジェクトの内容を取得する際には、*pcos_get_stream()* は、フィルタチェーンの先頭にある対応フィルタ群を除去しますが、残余の非対応フィルタ群は温存し、この残余非対応フィルタ群が適用されたままの状態のストリームデータを返します。適用されているフィルタのリストはストリーム辞書からクエリすることができ、また、フィルタされたストリーム内容は *pcos_get_stream()* で取得できます。対応フィルタ群の名前は、そのストリームのフィルタ群の名前をクエリする際には除去されませんので、対応フィルタ群の名前はクライアント側で無視する必要があります。

PDF 内のストリームは通常、バイナリデータを内容としています。しかし、まれな場合においては（テキストストリーム）、それはテキストデータを内容としていることがあります（JavaScript ストリームなど）。正しいテキスト変換を行わせるためには、*pcos_get_stream()* で *convert=unicode* オプションを用います。

3.2 複合データ構造

基本データ型のいずれかのオブジェクトは、2 種類の複合データ構造にまとめることもできます：配列と辞書です。pCOS では、複合オブジェクトを取得するための専用の関数は提供していません。しかし、辞書または配列の中に含まれているオブジェクトは、個別に指定して取得することができます。

配列　配列は、任意の数のオブジェクトの一次元の集合であり、各オブジェクトは任意の型を持つことができます。配列は、ネストした配列群を含むこともできますので、多次元のデータ構造を表現することも可能です。

配列の内容は、それが含む要素の数 ***N*** をクエリ（その配列のパスの前に ***length*** 接頭辞を用いて）したうえで、添字 0 から ***N-1*** までのすべての要素を取得することでなめることができます。

辞書　辞書（連想配列ともいいます）は、任意の数のオブジェクト対を内容とします。各対の第一のオブジェクトは**名前**型を持ち、キーといいます。第二のオブジェクトは値といい、***null*** 以外の任意の型を持つことができます。

辞書の内容は、それが含む要素の数 ***N*** をクエリ（その辞書のパスの前に ***length*** 接頭辞を用いて）したうえで、添字 0 から ***N-1*** までのすべての要素を取得することでなめることができます。辞書をなめると、その辞書のパスの末尾に ***.key*** 接尾辞を用いることで、すべての辞書キーを、PDF 内に格納されている順に取得できます。同様に、おのおのの値は ***.val*** 接尾辞でなめることができます。辞書キーをなめる際には、継承値（後述）と擬似オブジェクトは隠され、***length*** にも数えられません。

PDF 内のページ関連の辞書項目のいくつかは、ツリー状のデータ構造を通じて継承されるので、取得が難しくなっています。たとえば、ページの ***MediaBox*** は、そのページ辞書内に含まれているという保証はなく、任意に複雑なページツリーから継承されている可能性があります。pCOS は、すべての継承されたキーと値を ***pages[ ]*** 擬似オブジェクト内の辞書群の中へ透過的に挿入することによって、この問題を取り除きます。言い換えれば、pCOS を使用すれば、継承可能な項目がすべて辞書内で直接利用可能であると前提することができ、ツリー内の関連するすべての親項目でそれを探す必要がありません。この継承項目の合成は、ページツリーからの取得を ***pages[ ]*** 擬似オブジェクトを通じて行う場合にのみ行われます。***/Pages*** ツリーまたは ***objects[ ]*** 擬似オブジェクトを取得した場合、または ***pages[ ][ ]*** を通じてキーをなめた場合には、おのおのの辞書内に存在する実際の項目が返され、継承は適用されません。

辞書項目を読み取り　以下の例では、文書情報辞書内のキー / 値対をなめています：

```
count = (int) p.pcos_get_number(doc, "length:/Info");

for (i = 0; i < count; i++) {
        String info;
        String key;

        info = "type:/Info[" + i + "]";
        objtype = p.pcos_get_string(doc, info);

        info = "/Info[" + i + "].key";
        key = p.pcos_get_string(doc, info);
        System.out.print(key + ": ");
```

```
        /* 情報項目群は文字列または名前オブジェクトとして格納されている可能性がある */
        if (objtype.equals("name") || objtype.equals("string"))
        {
            info = "/Info[" + i + "]";
            System.out.println("'" + p.pcos_get_string(doc, info) + "'");
        }
    }
```

3.3 オブジェクト識別子（ID）

辞書・配列に対する pCOS ID　PDF オブジェクト ID と異なり、pCOS ID は、pCOS パスを通じて指定された要素に対する一意な識別子を与えることが保証されています（配列・辞書は入れ子になっている場合がありますので、オブジェクトは、その親の配列または辞書と同じ PDF オブジェクト ID を持つ可能性があります）。pCOS ID を取得するには、その辞書または配列のパスの前に ***pcosid*** 接頭辞を付けます。

ですので pCOS ID は、明示的にパスを指定する必要なく要素群を繰り返し取得するためのショートカットとして利用することができます。これにより、大きな配列のすべての要素にわたってループする際のパフォーマンスを向上させます。特定の ID によって指定される要素の内容を取得するには、***objects[]*** 擬似オブジェクトを用います。

4 pCOS パスリファレンス

4.1 pCOS パス文法

pCOS インタフェースのバックボーンは、PDF 文書内に含まれている任意のオブジェクトを指定して取得するためのシンプルなパス文法です。オブジェクトデータそのものだけではなく、pCOS ではオブジェクトに関する情報を取得することもできます。たとえばその型や数などです。オブジェクト型に応じて、関数 *pcos_get_number()*・*pcos_get_string()*・*pcos_get_stream()* のいずれかを用いてオブジェクトの値を得ることができます。pCOS パスの一般的な文法は以下のとおりです：

```
[<接頭辞>:][擬似名[<添字>]]/<名前>[<添字>]/<名前>[<添字>] ... [.key|.val]
```

このさまざまなパス構成要素の意味は以下のとおりです：

- オプショナルな**接頭辞**は、表 4.1 に挙げる値をとることができます。
- オプショナルな**擬似オブジェクト名**は、擬似オブジェクトの名前を内容とすることができます。擬似オブジェクトは PDF 内には存在していませんが、PDF 文書内の 1 個の値を読み取ることによっては容易に取得できない情報への簡便なショートカットを提供するために pCOS で使用できます。**辞書**型の擬似オブジェクトの中の項目群をなめることはできません。
- **名前**構成要素は、文書内で見つかる辞書キーです。複数の名前は / キャラクタで区切られます。pCOS パスは、その文書の Trailer 辞書の中の項目で始まるか、または何らかの人工的なオブジェクトで始まります。この人工的なオブジェクトは、擬似オブジェクトといい、さまざまなデータ構造（ページ等）へのアクセスを簡単化するために pCOS によって追加されているものです。名前はそれぞれ、直前の辞書内に存在する辞書キーである必要があります。フルパスは、初期辞書（Trailer か擬似オブジェクトのいずれかが可能です）からターゲットオブジェクトへの辞書キーの連なりを記述します。
- 配列または辞書を指定するパスまたはパス構成要素は、添字を含むことが可能です。添字は、角括弧に囲んだ 10 進形式で指定する必要があります。入れ子の配列または辞書は、複数の添字項目で指定することができます。配列または辞書の先頭項目の添字は 0 です。
- 辞書を指定するパスまたはパス構成要素は、添字に加えて接尾辞 *.key*・*.val* のいずれかを含むことが可能です。これを用いるとそれぞれ、添字で指定された辞書項目の辞書キーと、それに対応する値を取得することができます。辞書に対するパスが添字を含んでいる場合には、その後にこれらの接尾辞のいずれかを付ける必要があります。

pCOS パスのエンコーディング　多くの場合、pCOS パスは ASCII キャラクタだけを含みます。しかし場合によっては（PDFlib ブロック名など）、非 ASCII キャラクタが必要になる場合もあります。pCOS パスは、以下の規則に従って符号化する必要があります：

- パス構成要素がキャラクタ */[]#* のいずれかを含む場合には、これらは数値記号 # の後に 2 桁の 16 進数の ASCII コードを付けて表現する必要があります。
- Unicode 対応言語バインディングでは、パスは Unicode 文字列から成ります。これは ASCII・非 ASCII キャラクタを含むことができます。
- Unicode 非対応言語バインディングでは、パスは UTF-8 で与える必要があります。文字列は BOM を含んでも含まなくてもよいですが、いずれにせよ違いは生じません。COM はパスの先頭に入れることもできますし、各パス構成要素の先頭に（すなわちスラッ

シュキャラクタの後に）入れることもできます。EBCDIC システム上では、パスは通常、***ebcdic*** エンコーディングで与える必要があります。ASCII 文字集合外のキャラクタは、EBCDIC-UTF-8（BOM はあってもなくてもよい）として与える必要があります。

4.2 パス接頭辞

接頭辞を用いると、オブジェクトのさまざまな属性をクエリすることができます。表 4.1 に、使えるすべての接頭辞を挙げます。

length 接頭辞と、添字を通じて内容をなめることは、プレーン PDF オブジェクトと、**配列**型の擬似オブジェクトでのみ利用可能であり、辞書型の擬似オブジェクトでは利用できません。*pcosid* 接頭辞は擬似オブジェクトには適用できません。*type* 接頭辞は、すべての擬似オブジェクトに対して適用できます。

表 4.1 pCOS パス接頭辞一覧

接頭辞	説明
length	（数値）オブジェクトの子の数。オブジェクトの型によって異なります： ***array*** 配列内の要素の数 ***dict*** 辞書内のキー / 値対の数 ***stream*** ストリーム辞書内のキー / 値対の数（ストリーム長ではありません。ストリームデータの長さをバイト単位で知るには Length キーを用います） ***fstream*** stream と同じ ***other*** 0
pcosid	（数値）辞書型または配列型のオブジェクトに対する一意な pCOS ID。 パスが、PDF 内に存在しないオブジェクトを記述している場合には、結果は –1 になります。これを利用して、オブジェクトの有無をチェックすることができます。それが存在していれば同時に ID が得られます。
type	（数値か文字列）オブジェクトの型を数値または文字列で表したもの。 ***0, null*** ヌルオブジェクト、またはオブジェクトが存在しない（オブジェクトの有無チェックに使用）。pCOS インタフェース 9：このパスの中で指定されたオブジェクトへのアクセスを試みている最中に PDF 文法エラーに遭遇した場合にもこの値が返されます。 ***1, boolean*** 論理値オブジェクト ***2, number*** 整数または浮動小数点数 ***3, name*** 名前オブジェクト ***4, string*** 文字列オブジェクト ***5, array*** 配列オブジェクト ***6, dict*** 辞書オブジェクト（ただしストリームでない） ***7, stream*** 対応フィルタのみを用いているストリームオブジェクト ***8, fstream*** 1 個ないし複数の非対応フィルタを用いているストリームオブジェクト C・C++ 開発者の便宜のため、これらの型の enum が利用可能です。

4.3 普遍擬似オブジェクト

普遍擬似オブジェクトは、どの *pcosmode* レベルの場合にも、すなわち暗号化とパスワード入手可能性によらず利用可能です。表 4.2・表 4.3・表 4.4 に、すべての普遍擬似オブジェクトを挙げます。

4.3.1 一般文書情報

表 4.2 一般文書情報のための普遍擬似オブジェクト一覧

オブジェクト名	説明
encrypt	（辞書）文書の暗号化ステータスを記述したキー群を持つ辞書： ***length***　（数値）暗号化鍵の長さをビット単位で ***algorithm***（数値） ***description***（文字列）暗号化アルゴリズム番号または説明： ***-1*** 未知の暗号化 ***0*** 暗号化なし ***1*** 40 ビット RC4（Acrobat 2 ～ 4） ***2*** 128 ビット RC4（Acrobat 5） ***3*** 128 ビット RC4（Acrobat 6） ***4*** 128 ビット AES（Acrobat 7） ***5*** 128 ビット RC4 に公開鍵（Acrobat 5）[1] ***6*** 128 ビット AES に公開鍵（Acrobat 7）[1] ***7*** Adobe Policy Server（Acrobat 7）[1] ***8*** Adobe Digital Editions（EBX）[1] ***9*** （pCOS インタフェース 5）256 ビット AES（Acrobat 9） ***10*** （pCOS インタフェース 5）256 ビット AES に公開鍵（Acrobat 9）[1] ***11*** （pCOS インタフェース 7）256 ビット AES（Acrobat X/XI） ***master***　（論理値）セキュリティ設定（権限、ユーザーまたはマスターパスワード）を変えるためのマスターパスワードを PDF が必要とするなら true、そうでないなら false ***user***　（論理値）開くためのユーザーパスワードを PDF が必要とするなら true、そうでないなら false ***attachment*** （論理値、pCOS インタフェース 8）その PDF で添付を抽出するためにパスワードが必要ならば（しかし開くためには必要でないなら）true、そうでないなら false ***noaccessible・noannots・noassemble・nocopy・noforms・nohiresprint・nomodify・noprint*** （論理値）それぞれアクセス制限が設定されているなら true、そうでないなら false ***plainmetadata*** （論理値）PDF が暗号化されているが、暗号化されていないメタデータを含んでいるなら true、そうでないなら false
filename	（文字列）PDF ファイルの名前
filesize	（数値）PDF ファイルのサイズをバイト単位で
linearized	（論理値）PDF 文書が線形化されているなら true、そうでないなら false
pcosmode ***pcosmode-name***	（数値か文字列）pCOS モードを数値または文字列で： ***0・minimum*** ：最小 ***1・restricted*** ：限定 ***2・full*** ：完全

表 4.2 一般文書情報のための普遍擬似オブジェクト一覧

オブジェクト名	説明
revisions	（数値。pCOS インタフェース 9）その PDF に含まれている文書リビジョンの数。ここで各リビジョンは、増分的な 1 個の PDF 更新セクションによって記述されています。1 個の文書が複数の署名を含んでいる場合には、各署名は個々の更新セクション内で適用されていますが、1 個の更新セクションが、注釈の追加・削除やフォームフィールドの記入等、他の諸変更を含んでいる場合もあります。最初の署名は、必ずしも増分的な PDF 更新を追加するとは限りません。
shrug	（論理値。TET・PDFlib+PDI・PPS・PLOP・PLOP DS 製品でのみ）PDF 文書を開く際にセキュリティ設定が無視された場合にのみ true。文書作成者の意思を尊重するのはクライアント側の責務です。以下の条件がすべて真であればこの値は true になります： ▸ shrug オプションでシュラッグモードが有効にされている。 ▸ 文書はマスターパスワードを持っているが、これが与えられていない。 ▸ ユーザーパスワード（その文書に対して必要ならば）が与えられている。 ▸ TET 製品のみ：文書の権限設定で内容抽出が許されていない。

1. このアルゴリズムで暗号化された文書の識別はできますが、実際の復号には対応していません。

4.3.2 PDF バージョン情報

表 4.3 PDF バージョン情報のための普遍擬似オブジェクト一覧

オブジェクト名	説明
extension-level	（数値）ISO 32000 に基づく Adobe 拡張レベル、あるいは拡張レベルが存在しない場合には 0（ゼロ）。Acrobat 9 は拡張レベル 3 の文書を生成します。Acrobat X/XI は拡張レベル 8 を生成します。
fullpdf-version	（数値）PDF バージョン番号の数値を、100× ベースバージョン × 拡張レベルとして表したもの。 例： *150* PDF 1.5（Acrobat 6） *160* PDF 1.6（Acrobat 7） *170* PDF 1.7（Acrobat 8）＝ ISO 32000-1 *173* PDF 1.7 Adobe 拡張レベル 3（Acrobat 9） *178* （pCOS インタフェース 5）PDF 1.7 Adobe 拡張レベル 8（Acrobat X/XI） *200* （pCOS インタフェース 5）PDF 2.0 ＝ ISO 32000-2
pdfversion	（数値）PDF バージョン番号に 10 をかけたもの。例：PDF 1.7 なら 17
pdfversion-string	（文字列）PDF 出力互換性を設定するためにさまざまな API 関数が受け付ける形の完全 PDF バージョン文字列。例：1.5・1.6・1.7・1.7ext3・1.7ext8

pCOS 最小モード（すなわち、暗号化されたファイルに対して必要なユーザーパスワードが与えられない）では、この拡張レベルはバージョン情報から抜け落ちている可能性があります（たとえば 1.7ext3 でなく 1.7 と報告される）。なぜなら拡張レベルに関する情報を復号できないためです。

4.3.3 ライブラリ識別

表 4.4 ライブラリ識別のための普遍擬似オブジェクト一覧

オブジェクト名	説明
major *minor* *revision*	（数値）それぞれライブラリのメジャー・マイナー・リビジョン番号。
pcosinterface	（数値）基礎 pCOS 実装のインタフェースバージョン番号。特定の製品バージョンにどのバージョンの pCOS インタフェースが実装されているかを知るには、6 ページの 1.3「pCOS インタフェースの利用可能性」を参照してください。
version	（文字列）< メジャー >.< マイナー >.< リビジョン > の形式の完全ライブラリバージョン文字列に、ものによっては beta・rc といった修飾子が後に付いたもの

4.4 PDF 規格識別のための擬似オブジェクト

表 4.5 に、PDF 規格識別のための擬似オブジェクトを挙げます。これらの擬似オブジェクトの値は、文書内のそれぞれの規格識別項目に基づいて生成されています。これらは、規格に対する妥当性の検証は一切行いません。

表 4.5 PDF 規格識別のための擬似オブジェクト一覧

オブジェクト名	説明
pdfa	（文字列）文書の PDF/A（ISO 19005-1・19005-2）準拠レベル。とりうる値： none 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（pCOS インタフェース 8）
pdfe	（文字列。pCOS インタフェース 5）文書の PDF/E（ISO 24517-1・24517-2）準拠レベル。とりうる値： none PDF/E-1 PDF/E-2（pCOS インタフェース 7）
pdfua	（文字列。pCOS インタフェース 7）文書の PDF/UA（ISO 14289）準拠レベル。とりうる値： none PDF/UA-1
pdfvt	（文字列。pCOS インタフェース 7）文書の PDF/VT（ISO 16612-2）準拠レベル。とりうる値： none PDF/VT-1 PDF/VT-2
pdfx	（文字列）文書の PDF/X（ISO 15930-1 など）準拠レベル。とりうる値： none PDF/X-1:2001・PDF/X-1a:2001・PDF/X-1a:2003 PDF/X-2:2003 PDF/X-3:2002・PDF/X-3:2003 PDF/X-4・PDF/X-4p PDF/X-5g・PDF/X-5n・PDF/X-5p

4.5 ページのための擬似オブジェクト

表 4.6 に、ページ情報のための擬似オブジェクトを挙げます。

表 4.6 ページのための擬似オブジェクト一覧

オブジェクト名	説明
pages	（辞書の配列）各配列要素が、文書のページを指定します。これを、ページ番号 –1 の 10 進表現で添字指定すると、そのページを指定できます（先頭ページの添字が 0）。length 接頭辞を用いると、文書内のページの数を知ることができます。この方法で指定したページオブジェクトは、/Pages ツリーを通じて継承されるすべての属性を取り入れます。/MediaBox・/Rotate 項目は存在することが保証されています。各ページに対して、標準 PDF 辞書項目のほか、以下の擬似項目も利用可能です： ***colorspaces・extgstates・fonts・images・patterns・properties・shadings・templates*** （辞書の配列）表 4.9 に従ったページリソース群。 ***annots*** （辞書の配列）pCOS ではこの annots 配列内の辞書に対して、Annots 配列内の標準 PDF キーのほか、以下の擬似キーも用いることができます： ***destpage*** （数値。Subtype=Link、かつ Dest 項目か GoTo アクションが存在する場合のみ）ターゲットページの番号（先頭ページが 1） ***blocks*** （辞書の辞書）pages[]/PieceInfo/PDFlib/Private/Blocks、すなわち、PDFlib Personalization Server（PPS）で用いるためのそのページのブロック群のリスト。pCOS ではこの blocks 配列内の辞書に対して、既存 PDF キーのほか、以下の擬似キーも用いることができます： ***rect*** （矩形）Rect と同様ですが、ただし関連する CropBox/MediaBox ・Rotate 項目を考慮に入れ、また、座標順序を正規化します。 ***height*** （数値）ページの高さ。MediaBox または CropBox（存在すれば）を用いて高さが決定されます。Rotate 項目も適用されます。 ***fields*** （辞書の配列。pCOS インタフェース 9）そのページ上のフォームフィールド群に対する辞書群を持った配列。グローバルな fields[] 配列に対するのと同じ擬似辞書キー群を使うことができます（29 ページの表 4.7 参照）。署名フィールドに対しては、signaturefields[] 配列に対するのと同じ擬似辞書キー群も使うことができます（31 ページの表 4.8 参照）。 ***isempty*** （論理値）ページが空なら true ***label*** （文字列）ページのページラベル（接頭辞があればそれも含む）。ラベルは Acrobat と同様に表示されます。ラベルが存在しないときは（または PageLabel 辞書が破損しているときは）、この文字列の内容は 10 進ページ番号になります。ローマ数字は Acrobat のスタイル（VL など）で生成され、それとは異なる古典的スタイル（XLV など）では生成されません。/Root/PageLabels が存在しない場合には、その文書はページラベルを一切含んでいません。 ***usespagetransparency***[1] （論理値、pCOS インタフェース 8）ページ内容が透過要素を含んでいるなら true、そうでないなら false。 ***usesanytransparency***[1] （論理値、pCOS インタフェース 8）ページ内容かページ上の注釈が透過要素を含んでいるなら true、そうでないなら false。 ***width*** （数値）ページの幅（height と同じ規則） 右記の項目は継承されます：CropBox・MediaBox・Resources・Rotate.

1. これらのチェックは、そのページのリソース（フォーム XObject・画像等）内で見つかった透過を、これらのリソースが可視のページ内容を生成しているために実際に用いられているかどうかにかかわらずレポートします。透過は PDF/VT 規格と同じに定義されます。

4.6 インタラクティブ要素のための擬似オブジェクト

表 4.6 に、PDF オブジェクトを取得するために用いることができる、またはさまざまなインタラクティブ要素へのショートカットとして機能する擬似オブジェクトを挙げます。

表 4.7 PDF オブジェクトとインタラクティブ要素のための擬似オブジェクト一覧

オブジェクト名	説明
articles	（辞書の配列）文書に対するアーティクルスレッド辞書群を内容とする配列。文書がアーティクルスレッドを一切含んでいない場合にはこの配列は要素数 0 になります。pCOS ではこの articles 配列内の辞書に対して、標準 PDF キーのほか、以下の擬似キーも用いることができます： ***beads*** （辞書の配列）標準 PDF キーのほかに以下も持つ Bead 辞書： ***destpage*** （数値）ターゲットページの番号（先頭ページが 1）
bookmarks	（辞書の配列）文書に対するしおり（アウトライン）辞書群を内容とする配列。pCOS ではこの bookmarks 配列内の辞書に対して、標準 PDF キーのほか、以下の擬似キーも用いることができます： ***destpage*** （数値。Dest 項目か GoTo アクションが存在する場合のみ）しおりが同一文書内のページを指している移動先か GoTo アクションを含んでいる場合にはそのターゲットページの番号（先頭ページが 1）、そうでないなら –1。 ***level*** （数値）しおり階層構造内の字下げレベル
destpage	（数値。pCOS インタフェース 9）その文書が開かれた時に表示されるターゲットページの番号（先頭ページが 1）。この値は、その文書の open アクションか移動先が存在する場合にはそこから取られ、そうでないなら 1 となります。
fields	（辞書の配列）文書に対するフォームフィールド辞書群を内容とする配列。pCOS ではこの fields 配列内の辞書に対して、フィールド辞書内の標準 PDF キーと、紐付けられた Widget 注釈辞書内の項目のほか、以下の擬似キーも用いることができます： ***exportvalue***[1]（文字列。pCOS インタフェース 9）そのフィールドの書き出し値 ***fullname*** （文字列）そのフォームフィールドの完全名。無名のフィールドに対してはその親フィールドの名前が用いられます。同一レベルに複数の無名の兄弟がある場合には、この名前には接尾辞 #N が付加されます。ここで N は 0 で始まる連続する整数です。 ***level*** （数値）フィールド階層構造内のレベル（「.」を区切り文字として決定されます） ***parent*** （数値。pCOS インタフェース 9）そのフィールドの親ノードの fields[] 配列内における番号。そのフィールドが親フィールドを持たない場合、この値は –1 となります。 ***type*** （文字列。pCOS インタフェース 9）フィールド種別：barcode・container（フォームツリー内において、それ自身はフィールドを表さず、ラジオボタン以外の他のフィールド群のためのコンテナとしてのみ働くノード）・checkbox・combobox・listbox・pushbutton・radiobutton・radiogroup（ラジオボタン群のためのコンテナ）・signature・textfield ***value***[1] （さまざまな型。pCOS インタフェース 9）フィールド値（V キーから、あるいはラジオボタン・チェックボックスの場合には Opt 配列から取得される） ***visible*** （論理値。pCOS インタフェース 9）そのフィールドが可視なら true。

表 4.7 PDF オブジェクトとインタラクティブ要素のための擬似オブジェクト一覧

オブジェクト名	説明
names	（辞書）各項目が 1 個の名前ツリーへのシンプルなアクセスを提供する辞書。右記の名前ツリーを用いることができます：AP・AlternatePresentations・Dests・EmbeddedFiles・IDS・JavaScript・Pages・Renditions・Templates・URLS。 各名前ツリーは、その名前をキーとして、それに対応する値を取得するために用いることによってアクセスすることができます。例： names/Dests[0].key で移動先の名前を取得 names/Dests[0].val でそれに対応する移動先辞書を取得 Dests 名前ツリー内の辞書に対しては、標準 PDF 辞書項目のほか、以下の擬似キーも用いることができます： ***destpage*** （数値）移動先が同一文書内のページを指している場合にはそのターゲットページの番号（先頭ページが 1）、そうでないなら –1。 他の名前ツリー項目を取得するためには、それらは名前ツリー擬似オブジェクト内には存在していませんので、/Root/Names/Dests などを通じて直接クエリする必要があります。
objects	（配列）それまでに pcosid 接頭辞を用いて pCOS ID が取得されている要素を指定します。この ID を 10 進形式で配列添字として与える必要があります。結果として、与えた ID を持つ PDF オブジェクトが指定されます。この配列では length 接頭辞を用いることはできません。
tagged	（論理値）PDF 文書がタグ付きなら true、そうでないなら false

1. この擬似オブジェクトは必ず得られるわけではありません。その存在を、type 接頭辞を用いてチェックする必要があります。

4.7 署名のための擬似オブジェクト

表 4.8 に、署名関連情報を取得するために用いることのできる擬似オブジェクトを挙げます。

表 4.8 署名のための擬似オブジェクト一覧

オブジェクト名	説明
signaturefields	（辞書の配列。pCOS インタフェース 9）その文書の中のすべての署名済と未署名の署名を内容とする配列。この配列は、すべての署名済フィールドをその署名順に、次いですべての未署名フィールドを内容とします。この署名フィールドの照応する fields[ ] 擬似オブジェクト内の項目の中の項目群に加えて、pCOS では、この配列の中の辞書群に対して以下の擬似キーを使えます： ***cades*** （論理値）このフィールドが署名済で、かつ、CAdES 署名を内容としている場合には true、そうでないなら false ***field*** （整数）fields[] 配列内の照応するフォームフィールドの番号 ***fillablefields*** （論理値。sigtype=certification の場合のみ）その認証署名がフォームフィールド群を保護しているなら true、そうでないなら false ***permissions*** （文字列。sigtype=certification の場合のみ）認証署名を無効とせずに許される文書変更：nochanges・formfilling・formsandannotations ***preventchanges*** （論理値。sigtype=certification の場合のみ）その認証署名が Acrobat に対して、使用されるとその署名が無効となるユーザーインタフェース要素群を隠すよう指示しているならば true ***sigtype*** （文字列）署名の種別：none（未署名フィールドの場合）・approval・certification・doctimestamp のいずれか
usagerights	（論理値。pCOS インタフェース 9）その文書が、署名済の使用権限を内容としているなら true。このような文書は、Reader 拡張機能が有効な（Reader-enabled）文書としても知られています。

4.8 リソースのための擬似オブジェクト

リソースは、ページの内容を完全に記述するために必要なさまざまな種類のデータを管理するためのキーコンセプトです。PDF 内のリソースのコンセプトは非常に強力で効果的なものですが、再帰や継承といったさまざまな技術的なコンセプトでアクセスが複雑になっています。pCOS では、リソースの取得を大幅にシンプルなものにして、リソースを直接クエリするために利用できるいくつかのグループの擬似オブジェクトを提供しています。これらの擬似リソース辞書のいくつかは、標準 PDF キーのほかに、リソース情報の取得をさらにシンプルにするための項目も含んでいます。pCOS 擬似リソースは、ユーザーの視点から見たリソースを反映しており、ネイティブな PDF リソースとは異なっています:

- いくつかの項目が追加されたり（インライン画像や、シンプルな色空間など）、削除されたり（どのページでも使われていないフォントなど）している場合があります。
- pCOS リソース辞書は、オリジナルの PDF 辞書キーのほかに、さらなる情報のためのいくつかのユーザーフレンドリーなキーも含んでいる場合があります（フォントの埋め込みステータスや、色空間の構成要素数など）。

pCOS では、リソース取得のための 2 個のグループの擬似オブジェクトを用いることができます。グローバルリソース配列は、PDF 文書内の特定の種類のリソースすべてを内容とし、一方、ページベースリソースは、特定のページで使われているリソースだけを内容とします。おのおのの擬似配列は、表 4.9 に挙げるリソースの種類すべてに対して利用可能です：

- 文書内の全リソースのリストは、グローバルリソース配列（*images[ ]* など）内で得られます。グローバルリソース擬似配列のうちのいずれかの要素数を取得すると、全ページに対するリソーススキャンが行われます。
- 各ページ上のリソースのリストは、ページベースリソース配列（*pages[ ]/images[ ]* など）内で得られます。ページのリソース擬似配列のいずれかの要素数を取得すると、そのページに対するリソーススキャンが行われます（そのページ上で実際に使われているリソースすべてを収集するために、および、そのページ上の画像を連結するために）。

表 4.9 リソースのための擬似オブジェクト一覧。各リソースカテゴリ P は 2 個のリソース配列 P[ ] と pages[ ]/P[ ] を生成します。

オブジェクト名	説明
colorspaces	（辞書の配列。ただし name=ICCBased の場合には型は stream）ページ上の、または文書内のすべての色空間に対する辞書またはストリームを内容とする配列。色空間リソース群は、ネイティブ PDF リソースを必要としない色空間（すなわち DeviceGray・DeviceRGB・DeviceCMYK）を含め、任意の種類のオブジェクトから参照されるすべての色空間を含んでいます。色空間辞書（色空間が PDF 内の辞書によって表されている場合）・ICC プロファイルストリーム辞書内の標準 PDF キーのほか、以下の擬似キーも用いることができます： ***alternateid*** （整数。name=Separation・DeviceN の場合のみ）colorspaces[] 擬似オブジェクト内の基礎代替色空間の番号。 ***baseid*** （整数。name=Indexed の場合のみ）colorspaces[] 擬似オブジェクト内の基礎基調色空間の番号。 ***colorantname*** （名前。name=Separation の場合のみ）スポットカラーの名前。非 ASCII 日中韓色名は Unicode へ変換されます。 ***colorantnames*** （名前の配列。name=DeviceN の場合のみ）スポットカラー群の名前群 ***components*** （整数）色空間の構成要素の数 ***name*** （文字列）色空間の名前：CalGray・CalRGB・DeviceCMYK・DeviceGray・DeviceN・DeviceRGB・ICCBased・Indexed・Lab・Separation ***csarray*** （配列。name=DeviceGray/RGB/CMYK の場合は不可）基礎ネイティブ色空間、すなわち PDF 内のネイティブ色空間オブジェクトを記述した配列。
extgstates	（辞書の配列）ページ上の、または文書内のすべての拡張グラフィックステート（ExtGState）に対する辞書を内容とする配列。

表 4.9 リソースのための擬似オブジェクト一覧。各リソースカテゴリ P は 2 個のリソース配列 P[ ] と pages[ ]/P[ ] を生成します。

オブジェクト名	説明
fonts	（辞書の配列）ページ上の、または文書内のすべてのフォントに対する辞書を内容とする配列。フォント辞書内の標準 PDF キーのほか、以下の擬似キーも用いることができます： ***ascender*** （float。pCOS インタフェース 6）フォントのアセンダ。得られるかどうかによって、この値は PDF 内の FontDescriptor 辞書から採られるか、あるいは推算値になります。この値は、フォントサイズに対する比の 1000 倍で表現されます。すなわち 1000 単位がフォントサイズにちょうど等しくなります。 TET 製品：フォントメトリック値の決定にあたっては、PDF 内の辞書値群のほかに、埋め込まれているフォントと、Mac または Windows システムにインストールされているフォントも分析されます。フォント分析の結果は、この特定フォント内のグリフで TET_get_char_info( ) を呼び出した後にのみ利用可能です。言い換えれば、TET_get_char_info( ) が返すフォント ID は安全に使えますが、fonts[] 配列内の全フォントをなめた場合には必ずしも、埋め込まれているフォントデータ内のメトリック値が得られるとは限らず、PDF フォント記述子内の不正確かもしれない値を得る場合があります。 ***capheight*** （float。pCOS インタフェース 6）フォントのキャップハイト。ascender 参照 ***italicangle*** （float。pCOS インタフェース 6）フォントのイタリック（斜形化）角度を度単位で表したもの ***name*** （文字列）フォントの PDF 名に、サブセット接頭辞を一切付けないもの。非 ASCII 日中韓フォント名は Unicode へ変換されます。 ***descender*** （float。pCOS インタフェース 6）フォントのディセンダ。ascender 参照 ***embedded*** （論理値）フォントの埋め込みステータス ***fullname*** （文字列。pCOS インタフェース 5）フォントの PDF 名に、もしあればサブセット接頭辞を含めたもの。非 ASCII 日中韓フォント名は Unicode へ変換されます。 ***type*** （文字列）フォント種別：(unknown)・Composite・Multiple Master・OpenType・TrueType・TrueType (CID)・Type 1・Type 1 (CID)・Type 1 CFF・Type 1 CFF (CID)・Type 3 ***vertical*** （論理値）縦書き用フォントなら true、そうでないなら false ***weight*** （float。pCOS インタフェース 6）範囲 0 〜 900 のフォントウェイト：0= 何の情報も得られない、400=normal、700=bold ***xheight*** （float。pCOS インタフェース 6）フォントの x ハイト。ascender 参照

表 4.9 リソースのための擬似オブジェクト一覧。各リソースカテゴリ P は 2 個のリソース配列 P[ ] と pages[ ]/P[ ] を生成します。

オブジェクト名	説明
images	（ストリームの配列）ページ上の、または文書内のすべての画像に対する辞書を内容とする配列。TET 製品では、連結済（擬似）画像もこの images[ ] 配列に追加されます。 標準 PDF キーのほか、以下の擬似キーも用いることができます： ***bpc*** （整数）構成要素あたりビットの数。この項目は、PDF キー BitsPerComponent と通常同じですが、ただしこれとは異なり、つねに得られることが保証されています（特に、画像マスクはこのキーを内容として持っていない可能性があります）。JPEG 2000 画像の場合には、構成要素あたりビットの数は PDF 構造内で得られない場合があるため、bpc は –1 になる場合があります。 ***colorspaceid*** （整数）画像の色空間の、colorspaces[] 擬似オブジェクト内における番号。これを用いて、具体的な色空間特性群を取得することができます。JPEG 2000 画像の場合には、色空間は PDF 構造内に符号化されていない場合があるため、この色空間 ID は –1 になる場合があります。マスク画像の場合には DeviceGray の id が返されます。 ***filterinfo*** （辞書）非対応フィルタを持つストリームの場合の、または keepfilter オプションを true に設定してストリームデータを取得したときの残余フィルタを記述します。そのようなフィルタが一切ないときは、filterinfo 辞書は全く得られません。この辞書は以下の項目を内容とします： ***name*** （名前）フィルタの名前 ***supported*** （論理値）対応フィルタなら true ***decodeparms*** （辞書）DecodeParms 辞書を、フィルタ内にそれが存在する場合に ***maskid*** （整数。pCOS インタフェース 9。TET でのみ使用可能）その画像が Mask か SMask 項目を持つ場合には、images[ ] 擬似オブジェクト内におけるその画像のマスクの番号、そうでないなら –1。これを用いて、マスク画像を特定し、そのマスクの特性群を取得することができます。 ***mergetype*** （整数。TET 製品でのみ）以下の種別で画像のステータスを記述します： ***0*** （通常）画像は PDF 内の画像に対応しています。 ***1*** （擬似）画像は、複数の消費済画像（すなわち mergetype=2 の画像）を連結して 1 個の画像にした結果です。これでできる擬似画像は、PDF データ内にオブジェクトとしては存在しません。 ***2*** （消費済）画像は、連結されてより大きな画像にされているので、無視されるべきです。画像は PDF 内に存在していますが、擬似画像（すなわち mergetype=1 の画像）の一部分ですので、通常は抽出されるべきではありません。 この項目は、それまでに処理されたページすべてに関する情報を反映します。その値は、文書内の他のページが処理されるにつれて変わる可能性があります。最終（定常）情報が必要な場合は、文書内の全ページが処理されているか、または pCOS パス length:images の値が取得されている必要があります。 ***stencilmask*** （論理値。pCOS インタフェース 9。TET でのみ使用可能）その画像のステンシルマスクフラグ。これは、PDF キー ImageMask が存在する場合にはそれと同じですが、ただしそれとは異なり、得られることが保証されています。
patterns	（辞書の配列）ページ上の、または文書内のすべてのパターンに対する辞書を内容とする配列
properties	（辞書の配列）ページ上の、または文書内のすべてのプロパティに対する辞書を内容とする配列
shadings	（辞書の配列）ページ上の、または文書内のすべてのシェーディングに対する辞書を内容とする配列。シェーディング辞書内の標準 PDF キーのほか、以下の擬似キーも用いることができます： ***colorspaceid*** （整数）基礎色空間の、colorspaces[] 擬似オブジェクト内における番号。

表 4.9 リソースのための擬似オブジェクト一覧。各リソースカテゴリ P は 2 個のリソース配列 P[ ] と pages[ ]/P[ ] を生成します。

オブジェクト名	説明
templates	（辞書の配列）ページ上の、または文書内のすべてのテンプレート（Form XObject）に対する辞書を内容とする配列

4.9 保護された PDF 文書と pCOS モード

pCOS では、入力として暗号化・非暗号化 PDF 文書の両方に対応しています。しかし、暗号化文書から完全にオブジェクトを取得するには、その文書を開く際に正しいマスターパスワードを与える必要があります。ユーザー・マスターパスワードが得られるかどうかによって、暗号化文書は、以下に説明する pCOS モードのいずれかで処理することができます：

完全 pCOS モード（モード 2） 非暗号化文書は、つねに完全 pCOS モードで開かれます。暗号化された内容を持つ文書は、そのファイルを開く際にマスターパスワードが与えられれば、一切制約なく処理することができます。オブジェクトはすべて、暗号化されていない状態で返されます。

暗号化されていない文書が、暗号化されているファイル添付を含んでいるにもかかわらず、その添付のパスワードが与えなかったときは、下記の pCOS パス（すなわち添付内容）を取得すると、戻り値は空（C・C++ では NULL）になります：

```
pages[...]/annots[...]/FS/EF/F
names/EmbeddedFiles[...]/EF/F
```

限定 pCOS モード（モード 1） 文書が、正しいマスターパスワードなしで開かれており、かつ、ユーザーパスワードを必要としない場合には（あるいはユーザーパスワードだけが与えられている場合には）、**文字列**・*stream*・*fstream* 型のオブジェクトは取得できません。例外として、ページ内容の抽出が許可されている場合、すなわち *nocopy=false* の場合には、表 4.10 に挙げるオブジェクトも取得可能です。

表 4.10 限定 pCOS モードで、テキスト抽出許可すなわち nocopy=false の場合に取得可能なオブジェクト一覧

オブジェクト	pCOS パス
文書メタデータ[1]	/Root/Metadata（XMP メタデータ） /Root/Lang（pCOS インタフェース 8） /Info/*（文書情報フィールド）
しおり	bookmarks[...]/Title
注釈内容	pages[...]/annots[...] で始まるすべてのパス
文書レベルファイル添付（pCOS インタフェース 7）	names/EmbeddedFiles[...] で始まるすべてのパス

1. これらのオブジェクトは plainmetadata=true の場合にも取得できます。

最小 pCOS モード（モード 0） 暗号化ステータスと、パスワードが得られるかどうかにかかわらず、表 4.2・表 4.3・表 4.4 に挙げた普遍 pCOS 擬似オブジェクトはつねに利用可能です。たとえば *encrypt* 擬似オブジェクトを用いて、文書の暗号化ステータスをクエリすることができます。暗号化されたオブジェクトは、最初 pCOS モードでは取得できません。

pCOS 最小モードでは、*extensionlevel*・*fullpdfversion*・*pdfversion*・*pdfversionstring* 擬似オブジェクトによって報告されるバージョン情報から *ExtensionLevel* が欠落している場合があり（1.7ext3 でなく 1.7 と報告される等）、また、拡張レベルに関する情報を復号できないために、報告されるバージョンが低すぎるものとなる場合があります。

パスワード組み合わせのまとめ　表 4.11 に、保護された文書とさまざまなパスワード組み合わせに対する帰結 pCOS モードを挙げます。文書の暗号化ステータスと、そのファイルを開く際に与えられたパスワードによって、各種の PDF オブジェクトパスはそれぞれ、最小・限定・完全のいずれかの pCOS モードで利用可能です。各モードに対して不適切な pCOS パスを取得しようとすると例外が発生します。

表 4.11 さまざまなパスワード組み合わせと暗号化のある文書に対する帰結 pCOS モード一覧

知っているもの	pCOS の動作モード
いずれのパスワードも知らない	► ユーザーパスワードが必要な文書：最小 pCOS モード ► ユーザーパスワードが不要な文書：限定 pCOS モード ► 暗号化されたファイル添付を含んでいる文書：完全 pCOS モード、ただし添付は取得できません
ユーザーパスワード	限定 pCOS モード
マスターパスワード、あるいは暗号化されたファイル添付のある文書に対する添付パスワード	完全 pCOS モード

A pCOS 関数リファレンス

下記の表に、pCOS API 関数の概要を挙げます。詳細や、特定のプログラミング言語に対する情報については、おのおのの製品マニュアルを参照してください。

pCOS 関数プロトタイプ一覧

double pcos_get_number(int doc, String path)

String pcos_get_string(int doc, String path)

final byte[ ] pcos_get_stream(int doc, String optlist, String path)

B 改訂履歴

本マニュアルの改訂履歴

日付	変更点
2014 年 12 月 04 日	► PLOP/PLOP DS 5.0 内の pCOS インタフェース 9 に関する更新
2013 年 8 月 1 日	► pCOS 4.0 に同梱。内容に大きな変更なし
2013 年 5 月 16 日	► TET 4.2・TET PDF IFilter 4.2 に同梱。内容に大きな変更なし
2013 年 3 月 14 日	► PDFlib 9.0.0 に関する更新
2012 年 2 月 13 日	► pCOS インタフェース 8 へ更新
2011 年 3 月 04 日	► pCOS インタフェース 7 について PLOP 4.1 に言及
2010 年 11 月 29 日	► PDFlib 8.0.2 用 pCOS インタフェース 5 に関する再発行版
2010 年 10 月 29 日	► pCOS 3.0 内の pCOS インタフェース 7 に関する更新
2010 年 7 月 22 日	► 複数製品で用いるための pCOS インタフェース 6 に関するリファレンス再構成
2009 月 12 月 07 日	► PDFlib+PDI 8・PPS 8 内の pCOS インタフェース 5 に関する更新
2009 年 2 月 01 日	► PLOP 4.0・TET 3.0・TET PDF IFilter 3.0 内の pCOS インタフェース 4 に関する更新
2007 年 10 月 19 日	► pCOS 2.0 内の pCOS インタフェース 3 に関する更新
2006 年 3 月 28 日	► Perl 言語バインディングの説明を追加
2005 年 9 月 30 日	► pCOS 1.0 内の pCOS インタフェース 2 に関する版
2005 年 6 月 20 日	► TET 2.0 内の pCOS インタフェース 1 に関する版

索引

PDFlib GmbH

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

疑問がおありの際は、PDF メーリングリストと、
groups.yahoo.com/neo/groups/pdflib/info のアーカイブをチェックしてください

ライセンスに関するお問い合わせ
sales@pdflib.com

サポート
support@pdflib.com （お使いのライセンス番号をお書きください）