Antenna House PDF Tool API V8.0 ライブラリ説明書

索引

アンテナハウス株式会社

目次

第1章　Antenna House PDF Tool API について

豊富な機能

Antenna House PDF Tool API（PDF Tool API）は、PDF ファイルの加工処理、PDF のページ編集処理、PDF ファイルからの情報取得処理を支援するソフトウェアです。PDF ファイルの結合、ページ抽出、セキュリティ設定、透かしの挿入、しおりの作成・編集、テキスト抽出、テキスト検索などさまざま機能があります。

API で高度な開発

『PDF Tool API』では、PDF の各オブジェクトや機能をクラスライブラリ化しています。ライブラリに用意されている API を自在に組み合わせて独自の PDF 処理プログラムの作成が可能です。

API は、少ないコード行数でプログラムを書くことができるよう設計を工夫しています。高度な開発向きでありながら、PDF の仕様についてあまり深い知識がなくても使用できるようにしました。

対応するプログラム言語は、C++、.NET、Java です。

コマンドラインで手軽に実装

『PDF Tool API』のライブラリ版には、コマンドラインが同梱されています。

コマンドラインは、コマンドとオプションを指定して実行するしくみです。プログラム言語や PDF の仕様についての知識がほとんどなくても、PDF ファイルの情報取得や加工・編集処理ができます。

PHP、Phython、Ruby など、実行ファイルの呼び出しと制御ができるプログラム言語からも利用できます。

処理内容は限定されますが、PDF に対する処理として要望の多いコマンドとオプションをそろえています。開発工数の削減にもつながります。

第2章　動作環境

『PDF Tool API』の動作環境は以下の通りです。

2-1　Windows

OS の種類	Microsoft Windows 11 日本語版 Microsoft Windows Server 2025 / 2022 日本語版 注 ：ARM 版 Windows11 には対応していません。
必要なランタイムライブラリ	Microsoft Visual C++ 2022 ランタイムライブラリ（※1）

（※1）ランタイムライブラリの入手先：

『サポートされている最新の Visual C++再頒布可能パッケージのダウンロード』―Microsoft

「Visual Studio 2015、2017、2019、および 2022」項にある「アーキテクチャ」が「X64」の項目をダウンロードし、インストールしてください。

2-2　Linux

OS の種類	Linux X86 64bit（※2）
必要なランタイムライブラリ	glibc 2.34 libstdc++.so.6.0.29 もしくはこれらと互換のあるライブラリ 注 ：GCC11.4.1 でビルドされています。

（※2）Rocky Linux 9、Amazon Linux 2023 で動作確認を行っています。

2-3　仮想環境への対応

『PDF Tool API』を仮想環境（VMware や Hyper-V など）で実行する場合、実環境上と同じ動作が保証されているのであれば機能制限はありません。

第3章　対応プログラム言語

3-1　Windows

言語	対応バージョンなど
C++	Microsoft Visual C++2022 でビルドされています。MFC は使われていません。 互換性のあるコンパイラをご使用ください。Microsoft Visual Studio(Visual Studo) 2019 以前のバージョンの Visual Studio での開発は動作保証していません。
.NET	・.NET8 ・.NET Framework 4.8.x Microsoft Visual Studio 2022 でビルドされています。 互換性のあるコンパイラをご使用ください。Microsoft Visual Studio(Visual Studo) 2019 以前のバージョンの Visual Studio での開発は動作保証していません。 .NET8 用 API は、.NET6 以前の.NET ではコンパイルできません。 ※.NET インターフェースは C++ インターフェースのラッパーとして作成されています。
Java	・Java 25 ・Java 21 ・Java 17 ・Java 11 Eclipse Temurin JDK 11 でビルドされています。上記バージョンの Java でコンパイルを行ってください。Java 11 より前のバージョンの Java ではコンパイルできません。 ※Java インターフェースは C++ インターフェースのラッパーとして作成されています。
コマンドライン	C++ インターフェースを使用して作成されたコンソール型アプリケーション

3-2　Linux

言語	対応バージョンなど

C++	GCC 11.4 でビルドされています。
.NET	・.NET8 .NET8 用 API は、.NET6 以前の.NET ではコンパイルできません。 ※.NET インターフェースは C++ インターフェースのラッパーとして作成されています。
Java	・Java 25 ・Java 21 ・Java 17 ・Java 11 Eclipse Temurin JDK 11 でビルドされています。上記バージョンの Java でコンパイルを行ってください。Java 11 より前のバージョンの Java ではコンパイルできません。 ※Java インターフェースは C++ インターフェースのラッパーとして作成されています。
コマンドライン	C++ インターフェースを使用して作成されたコンソール型アプリケーション

第4章　インストール／アンインストール

本章ではインストールフォルダの構成やインストーラにより設定される環境変数などの情報を解説します。

本書における「フォルダ」表記について

本マニュアルの本項以降ではフォルダ／ディレクトリのことを「フォルダ」と総称して記述します。たとえば、フォルダ／ディレクトリパスのことは「フォルダパス」と表記します。

Linux で操作される方は以下項目では「フォルダ」を「ディレクトリ」と読み替えてお読みください。

4-1　　Windows 版

4-1-1 インストール方法

1. [AHPDFToolLib_V80_***_x64.exe]がインストーラです。（ファイルをダブルクリックするなどして起動してください）
   「***」には R1、MR1 などの改訂バージョン名が入ります。
2. [AHPDFToolLib_V80_***_x64.exe]のダイアログにしたがってインストールを実行します。
   デフォルトのインストール先フォルダパスは次の場所です。

   {システムドライブ}:¥AHPDFToolLib_80

   新規インストールが完了すると、インストール後 30 日間有効な評価版ライセンスが配置されます。
   評価版ライセンスの期限切れ、もしくはライセンス正規版に入れ替えるまでコマンドラインアプリケーションは評価版として動作します。評価版における制限などは『第 12 章_評価版の仕様について』を参照してください。

補足：Windows 版の上書きインストールについて

Windows 版では上書きインストールが可能です。改訂版の『PDF Tool API』をインストールする際は旧バージョンが残ったままインストーラを実行することができます。

上書きインストールを実行した場合の各ファイルは以下のように置換されます。

- 実行バイナリ・ファイル：日付とファイルバージョンを確認して新しいものに置換

- 非実行ファイル（ヘッダファイルなど）：日付を確認して新しいものに置換
- ライセンスファイル：置換しない（入替えの必要がある場合は手動での置換が必要）

4-1-2 アンインストール方法

1. 「コントロールパネル-プログラム-プログラムと機能」を開く
2. プログラム一覧にある「Antenna House PDF Tool API V8.0 Windows ライブラリ」を選択
3. 「アンインストール」をクリックしてアンインストールを実行

4-1-3 Microsoft Visual C++ランタイムライブラリについて

『PDF Tool API V8.0』の動作には、Microsoft Visual C++ 2022 ランタイムライブラリが必要です。

インストール時に動作環境にランタイムライブラリが存在しないと判定したとき、自動的に「Microsoft Visual C++ 2022 再頒布パッケージ」のインストーラが起動します。

再頒布パッケージのインストーラが起動した場合は、表示されるダイアログにしたがってインストールしてください。

4-1-4 インストールフォルダの構成

インストールフォルダ
|-- bin：ライブラリモジュール：コマンドラインアプリケーションを含みます
|-- CopyRightFiles：第三者ライブラリの著作権情報
|-- Include：ヘッダファイル
|-- fontconfig：フォント構築ファイル
|-- icc：カラープロファイル用 icc ファイル格納フォルダ
|-- lib：lib ファイル
|-- License：ライセンスファイル
|-- EULA.txt：使用許諾契約書

4-1-5 インストーラによりシステムに設定される内容

インストーラが値を設定する環境変数

インストール時、以下の環境変数について、インストーラにより値が設定されます。環境変数が無かった場合は環境変数が作成されます。

- 環境変数名　：PTL80_LIC_PATH
  設定値　　：{インストールフォルダ}¥License

- 環境変数名　：PTL80_FONT_CONFIGFILE
  設定値　　：{インストールフォルダ}¥fontconfig¥font-config.xml

- 環境変数名　：PTL80_ICCPROFILE_PATH
  設定値　　：{インストールフォルダ}¥icc

ダイアログで指定時にインストーラが値を設定する環境変数

以下の環境変数は、インストーラ実行時にダイアログで「環境変数に追加する」を指定した場合にインストーラにより値が設定・追加されます。

- 環境変数名　：PATH
  設定値　　：「{インストールフォルダ}¥bin」を追加する

[PTL80_LIC_PATH]に関しては『第 5 章 ライセンスファイルについて』を参照してください。

[PTL80_FONT_CONFIGFILE]に関しては『7-5-2 描画とフォント埋め込みに使用するフォントの参照先について』を参照してください。

[PTL80_ICCPROFILE_PATH]に関しては『7-11 カラープロファイルの扱い』を参照してください。

4-1-6 インストールされるライセンスファイルについて

インストールされるライセンスファイルに関しては以下の注意事項があります。

- インストーラによりインストールされるライセンスファイル「ptalic.dat」は、インストール後 30 日間有効の評価版用ライセンスです。
- 期限を過ぎると、『PDF Tool API』は利用不可になります。
- 評価版ライセンスで実行される『PDF Tool API』では、出力される PDF ファイルの各ページに透かし文字列が挿入されます。
- その他、評価版における制限などは『第 12 章 評価版の仕様について』を参照してください。

4-2　Linux 版

4-2-1 インストール方法

1. スーパーユーザーでログインします。
2. rpm コマンドを実行します。

```
rpm -i AHPDFToolLib80-8.0-***.x86_64.rpm [--prefix インストールパス]
```

3. インストールパスを指定しなかった場合、「/usr/AHPDFToolLib80」にインストールされます。
4. PDF Tool API は、インストール後 30 日間有効な評価版としてインストールされます。評価版における制限などは『第 12 章 評価版の仕様について』を参照してください。

Linx 版は上書きインストールできない

Linux 版では『PDF Tool API』を上書きインストールすることができません。そのため、再インストールをする際や改訂版の『PDF Tool API』をインストールする際は、その前に古い『PDF Tool API』をアンインストールしてください。

4-2-2 アンインストール方法

1. スーパーユーザーでログインします。
2. rpm コマンドを実行します。

```
rpm -e AHPDFToolLib80
```

3. シェルスクリプトなどで環境変数[PTL80_LIC_PATH]、[PTL80_FONT_CONFIGFILE]、[PTL80_ICCPROFILE_PATH]を設定している場合は元に戻します。その他の環境変数を設定していた場合も同様に記述を削除するなどしてください。

[PTL80_LIC_PATH]に関しては『第 5 章 ライセンスファイルについて』を参照してください。

[PTL80_FONT_CONFIGFILE]に関しては『7-5-2 描画とフォント埋め込みに使用するフォントの参照先について』を参照してください。

[PTL80_ICCPROFILE_PATH]に関しては『7-11 カラープロファイルの扱い』を参照してください。

4-2-3 インストールフォルダの構成

インストールフォルダ

|-- CopyRightFiles：第三者製ライブラリの著作権情報ファイル

|-- fontconfig：フォント構築ファイル

|-- icc：カラープロファイル用 icc ファイル格納フォルダ

|-- Include：ヘッダファイル

|-- lib：PDF Tool API モジュール

|-- License：ライセンスファイル

|-- ToolCmd：コマンドラインアプリケーション実行ファイル

|-- EULA.txt：使用許諾契約書

|-- run.sh：コマンドラインアプリケーション実行スクリプト

4-2-4 インストールされるライセンスファイルについて

インストールされるライセンスファイルに関しては以下の注意事項があります。

- インストーラによりインストールされるライセンスファイル「ptalic.dat」は、30 日間有効の評価版用ライセンスです。Linux 版の場合、最初に『PDF Tool API』を使用した日から 30 日間使用可能です。
- 期限を過ぎると、『PDF Tool API』は利用不可になります。
- 評価版ライセンスで実行される『PDF Tool API』では、出力される PDF ファイルの各ページに透かし文字列が挿入されます。
- その他、評価版における制限などは『第 12 章 評価版の仕様について』を参照してください。

第5章　ライセンスファイルについて

『PDF Tool API』を実行する際にはライセンスファイルが必要です。

ライセンスファイルは[ptalic.dat]の名前を持つファイルです。『PDF Tool API』が参照した先にライセンスファイルが存在しない場合、『PDF Tool API』はエラーを出力して動作を停止します。
動作停止の際のエラーコードに関しては『15-2 エラーコード一覧』を参照してください。

本章ではライセンスファイルの仕様・入手・参照先指定・入れ替えについて説明します。

5-1　　ライセンスファイルの区分と仕様

ライセンスファイルは、評価版ライセンスと正規版ライセンスの 2 種類に大別されます。
それぞれの仕様は以下の通りです。

5-1-1 評価版ライセンス

30 日の期限付きの評価用のライセンスです。
インストーラにより、インストールフォルダ内に配置されます。透かしが入るなど、評価版の動作には制限事項があります。
評価版の制限事項についての詳細は『第 12 章 評価版の仕様について』を参照してください

評価版ライセンスの正確な動作期限は、以下の通りです。

- Windows 版　：インストール日から 30 日
- Linux 版　　：最初に利用した日から 30 日

インストーラによって配置される評価版ライセンスファイル[ptalic.dat]のフォルダパスは以下です。

ライセンスファイルの配置先：{インストールフォルダ}¥License

5-1-2 正規版ライセンス

『PDF Tool API』を正規購入した際に得られるライセンスです。ファイル名は評価版と同じく[ptalic.dat]です。
正規版ライセンスには動作期限はないので無期限に使用できます。ライセンスファイルには保守期限が設けられています。保守契約期限内であれば製品をアップデートすることができます。

- 保守期限に関する詳細は『5-6 保守契約期限と改訂版アップデートにおける『PDF Tool API』の動作について』を参照してください。
- 『PDF Tool API』がメジャーバージョンアップ及びマイナーバージョンアップをする場合はライセンスファイルの入れ替えが必要です。詳細は『13-1 保守契約期限内のバージョンアップについて』を参照してください。
  「メジャーバージョンアップ」及び「マイナーバージョンアップ」の定義については『第 13 章 バージョンアップについて』を参照してください。

注意：

- 1 つの正規版ライセンスで運用可能な『PDF Tool API』のバージョンは 1 つだけです。詳細は『13-2 バージョンアップとライセンスに関する制限事項』を参照してください。

5-2 ライセンスファイルの入手

- 『PDF Tool API』を新規にご購入の場合、正規版のライセンスファイルはインストーラとは別途で弊社よりお送りします。
- 保守契約を更新された際には、その都度弊社より新しいライセンスファイルを送付いたします。
- バージョンアップをご希望の場合、該当バージョン用のライセンスファイルが必要です。その他バージョンアップ手続きに関する詳細は『13-1 保守契約期限内のバージョンアップについて』を参照してください。

5-3 ライセンスファイルの参照先指定

『PDF Tool API』は主に環境変数[PTL80_LIC_PATH]を用いてライセンスファイルを参照します。
この参照先の指定方法は大きく 3 つあり、以下の方法となります。

- インストーラによる参照先の指定：
  これが最も簡単な方法です。詳細は『5-3-1 インストーラによる参照先指定』を参照してください。
- 環境変数による任意の位置の指定：
  インストールフォルダ以外の位置を指定したい場合に用います。詳細は『5-3-2 任意の参照先指定』を参照してください。
- 環境変数を用いないライセンスファイル参照（Windows 版のみ）
  モジュールファイルと同じフォルダにライセンスファイルを配置することで環境変数を用いずに参照することができます。詳細は『5-3-3 環境変数を用いない参照先（Windows 版のみ)』を参照してください。

正規版ライセンスを購入した場合などはライセンスファイルの更新が必要です。『5-4 ライセンスファイルの入れ替えについて』を参照して参照先のライセンスファイルを入れ替えてください。

5-3-1 インストーラによる参照先指定

インストーラを用いてインストールした場合、評価版のライセンスファイル[ptalic.dat]はインストーラによってインストールフォルダ内に配置されます。
評価版ライセンスファイルが配置されるのは、具体的には以下のフォルダパスです。

ライセンスファイルの配置先：{インストールフォルダ}¥License

Windows 版の場合はインストーラ実行時に環境変数[PTL80_LIC_PATH]も同時に設定されます。設定される内容に関する詳細は『4-1-5 インストーラによりシステムに設定される内容』を参照してください。
そのため、例えばインストール後すぐに評価版ライセンスを使いたい場合はコマンドラインアプリケーションをそのまま使い始めることが可能です。

Linux 版では Windows 版とは異なりインストーラが環境変数の設定を行いません。
その代わりに、インストール先情報が設定されたファイル[run.sh]がインストーラによって配置されます。

[run.sh]はインストール先情報を持った環境変数を設定した上でコマンドラインアプリケーションを実行するシェルスクリプトです。詳細はコマンドライン説明書に記載の『シェルスクリプト[run.sh]について』を参照してください。

[run.sh]を使用しない場合は後記の『5-3-2 任意の参照先指定』を参考に、ライセンスファイルの配置先を環境変数[PTL80_LIC_PATH]に指定してください。

5-3-2 任意の参照先指定

任意の位置にあるライセンスファイルを参照したい場合、環境変数[PTL80_LIC_PATH]の値を指定する必要があります。
具体的な手順は以下の方法です。

1. ライセンスファイル[ptalic.dat]を任意の場所に配置します。
2. 配置したフォルダパスを下記の環境変数に設定します。

- 環境変数名：PTL80_LIC_PATH
  設定値：ライセンスファイルを配置したフォルダパス

5-3-3 環境変数を用いない参照先（Windows 版のみ）

Windows 版の場合は、『PDF Tool API V8.0』のモジュールファイル[PdfTk80.dll]と同一の場所にライセンスファイルを配置して使用することも可能です。
このとき、環境変数「PTL80_LIC_PATH」は必要ありません。

5-4 ライセンスファイルの入れ替えについて

新規ご購入や保守契約の延長に伴い保守窓口から入手したライセンスファイルを有効にするには、古いライセンスファイルを新しいライセンスファイルに入れ替える必要があります。
入れ替え方法は以下のいずれかの方法から選択できます。

- 古いファイルを新しいライセンスファイルで上書きする方法：
  これが最も簡単な方法です。（各ライセンスファイルは[ptalic.dat]で同一名です）
- 任意の位置に新しいライセンスファイルを配置する方法。
  この場合は、環境変数で指定するフォルダパスを新しいものに変更してください。
  指定する環境変数に関する詳細は『5-3-2 任意の参照先指定』を参照してください。
  この場合、古いライセンスファイルは元のフォルダに残されます。

補足事項：

- ライセンスファイルの入れ替えは、『PDF Tool API』を利用するプログラムが実行されていないときに行ってください。
- ライセンスファイルの入れ替えは、ライセンスファイルの入れ替えた次の実行時から反映されます。マシンの再起動の必要は

ありません。

5-5　保守契約期限内における改訂版アップデートとバージョンアップについて

新規ご購入頂いたライセンスファイル及び保守契約を延長されたライセンスファイルには、保守契約期間情報が記入されています[*1]。
保守契約期間内に改訂版や新バージョンがリリースされた場合、改訂版へのアップデートやバージョンアップを行うことができます。改訂版へのアップデートとバージョン変更を伴うバージョンアップでは、ライセンスファイルの扱いに違いがあります。

- 保守契約期限内にリリースされた『PDF Tool API』の改訂版[*2]へアップデートする場合、現在お使いのライセンスファイルをご利用いただけます。詳細は『14-1 保守契約期限内の改訂版へのアップデートについて』を参照してください。
- 保守契約期限内に『PDF Tool API』の新バージョンがリリースされた場合のバージョンアップ[*3]に関しては、該当バージョンに合わせたライセンスファイルの入手が必要です。バージョンアップに関する製品仕様などの詳細は『13-1 保守契約期限内のバージョンアップについて』を参照してください。
- 改訂版へのアップデートやバージョンアップは必須ではありません。古いバージョンのままの使用も可能です。

[*1]：
ライセンスファイルの保守契約期間はライセンスファイルから読み取ることが可能です。具体的な操作については『5-7 ライセンスファイルの情報確認について』を参照してください。
[*2]：
「改訂版」の定義については『第 14 章 改訂版について』を参照してください。
[*3]：
バージョンアップの定義や詳細については『第 13 章 バージョンアップについて』を参照してください。

5-6　保守契約期限と改訂版アップデートにおける『PDF Tool API』の動作について

改訂版へのアップデートにおいては、ライセンスファイルの入れ替えは必要ありません。現在ご利用中のライセンスファイルをそ

のままご利用頂けます。
しかし、アップデート先の改訂版が保守契約期間内にリリースされたか否かで『PDF Tool API』の動作が異なります。

- 保守契約期限内にリリースされた『PDF Tool API』の改訂版に関してはアップデートしてそのままご利用いただけます。
- 保守契約期限後にリリースされた改訂版にアップデートした場合、『PDF Tool API』は評価版[*1]として動作します。評価版では出力される評価用の透かしが挿入されるなどの制限事項が発生します。
  最新の改訂版を正規利用するには、保守契約の更新をご検討ください。
- 保守契約期限後も『PDF Tool API』をアップデートしない場合は引き続き正規版としてご利用いただけます。

[*1]：
評価版の制限事項についての詳細は『第 12 章 評価版の仕様について』を参照してください。

5-7　ライセンスファイルの情報確認について

同梱のコマンドラインアプリケーションを利用して、ライセンスファイル情報と使用中の『PDF Tool API』のバージョン確認ができます。指定したファイルパスのライセンスファイルの情報を取得することも可能です。

コマンドラインアプリケーションの使用方法やコマンドの詳細な仕様に関しては、『PDF Tool API コマンドラインマニュアル』及び該当コマンド「-v」の説明を参照してください。

コマンドラインアプリケーションと同じ場所、あるいは環境変数「PTL80_LIC_PATH」にあるライセンスファイルの情報と、『PDF Tool API』のバージョン確認を行う場合

Windows 版

```
AHPDFToolCmd80.exe -v
```

Linux 版

```
AHPDFToolCmd80 -v
```

指定したファイルパスにあるライセンスファイルの情報と、『PDF Tool API』のバージョン確認を行う場合

Windows 版

```
AHPDFToolCmd80.exe -v C:¥ahlic¥ptalic.dat
```

Linux 版

```
AHPDFToolCmd80 -v /home/ahlic/ptalic.dat
```

第6章　モジュールファイルについて

6-1　Windows 版

API の各プログラム言語のメインモジュールファイル名とこれと依存関係にあるファイル名、コマンドラインの実行ファイル名とこれと依存関係にあるファイル名は以下の通りです。

『PDF Tool API』を使用して作成したプログラムの実行環境には、プログラムの実行ファイルとともにこれらのモジュールファイルが必要です。

『PDF Tool API』のインターフェース	メインモジュール	依存関係にあるファイル
C++	PdfTk80.dll	PdfTkEx80.dll PtkAHCommon80.dll PtkAHDMC80.dll PtkAHFontService80.dll PtkAHGraphicService80.dll PtkAHPDFEditLib80.dll PtkAHPDFLib80.dll PtkPDFLinearizer80.dll PtkAHCertificate80.dll PtkAHPDFFixUp80.dll icudt72.dll icuin72.dll icuio72.dll icutu72.dll icuuc72.dll
.NET Framekwork	PdfTkNet80.dll	PdfTk80.dll PdfTkEx80.dll PtkAHCommon80.dll PtkAHDMC80.dll PtkAHFontService80.dll PtkAHGraphicService80.dll PtkAHPDFEditLib80.dll PtkAHPDFLib80.dll PtkPDFLinearizer80.dll

		PtkAHCertificate80.dll PtkAHPDFFixUp80.dll icudt72.dll icuin72.dll icuio72.dll icutu72.dll icuuc72.dll
Java	PdfTkJava80.jar PdfTk8JNI.dll	PdfTk80.dll PdfTkEx80.dll PtkAHCommon80.dll PtkAHDMC80.dll PtkAHFontService80.dll PtkAHGraphicService80.dll PtkAHPDFEditLib80.dll PtkAHPDFLib80.dll PtkPDFLinearizer80.dll PtkAHCertificate80.dll PtkAHPDFFixUp80.dll icudt72.dll icuin72.dll icuio72.dll icutu72.dll icuuc72.dll
コマンドライン	AHPDFToolCmd80.exe	PdfTk80.dll PdfTkEx80.dll PtkAHCommon80.dll PtkAHDMC80.dll PtkAHFontService80.dll PtkAHGraphicService80.dll PtkAHPDFEditLib80.dll PtkAHPDFLib80.dll PtkPDFLinearizer80.dll PtkAHCertificate80.dll PtkAHPDFFixUp80.dll icudt72.dll icuin72.dll icuio72.dll icutu72.dll icuuc72.dll

.NET8	PdfTkNet8_80.dll PdfTkNet8.dll	PdfTk80.dll PdfTkEx80.dll PtkAHCommon80.dll PtkAHDMC80.dll PtkAHFontService80.dll PtkAHGraphicService80.dll PtkAHPDFEditLib80.dll PtkAHPDFLib80.dll PtkPDFLinearizer80.dll PtkAHCertificate80.dll PtkAHPDFFixUp80.dll icudt72.dll icuin72.dll icuio72.dll icutu72.dll icuuc72.dll

※.Net Framework 4.8 入手先：

実行環境には『.NET Framework 4.8 Web Installer』をダウンロードしてセットアップしてください。

※.NET8 Runtime 入手先：.NET8 のダウンロード

実行環境には『.NET Runtime 8.0.xx』をダウンロードしてセットアップしてください。

6-2　Linux 版

API の各プログラム言語のメインモジュールファイル名とこれと依存関係にあるファイル名、コマンドラインの実行ファイル名とこれと依存関係にあるファイル名は以下の通りです。

『PDF Tool API』を使用して作成したプログラムの実行環境には、プログラムの実行ファイルとともにこれらのモジュールファイルが必要です。

『PDF Tool API』のインターフェース	メインモジュール	依存関係にあるファイル
C++	libPdfTk.so.8.0	libPdfTkEx.so.8.0 libPtkAHCertificate.so.8.0 libPtkAHCommon.so.8.0 libPtkAHDMC.so.8.0 libPtkAHFontService.so.8.0 libPtkAHGraphicService.so.8.0 libPtkAHPDFEditLib.so.8.0 libPtkAHPDFFixUpLib.so.8.0 libPtkAHPDFLib.so.8.0 libPtkPDFLinearizer.so.8.0 libicudata.so.72.1 libicui18n.so.72.1 libicuio.so.72.1 libicutu.so.72.1 libicuuc.so.72.1
Java	PdfTkJava80.jar libPdfTk8JNI.so.8.0	libPdfTk.so.8.0 libPdfTkEx.so.8.0 libPtkAHCertificate.so.8.0 libPtkAHCommon.so.8.0 libPtkAHDMC.so.8.0 libPtkAHFontService.so.8.0 libPtkAHGraphicService.so.8.0 libPtkAHPDFEditLib.so.8.0 libPtkAHPDFFixUpLib.so.8.0 libPtkAHPDFLib.so.8.0 libPtkPDFLinearizer.so.8.0 libicudata.so.72.1 libicui18n.so.72.1 libicuio.so.72.1 libicutu.so.72.1 libicuuc.so.72.1
コマンドライン	AHPDFToolCmd80	libPdfTk.so.8.0 libPdfTkEx.so.8.0 libPtkAHCertificate.so.8.0 libPtkAHCommon.so.8.0 libPtkAHDMC.so.8.0 libPtkAHFontService.so.8.0

		libPtkAHGraphicService.so.8.0 libPtkAHPDFEditLib.so.8.0 libPtkAHPDFFixUpLib.so.8.0 libPtkAHPDFLib.so.8.0 libPtkPDFLinearizer.so.8.0 libicudata.so.72.1 libicui18n.so.72.1 libicuio.so.72.1 libicutu.so.72.1 libicuuc.so.72.1
.NET8	PdfTkNet8_80.dll libPdfTkNet8.so.8.0	libPdfTk.so.8.0 libPdfTkEx.so.8.0 libPtkAHCertificate.so.8.0 libPtkAHCommon.so.8.0 libPtkAHDMC.so.8.0 libPtkAHFontService.so.8.0 libPtkAHGraphicService.so.8.0 libPtkAHPDFEditLib.so.8.0 libPtkAHPDFFixUpLib.so.8.0 libPtkAHPDFLib.so.8.0 libPtkPDFLinearizer.so.8.0 libicudata.so.72.1 libicui18n.so.72.1 libicuio.so.72.1 libicutu.so.72.1 libicuuc.so.72.1

※.NET8 Runtime 入手先：.NET8 のダウンロード

実行環境には『.NET Runtime 8.0.xx』をダウンロードしてセットアップしてください。

モジュールファイルの参照について

- 環境変数「LD_LIBRARY_PATH」に「{インストールフォルダ}/lib」を設定してください。
- Java 版では、環境変数「CLASSPATH」に「{インストールフォルダ}/bin/ PdfTkJava80.jar」を設定してください。

第7章　製品仕様

7-1　『PDF Tool API』使用の既存プログラムの V8.0 へのバージョンアップ方法

『PDF Tool API V8.0』は、『PDF Tool API V4.0 / V5.0 / V6.0 / V7.0』を継承しています。『V4.0 / V5.0 / V6.0 / V7.0』を使用しているプログラムは、ヘッダや lib ファイル、あるいは参照するバイナリファイル を『V8.0』のものに置き換えてリビルドすることにより、『V8.0』用のプログラムとして実行可能となります。

ただし、一部の API では名前が変更になっているものがあります。その場合は、V8.0 で修正された名称に変更してください。

変更箇所の詳細については『第 11 章 仕様変更について』を参照してください。

7-2　使用に際する注意点

- 『PDF Tool API』は、複数のスレッドでの同時利用はできません。
- メインモジュールファイルとこれと依存関係にあるモジュールファイルは、必ず、同一の改訂版をご使用ください。メインモジュールと依存関係モジュールが異なる改訂版であるファイルや、他の製品でリリースされたファイルを依存関係モジュールとして使用しないでください。

7-3　処理対象 PDF ファイルと表示あるいは描画について

- 『PDF Tool API』の処理対象は、PDF バージョン「1.7」までと、「2.0」のファイルです。
- 入力 PDF がどのようなソフトウェアから生成されたものかは特に限定していません。
- 『PDF Tool API』の開発は、「ISO 32000-1」および「ISO 32000-2」の仕様を基準に行っています。

- 『PDF Tool API』では、出力 PDF のバージョン番号を任意に指定することはできません。

7-3-1 PDF2.0 対応について

- PDF2.0 の PDF ファイルを入力ファイルとして指定できます。この場合に限り、出力 PDF の PDF バージョンが「2.0」となります。
- PDF2.0 のファイルに対するセキュリティ設定は PDF2.0 仕様のものとなります。つまり、セキュリティを施した場合、AES256bit、「R(リビジョン)6」の暗号化処理指定が必要です。これ以外のレベルの暗号化処理を施そうとした場合、エラーとなります。
- PDF2.0 のファイルとその他の PDF バージョンのファイルとの結合処理が可能です。この場合、出力結果ファイルの PDF バージョンは PDF2.0 となります。ただし、PDF2.0 以外の PDF バージョンファイルの部分は元の構成のままです。つまり『PDF Tool API』が他の PDF バージョンのファイル部分を PDF2.0 仕様にそったものに変更することはありません。結合とともにその他の編集処理を行った場合についても、内容を PDF2.0 仕様に変更することはありません。
- PDF 透かしの入力ファイルとして PDF2.0 のファイルを指定することができます。出力 PDF は、入力 PDF のバージョンとなります。

7-3-2 PDF の表示あるいは描画について

PDF ビューワ上で問題なく表示できるファイルであっても、『PDF Tool API』上で情報の取得や編集の処理をする際に問題が発生する場合があります。

PDF には ISO によって定められた仕様が存在します。しかし同時に、PDF は作り方の自由度が高く、カスタマイズできる部分が多くあります。したがって、PDF はその生成するソフトウェアにより、ファイル内部の作り方がさまざまです。このため、PDF ファイルに対して行う処理によっては問題現象が発生する場合があります。

PDF ファイルに対して行われる処理は、以下に大別されます。

- PDF を表示すること
- PDF から情報を取得すること
- PDF を編集すること

上記の一部の処理で問題が無かったファイルでも、別の処理をしようとすると問題が発生することがあります。そして、『PDF Tool API』では、上記のうち「PDF から情報を取得すること」、「PDF を編集すること」を扱います。

具体的には、表示上問題なかったファイルでも情報の取得や編集の処理においては問題が発生する場合があります。

もしくは、情報の取得はできるが、PDF の表示あるいは描画が正しく行われない問題が発生する場合があります。例えば、ある PDF ファイルに対して文字列や画像の挿入などの編集処理を行うと、既存の線や図がずれる、テキストが表示されないあるいは反転してしまう場合があります。

『PDF Tool API』では確認済みの現象への対策は行っていますが、処理する PDF ファイルによって未知の問題が発生する可能性は排除できません。

7-4　文書情報の Metadata について

文書情報設定時、入力ファイルに Metadata が存在しない場合、『PDF Tool API』は Metadata を新たに作成します。

例えば、『PDF Tool API』では、保存処理時に文書情報の「更新日時」を設定します。そのため、『PDF Tool API』で保存処理を行うと自動的に Metadata が作られることになります。

7-5　フォントについて

『PDF Tool API』で扱うフォントの仕様について説明します。

7-5-1 『PDF Tool API』におけるフォントを扱う機能

『PDF Tool API』でフォントを扱う機能には、以下の種類があります。

- PDF に文字列を描画する
- PDF 上に存在するフォントを埋め込み状態にする
- PDF 上に存在するフォントを統合する
- PDF 上のフォント情報を取得する

7-5-2 描画とフォント埋め込みに使用するフォントの参照先について

文字列の描画処理とフォントの埋め込み処理では、動作環境に存在するフォント使用します。参照先は OS により異なります。

[フォントの参照先]

Windows

動作環境のフォントフォルダに存在するフォントを使用します。

フォントフォルダは、「{システムドライブ}:¥WINDOWS¥Fonts」です。

フォントフォルダとは別の場所にあるフォントを使用する場合は、「フォント構築ファイル」を設定します。

Linux(X86)

使用するフォントを「フォント構築ファイル」に設定します。

フォントはあらかじめ動作環境にインストールしてください。

[フォント構築ファイルの設定]

フォント情報を特定のフォルダから取得する場合、フォルダパスをフォント構築ファイル内に記入・設定する必要があります。

フォント構築ファイルは「font-config.xml」の名前を持つ XML ファイルです。Windows 版、Linux 版を問わず、インストール時にインストーラによって以下のパスに配置されます。

{インストールフォルダ}¥fontconfig¥font-config.xml

フォントファイルが存在するフォルダパスを、「font-config.xml」内で「font-folder path」タグに記述します。

font-config.xml 記述例（Windows で「C:¥TestFont」フォルダを指定したい場合）

```
<font-config>
  <font-folder path="C:¥TestFont"></font-folder>
</ font-config>
```

[環境変数の作成]

フォント構築ファイルの設定を反映させるためには、環境変数を作成し値を設定します。

- 環境変数名：PTL80_FONT_CONFIGFILE
  設定する値：font-config.xml のフルパス

7-5-3 PDF に文字列を描画する

PDF のページコンテントに文字列を描画したり、テキスト透かしを挿入する処理を行う場合、動作環境に存在するフォント情報を取得しこれを利用して処理を行います。

フォント情報取得のために参照する場所などについては、『7-5-2 描画とフォント埋め込みに使用するフォントの参照先について』を参照してください。

- 以下の種類のフォントを用いて描画処理が可能です。
  - TrueType フォント（Unicode cmap を持つもの）
  - OpenType フォント（Unicode cmap を持つもの）
  - Type1 フォント
- ビットマップフォント、Type3 フォントでの描画には対応していません。
- 縦書き処理の場合、イタリック（斜体）の指定があってもフォントがグリフを持っていない場合はイタリックでは描画されません。

7-5-4 PDF 上に存在するフォントを埋め込み状態にする

PDF 上に存在するフォントについて、動作環境に存在するフォント情報を使用して PDF に埋め込みます。

使用するフォント情報の参照先については、『7-5-2 描画とフォント埋め込みに使用するフォントの参照先について』を参照してください。

ただし、次のフォントについては、埋め込み指示があってもフォント情報は埋め込まれません。

- 動作環境に存在しないフォント
- TrueType：フォント埋め込みが禁止されたフォント

- Type1：PFB ファイルが存在しないフォント

7-5-5 PDF 上に存在するフォントを統合する

- フォント統合は、PDF に埋め込まれているフォントに対して行われます。
- 次のフォントが統合対象です。
  - TrueType
  - Type1(CFF 形式のみ)
  - CIDFontType2
  - CIDFontType0
- 統合対象は、埋め込まれたフォントファイルとフォント辞書です。
- フォント辞書の統合について、
  - TrueType、Type1(CFF 形式のみ)の場合、FontDescriptor 辞書が統合されます。
  - CIDFontType2、CIDFontType0 はの場合、CIDFont 辞書が統合されます。
- ページと FormXObject のフォントが統合対象です。

フォント統合における例外について

- 注釈とフォームフィールドのフォントは統合の対象となりません。
- 同じ文字で描画命令が違うものは統合の対象となりません。
- 重複するフォント情報をひとつに統合しますが、統合により必ずファイルサイズが小さくなるわけではありません。
- 埋め込まれたフォントファイルの内容をチェックし以下の条件の場合は統合されません。
  - TrueType フォントの場合、cmap、head,hhea,hmtx,maxp,cvt,fpgm,prep の各テーブルで違うものがあれば統合の対象となりません。
  - CIDFontType2 フォントの場合、head,hhea,hmtx,maxp,cvt,fpgm,prep の各テーブルで違うものがあれば統合の対象となりません。
  - Type1(CFF 形式のみ)、CIDFontType0 において、メジャーバージョン、マイナーバージョンが異なる場合、統合の対象となりません。
- フォント辞書の内容をチェックし以下の条件の場合は統合されません。
  - CIDFontType2、CIDFontType0 の場合、
    - 「CIDSystemInfo」辞書の「Registry」キー（文字コレクションの発行者を識別する文字列)、「Ordering」キー（「Registry」の文字コレクションに一意の名前を付ける文字列）が違うものは統合対象外
    - 「DW」キー（グリフのデフォルトの幅）が違うものは統合対象外
    - 「DW2」キー（縦書き用のグリフのメトリック）が違うものは統合対象外
    - 「CIDToGIDMap」キーが「Identity」という値以外の名前、もしくは型がストリームの場合は統合対象外
    - 同じ文字で「W」キー（「DW」の説明）の値が違うものがある場合は統合対象外
    - 同じ文字で「W2」キー（「DW2」の説明）の値が違うもがある場合は統合対象外

- ✧ 「Weight」キー（太さ）が違う場合は統合対象外

- TrueType、Type1(CFF 形式のみ)の場合、
  - ✧ 「Weight」キー（太さ）が違う場合は統合対象外

7-6 原点・座標・単位

7-6-1 原点

- デフォルトの座標の原点は、ページ表示上の「左下」です。
- PtlOption クラスを利用して、原点を「左上」に切り替えて処理を行うことができます。

7-6-2 座標

- 『PDF Tool API』では、デフォルトは表示上の座標を扱います。『PDF Tool API』 を利用する側においては、PDF ページの加工や編集処理のさいに元のページが回転されているかどうかを考慮する必要はありません。原点を「左下」としている場合、どのページにおいても表示上の「左下」が原点となります。
- PtlOption クラスを利用して、PDF のユーザースペース座標に切り替えて扱うことができます。ユーザースペース座標では、ページが回転していると座標軸も回転します。たとえば、回転しているページに文字を挿入した場合、文字も回転します。

7-6-3 単位

- デフォルトは、「mm」（ミリ）です。
- PtlOption クラスを利用して、「PT」（ポイント）単位に切り替えて処理を行うことができます。

7-7　読み書きの制限サイズ

読み書き可能な PDF ファイルサイズは 2GB までです。

- 入力ファイルとして読み込み（load）可能な PDF ファイルのファイルサイズは 2GB までです。2GB を超える PDF ファイルの場合は load 時にエラーになります。
- 『PDF Tool API』で保存される PDF のファイルサイズの上限は 2GB です。2GB を超える場合[*]は保存時にエラー「103」になります。

[*]：
出力ファイルサイズの上限は、『PDF Tool API』で PDF の加工や編集を行った結果の出力 PDF が 2GB を超えるかで判定します。
たとえば複数の PDF ファイルを結合してひとつのファイルにする場合、『PDF Tool API』では物理的に PDF ファイルを結合する処理ではないため、結合前の PDF ファイルサイズの合計が 2GB 未満であっても保存できない場合があります。

7-8　ロングパス対応（Windows）

パスの指定にて、ロングパスに対応しています。これにより、Windows 10 バージョン 1607 以降で対応している長いパスが使用可能です。

7-9　PDF/A、PDF/X および PDF/E に対する編集について

PDF/A、PDF/X、および PDF/E の規格を持つファイルに対してどのような処理を行うのか、オプションで指定可能です。オプション指定がない場合、規格情報を削除し処理を行います。

[処理オプションの種類]

- PDF の規格情報は削除する：規格情報を削除し編集処理を行う　※デフォルト
- エラーにする：編集処理を行わないようエラーを返すようにする

- PDF 規格情報はそのままの状態で扱う：規格情報はそのままの状態で編集処理を行う

7-10　画像の扱い

7-10-1　対応形式

『PDF Tool API』 が対応している画像形式は以下の通りです。

- bitmap
- jpeg
- png
- tiff
- gif

7-10-2　画像のマスク処理

『PDF Tool API』 でサポートしている画像マスクは以下の 4 種類です。

- ステンシルマスク
- カラーキーマスク
- 明示マスク
- ソフトマスク

マスクの種類により、取り扱う画像に特長があります。

[ステンシルマスク]

入力画像には白黒 2 値画像を指定してください。白黒 2 値画像以外の場合はエラー（234:STENCIL_MASK_IS_NOT_SINGLE）となります。

[カラーキーマスク]

入力画像には、カラースペースが「RGB」であるビットマップを指定してください。ビットマップ以外の画像や、indexed color を持つビットマップの場合はエラー（235:UNSUPPORTED_IMAGE_FOR_COLORKEY_MASK）となります。

[明示マスク]

マスクする画像には、白黒 2 値画像を指定してください。白黒 2 値画像以外の場合はエラー（237:EXPLICIT_MASK_IS_NOT_SINGLE）となります。

入力画像にサポートされていない画像が指定された場合は、エラー（232:UNSUPPORTED_IMAGE）となります。

[ソフトマスク]

マスクする画像には、グレースケール画像の指定を推奨します。指定された画像でマスクすることができなかった場合はエラー（232:UNSUPPORTED_IMAGE）となります。

入力画像にサポートされていない画像が指定された場合は、エラー（232:UNSUPPORTED_IMAGE）となります。

7-10-3　　ダウンサンプリングと JPEG 圧縮

- ダウンサンプリングの対象とする画像の条件として、Filter の種類を指定することができます。対象 Filter を 1 種類だけ指定した場合は、指定したその Filter のみを持つ画像だけがダウンサンプリングの対象となります。
- 画像データの中には、Filter を配列で複数持つものがあります。その Filter 配列の中にダウンサンプリングの対象として指定した Filter が含まれていても、その画像データはダウンサンプリングの対象とはなりません。
- ダウンサンプリングを行ったとき、カラースペースが元のデータとは異なる場合があります。

7-10-4　　画像抽出

- 画像抽出処理は、オブジェクト 1 個をひとつの画像ファイルとして出力します。そのため、出力される画像ファイル数が見えている画像数と異なる場合があります。例えば、表示上は 1 個の画像に見えてもオブジェクトが細かく分かれている場合では、出力される画像ファイル数が多くなります。
- 画像抽出処理は、ページコンテンツ全体を対象としています。表示領域がページコンテンツの一部分である場合、表示されていない部分の画像も抽出対象となります。

- 画像抽出処理は、クリッピングされて表示されていない部分の画像も抽出対象となります。

7-11　カラープロファイルの扱い

本節では『PDF Tool API』におけるカラープロファイルの扱いについて説明します。

カラープロファイルは、PDF/A 変換処理で使用するファイルです。

7-11-1　　カラープロファイルの検索

PDF/A 変換処理では、RGB 系、CMYK 系の 2 種類のカラープロファイルを指定します。

指定がない場合、『PDF Tool API』は sRGB2014.icc と JapanColor2001Coated.icc のカラープロファイルを検索します。各カラープロファイルの検索順は以下の仕様となり、Windows 版と Linux 版で検索順序が異なります。

[Windows 版]

1. dll ファイル[PdfTk80.dll]があるフォルダ内
2. 環境変数[PTL80_ICCPROFILE_PATH]で指定されたフォルダ内

[Linux 版]

1. 環境変数[PTL80_ICCPROFILE_PATH]で指定されたフォルダ内

7-11-2　　カラープロファイルに対するインストーラの動作

『PDF Tool API』のインストーラはカラープロファイルに対し、以下の内容で動作・設定します。

- sRGB2014.icc と JapanColor2001Coated.icc をインストールフォルダ内の「icc」フォルダに配置します。
- Windows 版では[PTL80_ICCPROFILE_PATH]をインストールフォルダ内の「icc」フォルダに指定します。

7-12　PtlParamString クラス（C++ API）の概要

- 「PtlParamString」は、C++でだけ用いられる文字列クラスです。本クラスはマルチプラットフォームにおける文字列処理を統一的に扱うための、抽象化された文字列クラスです。

使用上の注意点

- 『PDF Tool API』の C++ API を利用する際、文字列は「PtlParamString」を使用してください。
- PtlParamString の初期化の際、文字列に Unicode を使用する場合は CP_UChar ベースの文字列を用いたコンストラクタを呼び出してください。
  wchar_t 型は、Windows では「UTF-16」で 2 バイト、Linux では「UTF-32」で 4 バイトです。「CP_UChar」を使うことで、Windows と Linux の違いを吸収します。

CP_UChar 型について

「CP_UChar」は、以下のように定義される『PDF Tool API』独自の型です。

```
#ifdef WIN32
typedef unsigned short CP_UChar; // UTF-16
#else
typedef unsigned int CP_UChar; // UTF-32
#endif
```

7-13　透かし（ウォーターマーク）の挿入と削除　*V8.0NEW（一部）

- PDF ファイルの各ページに透かしを挿入します。
- 挿入可能な透かしの種類は以下の通りです：
  - テキスト透かし
  - 画像透かし（画像ファイルを挿入）
  - PDF 透かし（PDF ファイルを挿入）
  - 色透かし
- 『PDF Tool API』により挿入した透かしは、『PDF Tool API』により削除可能です。

- 透かしの挿入時に指定可能なパラメータは以下の種類です：
  - 透かしの配置
  - 透かしの角度
  - 挿入ページ
  - ビューワ上での表示・非表示
  - プリント時の表示・非表示
  - 透かしの名前
  - 透かしの不透明度
  - タイリング
  - 文書の前面配置・後面配置
- 挿入ページの指定は以下の種類から選択可能です。
  - 全ページに挿入
  - 奇数ページに挿入
  - 偶数ページに挿入
  - 挿入ページを指定（複数指定可能）
- 挿入する透かしを Adobe Acrobat©が持つ「透かし」機能で編集可能なものにするオプションが選択できます。（Acrobat 互換）

[テキスト透かし]

- 指定したテキストを透かしとしてページに挿入します。横書き・縦書きの指定、複数行テキストの挿入が可能です。
- 以下のオプションが指定可能です。：
  フォントの種類、文字サイズ、文字色、透明度、挿入範囲の矩形、挿入角度、タイリング、下線付きテキスト
- 複数行テキストを挿入する場合は、テキストを指定するときに改行コード（0x0A）を含めてください。
  ただし、挿入角度あるいはタイリングのオプションが指定された場合、改行は行われません。テキストは一行になります。
- フォントは、動作環境に存在するフォントを指定してください。
- 文字サイズが大きくテキスト全体が挿入範囲の矩形より大きい場合、テキスト全体が矩形に収まるよう縮小します。
- 文字サイズが小さくテキスト全体が挿入範囲の矩形に収まる場合、指定した大きさで挿入します。
- 文字色は、文字の塗りつぶし色と輪郭の色をそれぞれ指定できます。
- 挿入範囲の矩形が指定されていない場合、ページサイズが範囲となります。
- 挿入角度は、任意の数値が指定可能です。指定した角度だけ左回りに傾けます。
- 角度とタイリングを同時に設定した場合、指定角度に傾いたテキストをタイリングします。
- Acrobat 互換のオプションと以下のオプションは同時指定できません。指定があるとエラーになります。
  - 文字の輪郭色、ページ対角線配置、挿入範囲の矩形、余白、タイリング、連続しない挿入ページ

[画像透かし]

- 指定した画像ファイルを透かしとして挿入します。
- 透かしに指定可能な画像形式は以下になります。：bitmap、jpeg、png、tiff、gif
- マルチ tiff ファイルを透かしとして指定できます。この場合、どのページを透かしにするかの指定が必要です。

- 以下のオプションが指定可能です：
  画像の倍率、透明度、挿入範囲の矩形、挿入角度、タイリング
- 挿入範囲の矩形が指定されていない場合、ページサイズが範囲となります。
- 画像サイズが挿入範囲の矩形より大きい場合、矩形に収まるよう縦横比を維持したまま縮小します。
- 画像サイズが挿入範囲の矩形より小さい場合、画像サイズのまま挿入します。
- 挿入角度は、任意の数値が指定可能です。指定した角度だけ左回りに傾けます。　*V8.0NEW
- 角度とタイリングを同時に設定した場合、指定角度に傾いた画像をタイリングします。
- Acrobat 互換のオプションと以下のオプションは同時指定できません。指定があるとエラーになります。
  - 挿入範囲の矩形、余白、タイリング、連続しない挿入ページ

[PDF 透かし]

- 指定した PDF ファイルを透かしとして挿入します。
- 複数ページある PDF ファイルを透かしとして指定できます。この場合、どのページを透かしにするかを指定します。
- その他は、[画像透かし]と同等の仕様です。

[色透かし]

- 指定した色で透かしとして塗りつぶします。
- オプション：透かしとする色（RGB 指定)、透明度、挿入範囲の矩形などのオプションがあります。
- Acrobat 互換の透かしとして挿入する場合、次のオプションは同時指定できません。指定があるとエラーになります。
  - 挿入範囲の矩形、余白、タイリング、連続しない挿入ページ

7-14　PDF のセキュリティ

PDF セキュリティの設定、セキュリティの削除、設定されているセキュリティ情報の取得を行うことができます。

7-14-1　　PDF セキュリティ設定：パスワードセキュリティ

- パスワードによる暗号化が可能です。
- 使用可能なパスワードは以下の種類です：
  - ユーザーパスワード
  - オーナーパスワード
- パスワードを入力することで、セキュリティをかけられたファイルを開くことも可能です。
- オーナーパスワードによる暗号化ではセキュリティの権限設定が可能です。

- 権限設定には以下の項目があります：
  - 印刷権限
  - 変更権限
  - コピー可否
  - アクセシビリティに基づいたアクセスの可否
- 設定可能な暗号化方式の種類は以下の種類です：
  - 128bit RC4
  - 128bit AES
  - 256bit AES
- パスワードセキュリティでは、暗号化する文書コンポーネントの範囲を設定することができます。
  範囲は以下から指定できます：
  - 文書のすべてのコンテンツを暗号化
  - 文書のメタデータを除くすべてのコンテンツを暗号化
  - 添付ファイルのみを暗号化
- 「添付ファイルのみを暗号化」は、PDF ファイルに添付されているファイルの暗号化です。PDF の文書の暗号化は行われません。添付ファイルを開くためのパスワードとして、ユーザーパスワードを設定してください。
- 「添付ファイルのみの暗号化」では、セキュリティの権限設定は行われません。権限パスワード（＝オーナーパスワード）設定を含む権限に関する設定は行わないでください。

7-14-2　　PDF セキュリティ設定：証明書セキュリティ

- 証明書を用いて PDF ファイルを暗号化します。PDF ファイルを開く際には、暗号化した証明書に対応した秘密鍵が必要です。
- 暗号化の際には X.509 形式の証明書、または PKCS#12 形式の秘密鍵を使用します。
- 設定可能な暗号化の種類は以下の種類です：
  - 128bit RC4
  - 128bit AES
  - 256bit AES
- 暗号化されているファイルを読み込む場合、PKCS#12 形式の秘密鍵を使用します。
- ひとつの PDF に対し複数の証明書を受信者として登録することができます。また、受信者ごとに権限設定を変えることが可能です。
- 権限設定には以下の項目があります：
  - 印刷権限
  - 変更権限
  - コピー可否
  - アクセシビリティに基づいたアクセスの可否
  - 暗号化の解除を含むすべての権限の付与

- 証明書セキュリティで複数の受信者を登録した PDF ファイルに対しセキュリティ削除処理を行った場合、すべての証明書セキュリティが削除されます。
- 証明書セキュリティの権限を個別に変更することはできません。
- すでにセキュリティが設定されているファイルに対し、受信者を追加して設定することはできません。
- セキュリティ権限や受信者を変更したい場合は一旦セキュリティを削除した上で再度新しく暗号化してください。
- 証明書によるセキュリティ設定を行う場合、Web 表示用に最適化（＝線形化）する保存は行われません。
- Adobe Acrobat©で作成した 1024bit RSA の pfx ファイルは使用できません

7-14-3　　PDF セキュリティの削除

- ファイルに付与されている PDF セキュリティを削除します。
  削除した場合、セキュリティの付いていない PDF ファイルとして保存ができます。
- 以下のセキュリティ方式の PDF のセキュリティを削除可能です
  - パスワードセキュリティ
  - 証明書セキュリティ
- 削除するには、セキュリティを解除するためのパスワードや秘密鍵を用意してください。
- 証明書セキュリティの削除をする場合、すべての権限を付与した受信者の秘密鍵で PDF ファイルを開く必要があります。
  そのため、セキュリティの削除をしたい場合は暗号化の時点ですべての権限を付与した受信者を最低 1 つ登録してある必要があります。

7-14-4　　PDF セキュリティの情報取得

- 付与されているセキュリティの内容を取得します。
- 以下のセキュリティ方式の PDF からセキュリティ情報が取得可能です
  - パスワードセキュリティ
  - 証明書セキュリティ
- セキュリティの内容を取得するためには、ファイルを開く必要があります。
  ファイルを開くためのパスワードや秘密鍵が必要な場合はそれらを用意してください。

7-15　文書情報の取得と設定

- PDF ファイルに設定されている文書情報の取得、PDF ファイルへの文書情報の設定ができます。
- 取得・設定可能な文書情報は以下の項目です：
  - タイトル
  - 著者（作成者）
  - サブジェクト（サブタイトル）
  - キーワード
  - 作成日
  - 更新日
  - クリエータ（PDF 生成アプリケーション名）
  - プロデューサ（PDF 変換ライブラリ名）
  - カスタムプロパティの名前・値
- 任意の項目名を持つカスタムプロパティの設定と削除ができます。ただし、次の単語はプロパティの項目名として指定することはできません。エラーとなります。
  - Title , Author , Subject , Keywords , Creator , Producer , CreationDate , ModDate , Trapped

文書情報の他に取得可能な情報に関しては『PDF の情報取得』を参照してください。

7-16　ページ操作

ページの作成・挿入・移動・削除、及びそれらを組み合わせた PDF の分割などの操作について説明します。

7-16-1　　ページの作成

- 空のページを作成することが可能です。
- 作成したページの描画領域を mm 単位で指定します。

7-16-2　　ページの挿入

- PDF ファイルの任意のページを挿入することが可能です。
  以下のような挿入方法が選べます。

- ファイル末尾への挿入
- 任意の位置への挿入

- 挿入する対象をページ単体でなく、PDF ファイルの任意の範囲を指定することも可能です。

- 注釈やフォーム、しおり、添付ファイルとともに結合するかどうかのオプションがあります。

7-16-3　　ページの削除

- PDF ファイルの指定したページを削除します。

7-16-4　　ページの入れ替え

- ページ順序の入れ替えが可能です。

7-16-5　　ページの回転

- PDF ファイルの任意のページを回転します。

7-16-6　　ページ描画範囲の変更

- ページが持つ境界値を変更可能です。
  具体的には以下の項目を変更可能です。
  - ArtBox
  - BleedBox
  - CropBox
  - MediaBox
  - TrimBox
  - ViewBox

7-16-7　　ページの拡大・縮小　*V8.0NEW

- PDF ファイルのページの内容を拡大、縮小します。

- 縦横比は維持されます。
- 拡大・縮小の方法は以下です。
  - 倍率指定：現在のページに対する倍率を指定します。
  - 用紙サイズ指定：あらかじめ定義されている用紙サイズを指定します。　*V8.0NEW
  - 任意のサイズ：幅と高さを指定します。　*V8.0NEW

7-17　ページ操作の応用

ページ操作を応用することで PDF 文書の結合やページ抽出、分割が可能です。

7-17-1　PDF ファイルの結合

- PDF ファイルの結合が可能です。複数の PDF ファイルを結合してひとつの PDF として保存します。
- ページの挿入操作を使って実現します。
  挿入対象を個々のページではなく PDF ファイルで指定することが可能です。
- 注釈やフォーム、しおり、添付ファイルとともに結合するかどうかのオプションがあります。

仕様上の注意：

『PDF Tool API』で行う PDF ファイルの結合は、物理的に PDF ファイルを結合する処理ではありません。また、『PDF Tool API』は仕様上、ファイルサイズが 2GB を超える PDF の読み書きができません。

そのため、結合結果の出力 PDF が 2GB を超えてしまった場合、たとえ結合前の PDF ファイルサイズの合計が 2GB 未満であってもエラーになることがあります。

7-17-2　ページ抽出

- 任意のページの抽出が可能です。
- 入力 PDF から任意のページを抽出し、別の PDF として保存します。
- ページの挿入と新規ドキュメントの作成を組み合わせることで実現します。

7-17-3　PDF ファイルの分割

- PDF ファイルの分割が可能です。

- 入力 PDF から分割したいページを抜き出し、それぞれ別の PDF として保存します。
- ページの挿入と新規ドキュメントの作成を組み合わせることで実現します。

7-18　データ削除処理（墨消し）

- 指定された矩形内の指定した要素を削除します。
  この「削除」とは、非表示にするということではなく、PDF ファイルからデータそのものを取り除くことです。
- 削除可能な要素は以下の種類です：
  - テキスト
  - 画像
  - パス
- 以下のオプションを指定可能です：
  - 矩形内のどの要素（テキスト、画像、パス）を削除するか（フラグ指定による複数指定も可）
  - 指定した矩形の色
- テキストの削除
  - 指定矩形内の文字が削除されます。
  - 指定矩形の線上の文字も削除対象です。
  - 文字が矩形にどの程度重なっていたら削除するかを指定可能です。
  - リガチャ文字に対して、グリフ自体がリガチャの場合、その一部分を削除することはできません。
- 画像の削除
  - 指定矩形内の画像データを部分削除します。
  - 画像の残された部分のカラースペース、圧縮形式が元のデータから変更される場合があります。
- 図形の削除
  - 線や四角形などの図形（＝パス）に対し、図形全体が矩形内に指定された場合に削除します。指定矩形内に図形全体が収まっていない場合、その図形は削除されません。

7-19　画像抽出

- PDF ページコンテンツ上の画像をファイルに書き出します。
- 出力画像形式は、bitmap、jpeg、png。また、各画像の幅、高さ、PPI（Pixel Per Inch）が取得可能です。

- 抽出時のオプションとして、以下の機能があります。
  - 変更を加えず抽出する機能
  - 画像のマスクの種類を取得する機能
- 抽出対象範囲は、表示領域か否かに関係なく、ページコンテンツ全体です。
  つまり、PDF の表示領域外の部分を持つ画像が抽出される場合があります。

7-20　テキスト抽出　*V8.0NEW（一部）

- PDF ページコンテンツ上のテキストを取得・抽出します。
- 抽出対象範囲は、表示領域内か否かに関係なく、ページコンテンツ全体です。
- 抽出範囲の矩形を指定した場合、矩形内のテキストを取得します。クリッピングされて表示されない部分のテキストも取得対象となります。
- 矩形を複数指定した場合、それぞれの範囲のテキストが抽出されます。　*V8.0NEW
- 注釈や PDF フォームデータなどのテキストは抽出対象になりません。

[各オプションについて]

- 「取得順」オプション
  - 取得したテキストをそのまま抽出します。このため、見た目とは異なる順序で抽出される場合があります。
  - 以下の「ソート」オプションとどちらかを選択します。
  - こちらがデフォルトです。
- 「ソート」オプション
  - テキストを座標順にソートします。
  - 抽出対象が欧文の場合、単語間に空白(スペース)を挿入します。単語のかたまりとして判定できなかった場合、次の単語との間に空白が入らず続いた状態で抽出されます。
  - 行末の「ハイフン（-)」を、単語が 2 行にわたっている場合に表記されるものとみなして削除します。
- 「改行コード挿入」オプション　*V8.0NEW
  - テキスト抽出時に行を判定し末尾に改行コード「0x0D 0x0A」を挿入します。「ソート」オプションとともに指定することで動作します。
  - 縦書き文書に対しても処理ができます。
  - 抽出範囲のテキストの並びが、段組み、縦横が混在しているなどテキストの方向が一定でない場合は、期待する位置に改行コードの挿入は行われません。
- 「特定文字の削除・スペース置換をする」オプション
  - 特殊なスペース文字やハイフンなどに対し、抽出時に削除、またはスペースに置き換える処理をします。

- 処理対象の文字種別はフラグ指定により、複数の同時指定が可能です。

- 「指定されたユニコード文字を削除する」オプション
  - 指定されたユニコードの文字を削除します。したがって該当の文字は抽出されません。ユニコード文字の指定がない場合、削除は行われません。
- 「同じ行にある文字とみなすしきい値」オプション
  - 文字と文字がどれだけ重なっていると同じ行とみなすか、その割合を設定します。この設定は、座標でソートする場合に有効です。
- 「同じ文字が重なっているとき抽出から除外するしきい値」オプション
  - 同じ文字が重なっているとき、文字と文字がどれだけ重なっているときに取り除くか、その割合を設定します。この設定は、座標でソートする場合に有効です。
- 「違う文字が重なっているとき抽出から除外するしきい値」オプション
  - 違う文字が重なっているとき、文字と文字がどれだけ重なっているときに取り除くか、その割合を設定します。重なっていると判定したとき、先に現れた文字、つまり、重なりの下になっている文字が取り除かれます。この設定は、座標でソートする場合に有効です。
- 「文字が矩形内にあるとみなすしきい値」オプション
  - 文字が抽出範囲の矩形とどれだけ重なっているときに抽出対象とするか、その割合を設定します。
- 「ActualText を無視する」オプション
  - 異体字などで文字を置き換える ActualText があった場合に無視して元 PDF に描画された文字を抽出します。

[「ソート」オプションに関する注意事項]

- ２段以上に段組みされたページに対して「ソート」オプションで抽出しても段ごとにテキストは抽出されません。これは段をまたいでソートを行うためです。１段ずつ抽出範囲の矩形指定をすることで段ごとに抽出できるようになります。
- 縦書き、横書きが混在したページに対して「ソート」オプションで抽出しても期待通りにテキスト抽出できません。「ソート」オプションを利用するには、縦書きのみ、横書きのみとなるよう抽出範囲を指定してください。
- 異なった文字サイズが混在している場合など１行と認識できないとき、期待通りに抽出されません。

7-21　テキスト検索　*V8.0NEW（一部）

- PDF ページコンテンツ上のテキストを対象に検索を行います。
- 検索対象範囲は、表示領域か否かに関係なく、ページコンテンツ全体です。
- 複数キーワード検索に対応しています。
- PDF の表示上はテキストであっても、オブジェクトの種類がテキストではなく画像やパスである場合があります。この場合、テキスト検索の対象とはなりません。

- 検索キーワードがページをまたいでいる場合、その箇所は検索されません。
- 検索キーワードのスペースは、取り除いた状態で検索処理が行われます。
- 検索キーワードに正規表現が使用できます。　*V8.0NEW
- 検索とともに、以下の処理を組み合わせた API を用意しました。
  - 検索結果場所の矩形取得
  - 検索結果にハイライト注釈付与
  - 検索結果に Redaction 注釈（墨消し注釈）付与　*V8.0NEW
  - 検索結果箇所のデータ削除（マスク処理）

[各オプションについて]

- 「取得順」オプション
  - 取得したテキストをそのまま抽出します。このため、見た目とは異なる順序で抽出される場合があります。
  - こちらと「ソート」オプションのどちらかを選択します。デフォルト値は「取得順」です。
- 「ソート」オプション
  - テキストを座標順にソートします。
- 「同じ行にある文字とみなすしきい値」オプション
  - 文字と文字がどれだけ重なっていると同じ行とみなすか、その割合を設定します。この設定は、座標でソートする場合に有効です。
- 「大文字と小文字を区別する」オプション
  - 検索時にアルファベットの大文字と小文字を区別して検索します。
- 「ActualText を無視する」オプション
  - 異体字などで文字を置き換える ActualText があった場合に無視して元 PDF に描画された文字を検索対象とします。

7-22　フォントの埋め込み

- フォント情報が埋め込まれていないフォントに対し、フォント情報を埋め込み状態に変更します。
- 埋め込み対象のフォント情報は、動作環境に存在していなければなりません。

7-23　しおりの取得・作成・削除

しおりの編集に関して、以下の加工が可能です。

- しおり情報の取得
- 既存のしおりの編集
- 新規しおりの追加
- しおりの削除

7-24　PDF の開き方の取得と設定

- PDF ファイルの開き方情報の取得と設定が可能です。
- 取得・設定可能な開き方の種類は以下です：
  - オープンアクション
  - ページモード
  - ページレイアウト
  - UI 表示
  - ウィンドウオプション
  - タイトルバー上の文書タイトル表示の可否
- 開き方設定の削除も可能です。
- 『PDF Tool API』では「メニューバー非表示」と「Windows コントロール非表示」を同時に設定することはできます。ただし、この設定がされた PDF ファイルの表示については、PDF ビューワの仕様によります。
  例えば、Acrobat / Acrobat Reader ではメニューバーは表示されます。

7-25　添付ファイル情報の取得、添付ファイルの追加・削除・書き出し

- PDF ファイルに対して添付されている添付ファイル情報を取得します。
- 添付ファイルの追加、削除、添付ファイルの書き出しが可能です。

7-26　注釈の新規作成　*V8.0NEW（一部）

- 新しく注釈を作成することができます。
- 作成可能な注釈の種類は以下です
  - テキスト（PDF ビューアによっては「ノート」という名称の場合があります）
  - スタンプ：PDF にあらかじめ用意されているスタンプ
  - ファイル添付
  - リンク
  - ハイライト
  - 図形（円、矩形、多角形、線、折れ線）　*V8.0NEW
  - FreeText（引き出し線あり、引き出し線なし）　*V8.0NEW
  - 下線・波線・取り消し線　*V8.0NEW
  - Redaction（PDF ビューアによっては「墨消し」という名称の場合があります）　*V8.0NEW
- ハイライト注釈を扱う PtlAnnotTextMarkup クラスのオブジェクトに対しては、基底クラス PtlAnnot の setRect 関数で矩形座標の設定は行わないでください。
- 注釈には、ページに合わせて拡大や回転はしない、編集はさせない、などのフラグがあります。『PDF Tool API』では、これらのフラグ設定もできるようになっています。
  例えば、「NoRotate」というフラグは、これが ON のとき、ページの回転角度と関係なく注釈をページ左上に表示します。ただし、PDF ビューワにより、表示の位置や形状が指定したものと異なったり、注釈自体の位置が異なったりする場合があります。『PDF Tool API』では NoRotate フラグのデフォルトは OFF です。

7-27　注釈の編集　*V8.0NEW（一部）

- 注釈を編集することができます。
- 編集可能な注釈の種類は以下です。
  - テキスト（PDF ビューアによっては「ノート」という名称の場合があります）
  - スタンプ：PDF にあらかじめ用意されているスタンプ
  - ファイル添付
  - リンク
  - ハイライト

- 図形（円、矩形、多角形、線、折れ線）　*V8.0NEW
- 下線・波線・取り消し線　*V8.0NEW
- Redaction（PDF ビューアによっては「墨消し」という名称の場合があります）　*V8.0NEW

7-28　注釈情報の取得、削除

- 注釈の情報を取得することができます。注釈の種類、位置、表示状態を示すフラグ、アイコンの種類、更新日などが取得されます。
- 注釈を削除することができます。

7-29　カスタムスタンプ注釈作成

- 任意の画像ファイルや PDF ファイルからスタンプ注釈を作成します。

7-30　閲覧制限設定

- 指定の期間、あるいは指定の場所(URL/URI)で表示された場合にのみ閲覧可能にする制限設定をします。
- ページごとに分けて閲覧制限が設定可能です。
- 場所（フォルダパス）を指定する場合、フォルダパスの末尾や途中にワイルドカードが使用できます。
- 開始日・終了日の時刻設定には「協定世界時」（UTC）を使用しています。
- Acrobat Java Script を利用して制限制御を行います。制限動作は、Acrobat でファイルを開いた場合にのみ保証されます。

[閲覧制限機能の製品仕様]

- Acrobat Java Script をサポートしていない PDF ビューアや、Acrobat Java Script の機能が無効になっている PDF ビューア、PDF 表示機能を持つブラウザーなどでは設定された制限動作は行われません。

その場合は常に閲覧不可状態、もしくはページが真っ白な状態で表示されます。あるいは、ページや表示倍率の切り替え時、「レイヤー」操作が可能になっている PDF ビューアでレイヤーの有効・無効の切り替え時に、中身が一時的に表示される場合があります。Acrobat Java Script をサポートしていても、PDF ビューアによっては期待する制限動作をしない場合があります。

- 次のブラウザーでは閲覧制限は正しく動作しません（弊社確認)。これら以外のブラウザーでの動作は未確認です。
  - Microsoft Edge / Google Chrome / Firefox
- 閲覧制限設定は PDF を暗号化するものではありません。PDF を扱うツールによっては、ページの内容が見えたり抽出することができる場合があります。
- 閲覧制限を設定する場合、セキュリティの権限とオーナーパスワードも同時に設定することをお薦めます。
- 閲覧制限設定を行った PDF から閲覧制限設定を取り除くことはできません。
- すでに閲覧制限設定された PDF に対し再度閲覧制限設定を行った場合、設定された閲覧制限の制限動作は保証されません。
- 閲覧制限設定された PDF を Acrobat / Adobe Reader で表示させたとき、閲覧できる状態であっても「サムネイル」ナビゲーションパネルでは内容が表示されません。
- 閲覧制限設定された PDF のしおり、添付ファイル、文書情報は、閲覧不可や Java Script が無効な場合であっても非表示にはなりません。
- 閲覧制限設定された PDF がポートフォリオの場合、ポートフォリオは添付ファイル機能であるため、閲覧不可や Java Script が無効な場合であっても非表示にはなりません。
- 閲覧制限設定された PDF を閲覧不可や Java Script が無効なときに Acrobat / Adobe Reader で表示している場合、「注釈」ナビゲーションパネルは非表示にはなりません。
- 閲覧制限設定を行った PDF ファイルに対して追加した注釈やフォームは、閲覧不可や Java Script が無効な場合に非表示にはなりません。
- 閲覧制限設定された PDF において閲覧不可や Java Script が無効なときにページ上の注釈あるいはフォームを選択できた場合には、注釈、フォームは表示状態になります。
- 閲覧制限設定された PDF が閲覧不可や Java Script が無効なとき、Acrobat / Adobe Reader 以外の PDF ビューアでは注釈とフォームが表示される場合があります。
- 閲覧制限設定処理において、閲覧制限時に表示する透かしは 1 つの「PtlParamWaterMark」のみ設定可能です。
- 閲覧制限設定において appendValidURL()でパスを指定する場合、ファイル名やフォルダ名に使用できない文字（¥ / : * ? " < > |）やサロゲートペア文字は指定できません。指定された場合、動作結果は不定となります。

7-31　ページコンテンツへの描画：テキスト　*V8.0NEW（一部）

- PDF のページ上にテキストを挿入します。

- 横書き、縦書きができます。
- 絵文字が書けます。絵文字に対応しているフォントを指定して描画します。　*V8.0NEW
- テキストを直接挿入するほか、以下の装飾や機能を備えた「テキストボックス」機能を用いてのテキスト挿入が可能です。
  - 下線や取り消し線の装飾や指定範囲での折り返し描画処理などが可能です。
  - 矩形（＝テキストボックス）内のテキストに対し、行間隔や上付き・下付き文字といった書式を指定できます。
  - 矩形自体の輪郭線の色や背景色の指定、テキストの量に合わせたサイズ変更が可能です。
  - 矩形の枠線の線種が設定可能です。　*V8.0NEW

7-32　ページコンテンツへの描画：図形　*V8.0NEW（一部）

- PDF のページ上に図形を描画します。
- 描画可能な図形は以下です。
  - 直線
  - 矩形
  - 角丸矩形
  - 楕円／円
  - 多角形　*V8.0NEW
  - 折れ線　*V8.0NEW
- それぞれ必要な座標や長さを指定して PDF 内に挿入します。

7-33　ページコンテンツへの描画：フォーム XObject

- 指定した PDF のページをフォーム XObject として描画します。
- この機能は例えば、複数のページを 1 ページに描画することでページ割付処理に応用できます。

注意；

フォーム XObject は、いわゆる「入力フォーム」であるフォームフィールドとは別種のオブジェクトです。

7-34　ページコンテンツへの描画：画像

- 画像ファイルを PDF ページ上の指定矩形内に描画します。
- 矩形に合わせて画像を拡大・縮小して描画することができます。
- 任意の角度に回転して描画することも可能です。

7-35　コンテンツへの描画：注釈　*V8.0NEW

- 注釈を同じ外観を持つ要素をページのコンテンツとして描画します。
  処理対象の注釈を、同じ位置・同じ座標で PDF 内に埋め込んだような処理となります。
- 描画されるのは外観だけです。注釈のプロパティ情報・プロパティダイアログは描画されません。
- 以下の種類の注釈が対象です。
  - スタンプ
  - 図形（円、矩形、線、多角形、折れ線）
  - FreeText(引き出し線あり、引き出し線なし)
  - Widget（ボタン、ラジオボタン、チェックボックス、テキストフィールド、リストボックス、コンボボックスの各フォームフィールド）

7-36　エレメント情報の取得　*V8.0NEW（一部）

- PDF ページのコンテント上にあるエレメント（テキスト、画像、図形など）の情報を取得します。
- 全ての種類に共通して取得されるのは以下の情報です。
  - エレメントを囲む矩形座標
- テキストエレメントからは、さらに以下の情報を取得できます。
  - テキスト
  - テキストの固まりとしての座標
  - 1 文字ごとの座標
  - フォント名
  - エンコーディング名
  - 埋め込みか否か

- フォントサイズ
- 文字の塗りつぶし色と輪郭の色
- 文字の塗りつぶし・輪郭のカラースペース　*V8.0NEW

- 画像エレメントからは、さらに以下の情報を取得できます。
  - 高さ・幅の情報（ピクセル数）
  - 解像度（PPI）
- 図形（パスエレメント）からは、さらに以下の情報を取得できます。
  - 図形を描画するために必要な座標の情報
  - ストロークや塗りつぶしをする・しないのペイントフラグ
  - 塗りつぶし色と線の色

7-37　画像ファイルからの PDF ページ生成

- 画像ファイルから PDF のページを生成します。
- この機能を応用すれば、画像ファイルから直接 PDF を作ることも可能です。

7-38　リニアライズ保存（Web 表示用に最適化する保存処理）

- 出力する PDF に対し、リニアライズ保存をするかどうかの指定ができます。
- 一度リニアライズされて保存されたファイルを読み込んだ場合でも、増分を追加して保存などの処理をすると出力時にリニアライズが解除されてしまいます。その場合は出力時に再度リニアライズを指定する必要があります。

7-39　画像の最適化：ダウンサンプリングと JPEG 圧縮

- PDF ファイル内の画像データに対し、ダウンサンプリングと JPEG 圧縮を行います。
- ダウンサンプリングの種類は、「バイリニア」「バイキュービック」「ニアレストネイバー」の 3 種類です。
- 指定された解像度以上の解像度を持つ画像データに対し指定の解像度となるようダウンサンプリングが行われます。

- ダウンサンプリング率の下限値を設定することで、必要以上に画像を小さくして劣化が発生しないようにすることができます。
- JPEG 圧縮はダウンサンプリング処理と同時に行われます。また、JPEG 圧縮処理のみ行うことも可能です。

7-40　PDF の最適化　*V8.0NEW（一部）

- PDF ファイルを最適化します。
  ここでの「最適化」とは不要なデータの削除・統合・ダウンサンプリングをして PDF の軽量化をすることを指します。
- 最適化として以下の処理を選択可能です：
  - 画像のダウンサンプリング・圧縮による最適化
  - フォントの統合
  - オープンアクションの削除
  - しおりの削除
  - 注釈・フォームの削除
  - アーティクルの削除
  - サムネイルの削除
  - JavaScript の削除
  - タグ削除　*V8.0NEW
  - PieceInfo 削除　*V8.0NEW
  - 添付ファイル削除　*V8.0NEW

7-41　注釈データの読み書き

- マークアップ注釈（テキスト注釈、スタンプ注釈、図形注釈など）の FDF ファイルへの書き出しが可能です。
- 注釈データを書き出した FDF ファイルを PDF ファイルに対して取り込むことができます。
- 注釈データを PDF ファイルから別の PDF ファイルに対して取り込むことができます。

7-42　PDF フォームデータの FDF / XFDF のインポート・エクスポート

- PDF に記載されたフォームフィールドのデータをエクスポートできます。
  以下のファイル形式に対応しています。
  - FDF ファイルへのエクスポート
  - XFDF ファイルへのエクスポート
- PDF 内のフォームフィールドへデータをインポートできます。
  以下のファイル形式が扱えます。
  - FDF ファイルからのインポート
  - XFDF ファイルからのインポート
- 対応する PDF フォームの種類は以下の通りです。
  - テキスト
  - チェックボックス
  - ラジオボタン
  - リストボックス
  - ドロップダウン（コンボボックス）

[XFDF の仕様]

XFDF は「XML Forms Data Format」の頭文字で、PDF が持つフォームフィールドや注釈といったデータを XML 形式で扱うためのフォーマットです。ISO 19444-1 で規定された文書形式です。

＜記述例＞

```
<?xml version="1.0" encoding="UTF-8"?> ---①
<xfdf xmlns="http://ns.adobe.com/xfdf/" xml:space="preserve"> ---②
<f href="C:¥test¥test.pdf"/>
<ids original="7A0631678ED475F0898815F0A818CFA1" modified="BEF7724317B311718E8675B677EF9B4E"/> ---
③
<fields>
  <field name="Text1"> ---④
    <value>アンテナハウス株式会社</value> ---⑤
  </field>
  <field name="CheckBox1">
    <value>Off</value>
  </field>
</fields>
```

```
</xfdf>
```

『PDF Tool API』が出力する XFDF では、上記のような 1 行空けやインデントなどのレイアウトは行われません。

①		先頭 2 行はこの記述内容で固定されています。
②	<f href="xxx">	"xxx"は XFDF の元である PDF ファイル名
③	<ids original="xxx" modified="xxx">	「ids original」は XFDF の元である PDF ファイルの FDF 辞書の ID エントリ。 「modified」は ID エントリの第 2 識別子。 「ID エントリ」は PDF ファイルが持つ固有の情報です。
④	<field name="xxx">	"xxx"はフォームフィールドのフィールド名
⑤	<value="xxx">	"xxx"はフォームフィールドの値

7-43　PDF/A への変換と準拠確認　*V8.0NEW（一部）

- PDF/A への変換と準拠確認処理ができます。
- 変換処理・準拠確認に失敗した場合はエラーに応じたエラーコードを返します。
- 以下の規格に対応しています。
  - PDF/A-1a　*V8.0NEW
  - PDF/A-1b
  - PDF/A-2a　*V8.0NEW
  - PDF/A-2b
  - PDF/A-3a　*V8.0NEW
  - PDF/A-3b　*V8.0NEW
  - PDF/A-4 ：PDF2.0 用の PDF/A 規格　*V8.0NEW
  - PDF/A-4f：PDF/A-4 with Embedded Files　*V8.0NEW
  - PDF/A-4e：PDF/A-4 for Engineering　*V8.0NEW
- 変換時に使用する icc プロファイルを指定することができます。
- icc プロファイルが指定されなかった場合の動作は『カラープロファイルの扱い』を参照してください。

※　「PDF/A」は、文書の長期保存を目的とした国際基準規格です。

※　「PDF/A-1（PDF1.4 相当)」は、ISO 19005-1 に準拠した規格です。

※　「PDF/A-2（PDF1.7 相当)」は、ISO 19005-2 に準拠した規格です。

※　「PDF/A-3(PDF1.7 相当)」は、ISO 19005-3 に準拠した規格です。

※　「PDF/A-4(PDF2.0 相当)」は、ISO 19005-4 に準拠した規格です。

7-44　PDF/E-1 変換と準拠確認　*V8.0NEW

- 「PDF/E-1」(PDF/Engineering）への変換と、準拠確認に対応しています。
  - ✧　「PDF/E-1(PDF1.6 相当)」は、ISO 24517-1 に準拠した規格です。

7-45　PDF の情報取得　*V8.0NEW（一部）

- PDF ファイルが持つ様々な情報を取得できます。
  - 文書情報(*)
  - 総ページ数
  - ページサイズ
  - 開き方
  - 添付ファイル情報
  - セキュリティ情報
  - フォント情報
  - Web 表示用に最適化されているか否か（リニアライズ）
  - 添付ファイルを持っているか否か
  - 表示矩形の取得
  - サムネイルがあるか否か
  - 注釈があるか否か
  - 注釈の種類（全種類を判別可能）
  - 各注釈がマークアップ注釈か否か
  - 署名付きか否か
  - ページモード、ページレイアウトの有無判定
  - JavaScript アクション情報　*V8.0NEW
  - PDF のバージョン
  - PDF/A、PDF/X であるか否か
  - PDF/A のバージョン

(*):詳細は『文書情報の取得と設定』を参照

7-46　フォームフィールドの新規作成・情報取得　*V8.0NEW

- フォームフィールドを PDF ページ上に新規作成できます。
- フォームフィールドの情報が取得できます。フィールド名と値を取得します。
- 新規作成と情報取得が可能なフォームフィールドは以下の種類です。
  - テキストフィールド
  - チェックボックス
  - ラジオボタン
  - リストボックス
  - ドロップダウン

7-47　レイヤー関連処理　*V8.0NEW（一部）

- レイヤーとして PDF を挿入できます。
- レイヤー情報を取得可能です。　*V8.0NEW
- 取得可能な情報は以下の項目です：
  - レイヤー名
  - ロック状態
  - チェック状態
  - 親レイヤー・子レイヤーの情報
- レイヤー情報を編集可能です。　*V8.0NEW
- 編集可能な情報は以下の項目です：
  - レイヤー名
  - ロック状態
  - チェック状態
- レイヤーをページコンテントに統合できます。　*V8.0NEW
- グループ化されたレイヤーを解除し、それぞれ単独のレイヤーに分けることが可能です。　*V8.0NEW

7-48　Separation Color 対応

- Separation Color を用いた色指定・色情報の取得が可能です。
  扱える色は、ファンクションは「タイプ 2(指数補完)関数」、代替カラースペースは「CMYK」のみです。
- 指定した Separation Color は、図形やテキストの描画に使用できます。

7-49　DeviceN カラー対応

- DeviceN カラーを用いた色指定・色情報の取得が可能です。
- 指定した DeviceN カラーは、図形やテキストの描画に使用できます。

第8章　導入方法

8-1　　開発のための導入

プログラム開発において『PDF Tool API』を利用するには、付属のインストーラでセットアップを行います。

インストール方法については、『第 4 章 インストール／アンインストール』を参照してください。

8-2　　プログラム実行のための導入

『PDF Tool API』を利用して開発したプログラムを開発環境とは異なる実行環境で実行するには、プログラムの実行ファイルのほかに、『PDF Tool API』の実行ファイルを所定の場所に配置します。

『PDF Tool API』の実行ファイルは、開発を行ったプログラム言語により異なります。必要な実行ファイルについては、『第 6 章 モジュールファイルについて』を参照してください。

『PDF Tool API』の実行ファイルは 64bit 専用です。32bit のプログラムはサポートしていません。

Linux / Amazon Linux2023 版の実行ファイルは 64bit 用のみです。このため、32bit のプログラムからは使用できません。

8-3　　環境変数について

『PDF Tool API』では、以下の環境変数を使用します。

8-3-1 Windows

環境変数名	設定値	設定が必要な場合
PTL80_LIC_PATH	ライセンスファイル「ptalic.dat」が存在するフォルダパス	API のバイナリファイル「PdfTk80.dll」とは異なる場所にライセンスファイルを配置する場合
PTL80_FONT_CONFIGFILE	フォント構築ファイル「font-config.xml」のフルパス	システムのフォントフォルダとは異なる場所にあるフォントを参照する場合
PTL80_ICCPROFILE_PATH	ICC プロファイル (※) が存在するフォルダパス	ICC プロファイルを使用する PDF/A-1b、PDF/A-2b 変換処理を行うとき、かつ、ICC プロファイルを「PdfTk80.dll」とは異なるフォルダに配置する場合

8-3-2 Linux

環境変数名	設定値	設定が必要な場合
PTL80_LIC_PATH	ライセンスファイル「ptalic.dat」が存在するディレクトリ	設定は必須
PTL80_FONT_CONFIGFILE	フォント構築ファイル「font-config.xml」のフルパス	設定は必須 『PDF Tool API』 を特定の文字コードで使用する場合
PTL80_CODEPAGE	文字コード名	PTL80_CODEPAGE が存在しない場合、文字のエンコードは環境変数「LANG」の設定値にしたがいます どちらも存在しない場合、「UTF-8」で動作します。[設定例] PTL80_CODEPAGE=EUC-JP
PTL80_ICCPROFILE_PATH	ICC プロファイル（※）が存在するディレクトリ	ICC プロファイルを使用する PDF/A-1b、PDF/A-2b 変換処理を行うとき、この環境変数の設定は必須です

8-4 API を使用する開発のチュートリアル

『Antenna House PDF Tool API V8.0 サンプルコードのビルド手順』では、サンプルコードのビルドを例に開発環境やプロジェク

トの準備について説明しています。ご参考ください。

8-5　コマンドラインの使い方

コマンドラインの使い方については、『Antenna House PDF Tool API V8.0 コマンドライン説明書』をご参考ください。

第9章　簡単な使い方

「基本的な使い方」におけるコードはコードの中でも主な部分を抜粋したもので、細かい記述は省略してあります。

例えばオブジェクトの解放漏れ対策などの記述はありません。

9-1　　PDF を開く

9-1-1 PDF ファイルを開くために必要な要素

```
PtlPDFDocument doc = new PtlPDFDocument();                  //入力 PDF 用クラス
PtlParamInput inputFile = new PtlParamInput(inputFilePath); //入力 PDF のパス
doc.load(inputFile);                                        //パスを指定して読み込み
```

9-1-2 「PDF を開く」詳細

コード解説

```
PtlParamInput inputFile = new PtlParamInput(inputFilePath);
```

ファイルやストリームの入力指定に使うパラメータクラスを生成しています。

補足 1：

PtlParamInput は PDF に限らず、PDF に添付するファイルや挿入する画像の指定にも用います。

```
PtlPDFDocument doc = new PtlPDFDocument();
```

PDF ドキュメントのオブジェクトを生成します。

補足 2：

```
PtlPDFDocument doc = new PtlPDFDocument();
```

この時点の PtlPDFDocument は文書情報を持っていません。この後にある load を実行することで PDF ファイルの情報を持つことになります。

```
doc.load(inputFile);
```

load 関数で PDF ドキュメントの文書情報を読み込みます。

9-2　PDF を保存する

9-2-1 PDF ファイルを保存するために必要な要素

```
PtlPDFDocument doc = new PtlPDFDocument(); //出力 PDF 用クラス
//何らかの編集作業を行う
PtlParamOutput outputFile = new PtlParamOutput(outputFilePath); //ドキュメントの出力パス
doc.save(outputFile);                                    //出力パスを指定して出力
```

9-2-2 「PDF の保存」詳細

コード解説

```
PtlPDFDocument doc = new PtlPDFDocument();
```

あらかじめ、出力する PDF ドキュメントのオブジェクトを生成しておきます。

今回はオブジェクト名を仮に「doc」としています。

補足１：

PDF ドキュメントを出力するには何らかのデータが必要です。具体的には、ＰＤＦファイルを開くか、新規ページを追加するなどの操作が必要であり、内容が無いＰＤＦファイルは出力できません。

そのため、ここのコード記述だけを実行した場合はエラーになります。

```
PtlParamOutput outputFile = new PtlParamOutput(outputFilePath);
```

ファイルやストリームの出力指定に使うクラスで出力パスを指定します。

補足２：

PtlParamOutput は PDF に限らず、PDF に添付ファイルの出力や画像の抽出の指定にも用います。

```
doc.save(outputFile);
```

出力する PDF ドキュメントの情報を持つオブジェクトに対し、save 関数を実行します。

補足３：

出力する前に PtlPDFDocument.setSaveOption()を用いるとセーブ時のオプションを設定可能です。これにより、ＰＤＦのリニアライズ保存などが可能になります。

9-3　　ページ情報を操作する

9-3-1 ページ情報を操作するために必要な要素

```
//入力 PDF doc の取得は「PDF を開く」を参照
PtlPages pages = doc.getPages();          //ページコンテナを取得
PtlPage page = pages.get(pageIndexNum);     //ページ番号を指定して取得
```

9-3-2 「ページ情報の操作」詳細

前提

ページ情報は PDF 文書内部では「コンテナ構造」となっています。

そのため、文書全体のページコンテナを取得し、その後、目的のページを取得する必要があります。

（図を入れる）

コード解説

```
PtlPages pages = doc.getPages();
```

PDF ドキュメントからページコンテナを取得します。

```
PtlPage page = pages.get(pageIndexNum);
```

ページコンテナから欲しいページオブジェクトを得ます。

補足：

ページオブジェクトのインデックス番号は 0 オリジンです。

なお、空のページコンテナからページを取得しようとするとエラーとなります。

その場合は、以下のようなコードでエラーを予防できます。

```
if(!pages.isEmpty()){
    System.err.println("ページコンテナが空です");
}else{
    PtlPage page = pages.get(pageIndexNum);
}
```

ここでは isEmpty 関数でページコンテナが空でないことを確認しています。

9-4　ページコンテントを操作する

9-4-1 ページコンテントを操作するために必要な要素

```
//入力 PDF doc の取得は「PDF を開く」を参照
//ページコンテナ・ページ取得は「ページ情報を操作する」を参照
PtlContent content = page.getContent();     //ページコンテントを取得

//以降、取得したページコンテントに対する操作を行う

content.writeString(rectToWrite, PtlContent.ALIGN.ALIGN_CENTER, textToAdd, paramText);
```

9-4-2 「ページコンテントの操作」詳細

前提：

テキストや画像を操作したい場合には、ページコンテントを通じてアクセスする必要があります。

ページコンテントはページに 1 つ含まれており、そこにテキストや画像といった要素の情報が含まれています。

コード解説

```
PtlContent content = page.getContent();
```

ページオブジェクトからコンテントを取得します。

```
content.writeString(rectToWrite, PtlContent.ALIGN.ALIGN_CENTER, textToAdd, paramText);
```

この例ではコンテントに対してテキスト挿入をしています。

第10章 コードの書き方

10-1　PDF ドキュメントの作成

概説：

『PDF Tool API』で実施可能な加工例：

- 空白 PDF を新規に作成する
- 既存 PDF の指定ページを抽出し新規 PDF を作成する
- 既存の PDF のページを分割し、２つの PDF として保存する
- 画像から新規ページを作成する

空白 PDF を新規に作成する

```
//入力 PDF の取得は「PDF を開く」を参照
//ページコンテナ取得は「ページ情報を操作する」を参照

PtlPDFDocument newDoc = new PtlPDFDocument();   //新規ドキュメント用クラスの初期化
PtlPages pages = newDoc.getPages();             //新規ドキュメント用ページコンテナ
PtlPage newPage = new PtlPDFPage();             //新規ドキュメント用ページクラス
newPage.setViewBox(viewBoxRect);                //新規ページの表示矩形を指定

pages.append(newPage);

// 出力 PDF の保存は「PDF の保存」を参照
```

既存 PDF の指定ページを抽出し新規 PDF を作成する

```
//入力 PDF doc の情報取得は「PDF を開く」を参照
PtlPDFDocument doc_ext = new PtlPDFDocument(); //出力 PDF 用クラス
PtlPages pages_ext = doc_ext.getPages();       //出力 PDF 用ページコンテナ
```

```
pages_ext.append(doc, startPageNum, pagesToExtract, insertOption);

//出力用パラメータクラス outputFile の取得は「PDF の保存」を参照

doc_ext.save(outputFile);
```

画像から新規ページを作成する

```
PtlPDFDocument doc = new PtlPDFDocument();              //出力 PDF 用クラスの新規作成
//ページコンテナ取得は「ページ情報を操作する」を参照

PtlParamDrawImage paramDrawImage = new PtlParamDrawImage(); //出力画像描画パラメータ
PtlParamInput inputImage = new PtlParamInput(imagePath);    //出力画像指定用クラス
PtlParamImagePage paramImagePage = new PtlParamImagePage(); //画像でページを作るためのパラメータ
paramDrawImage.setImageStream(inputImage);
paramImagePage.setImage(paramDrawImage);
paramImagePage.setPaperType(PtlParamImagePage.PAPER_TYPE.PAPER_IMAGE_SIZE);
paramImagePage.setAlign(PtlParamImagePage.ALIGN.ALIGN_CENTER);
pages.append(paramImagePage);

// 出力 PDF の保存は「PDF の保存」を参照
```

参考：

『PDF Tool API』の解説本『PDF CookBook』では、本項に関連した内容として以下のようなプログラム例を提示しています。

・既存 PDF の指定ページを抽出し新規 PDF を作成する

『PDF CookBook（第 1 巻）1.1.3 ページの抽出』

https://www.antenna.co.jp/ptl/cookbook/vol1/i04-0006.html

・PDF を分割してそれぞれを別ファイルとして保存する

『PDF CookBook（第 1 巻）1.1.5 PDF 文書の分割』

https://www.antenna.co.jp/ptl/cookbook/vol1/i04-0007.html

・白紙のページを文書末尾に追加して保存する

『PDF CookBook（第 1 巻）1.1.6 白紙ページの挿入』

https://www.antenna.co.jp/ptl/cookbook/vol1/i04-0009.html

・画像から新規ページを作成する

『PDF CookBook（第 4 巻）4.1.1 ページの作成』

https://www.antenna.co.jp/ptl/cookbook/vol4/i02-0034.html

10-2　ページ構成の編集

概説：

- PDF 末尾に指定したページを追加する
- PDF の任意の位置に指定したページを追加する
- ２つの異なる PDF を結合して１つの PDF とする
- PDF から指定したページを削除する
- ページを移動して順序を入れ替える

PDF 末尾に指定したページを追加する

```
//入力 PDF の取得は「PDF を開く」を参照
//ページコンテナ取得は「ページ情報を操作する」を参照
PtlPage pageToAppend;   //用意された何らかのページ

pages.append(pageToAppend, opionFlag);

// 出力 PDF の保存は「PDF の保存」を参照
```

PDF の任意の位置に指定したページを追加する

```
//入力 PDF の取得は「PDF を開く」を参照
//ページコンテナ取得は「ページ情報を操作する」を参照
PtlPage pageToAppend;   //用意された何らかのページ

pages.insert(whereToInsert, pageToAppend, optionFlag);

// 出力 PDF の保存は「PDF の保存」を参照
```

2 つの異なる PDF を結合して 1 つの PDF とする

```
//入力 PDF の取得は「PDF を開く」を参照
//ページコンテナ取得は「ページ情報を操作する」を参照
//追加用 PDF の取得も「PDF を開く」を参照
pages.append(docToAppend, 0, PtlPages.PAGE_ALL, PtlPages.OPTION_COPY_OUTLINES);

// 出力 PDF の保存は「PDF の保存」を参照
```

PDF から指定したページを削除する

```
//入力 PDF の取得は「PDF を開く」を参照
//ページコンテナ取得は「ページ情報を操作する」を参照
pages.remove(removeStart, pagesToRemove);

// 出力 PDF の保存は「PDF の保存」を参照
```

ページを移動して順序を入れ替える

```
//入力 PDF の取得は「PDF を開く」を参照
//ページコンテナ・ページ取得は「ページ情報を操作する」を参照
pages.move(whereToMove, moveStart, movingPages);

// 出力 PDF の保存は「PDF の保存」を参照
```

各関数の詳細はリファレンスを参照してください。

参考：

・2 つの異なる PDF を結合して 1 つの PDF とする

『PDF CookBook（第 1 巻）1.1.2 PDF 文書の結合』

https://www.antenna.co.jp/ptl/cookbook/vol1/i04-0005.html

・PDF から指定したページを削除する

『PDF CookBook（第 1 巻）1.1.4 ページの削除』

https://www.antenna.co.jp/ptl/cookbook/vol1/i04-0008.html

・ページを移動して順序を入れ替える

『PDF CookBook（第 1 巻）1.1.7 ページの移動』

https://www.antenna.co.jp/ptl/cookbook/vol1/i04-0010.html

10-3　ページ属性の編集

概説：

- ページに境界値を設定する
- ページの拡大・縮小

ページに境界値を設定する

```
//入力 PDF の取得は「PDF を開く」を参照
//ページコンテナ・ページ取得は「ページ情報を操作する」を参照
PtlRect mediaBox = new PtlRect(left, bottom, right, top);
PtlRect cropBox = new PtlRect(left, bottom, right, top);
PtlRect bleedBox = new PtlRect(left, bottom, right, top);
PtlRect trimBox = new PtlRect(left, bottom, right, top);
PtlRect artBox = new PtlRect(left, bottom, right, top);

page.setMediaBox(mediaBox);
page.setCropBox(cropBox);
page.setBleedBox(bleedBox);
page.setTrimBox(trimBox);
page.setArtBox(artBox);

// 出力 PDF の保存は「PDF の保存」を参照
```

ページの拡大・縮小

```
//入力 PDF の取得は「PDF を開く」を参照
//ページコンテナ・ページ取得は「ページ情報を操作する」を参照
page.zoom(zoomingRatio);

// 出力 PDF の保存は「PDF の保存」を参照
```

参考：

『PDF Tool API』の解説本『PDF CookBook』では、以下のようなプログラム例を提示しています。

・ページに境界値を設定する

『PDF CookBook（第 1 巻）1.2.2 ページ境界値の設定』

https://www.antenna.co.jp/ptl/cookbook/vol1/i01-0003.html

・ページの拡大・縮小

『PDF CookBook（第 1 巻）1.2.6 ページの拡大・縮小』

https://www.antenna.co.jp/ptl/cookbook/vol1/i01-0007.html

10-4　コンテントの編集

概説：

- ページ上にテキストを追加
- ページ上に画像を描画
- 画像描画の際に角度を指定する
- 別 PDF のページを貼り付け
- ページ上に直線を描画

ページ上にテキストを追加

```
//入力 PDF の取得は「PDF を開く」を参照
//ページコンテナ・ページ・ページコンテント取得は「ページコンテントを操作する」を参照
PtlParamWriteString stringParam = new PtlParamWriteString();    //文字描画用パラメータ
PtlRect outputRect = new PtlRect(left, bottom, right, top);
content.writeString(outputRect, align, textToAdd, stringParam);

// 出力 PDF の保存は「PDF の保存」を参照
```

PtlParamWriteString は上記以外に様々なパラメータを指定可能

```
PtlParamFont font = new PtlParamFont(); //フォント指定用クラス。詳細は「テキストのフォントを指定」を参照
PtlColorRGB outlineColor = new PtlColorRGB(red, green, blue);
PtlColorRGB textColor = new PtlColorRGB(red, green, blue);

stringParam.setFont(font);                  //フォントを設定。
stringParam.setOpacity(opacity);      //不透明度を設定。
stringParam.setOutlineColor(outlineColor);  //文字の縁取り色を設定。
stringParam.setTextColor(textColor);        //文字色を設定。
```

テキストのフォントを指定

```
//入力 PDF の取得は「PDF を開く」を参照
//ページコンテナ・ページ・ページコンテント取得は「ページコンテントを操作する」を参照
//テキスト挿入用パラメータ stringParam 取得は「ページ上にテキストを追加」を参照
PtlParamFont font = new PtlParamFont(); //フォント指定用クラス。
font.setName(fontFamily);              //フォントファミリー名
font.setSize(fontSize);                //フォントのサイズ(ポイント)
font.setEmbed(true);                   //フォント埋め込みの可否
font.setItalic(false);                 //イタリックか否か

//テキストの挿入は「ページ上にテキストを追加」を参照
// 出力 PDF の保存は「PDF の保存」を参照
```

テキストボックスを用いてテキストを追加

```
//入力 PDF の取得は「PDF を開く」を参照
//ページコンテナ・ページ・ページコンテント取得は「ページコンテントを操作する」を参照

PtlRect outputRect = new PtlRect(left, bottom, right, top);
//テキストボックスの描画用パラメータ
PtlTextBox paramTextBox = content.drawTextBox(outputRect, align, width, height);
//挿入するテキスト用パラメータ
PtlParamWriteStringTextBox textParam = new PtlParamWriteStringTextBox();
paramTextBox.writeString(textToAdd, textParam);    //テキストボックスへの文字列の追加
paramTextBox.terminate();                          //テキストボックスのコンテントへの描画処理

// 出力 PDF の保存は「PDF の保存」を参照
```

PtlTextBox は上記以外に様々なパラメータを指定可能

```
PtlColorRGB backColor = new PtlColorRGB(red, green, blue);
PtlColorRGB outlineColor = new PtlColorRGB(red, green, blue);

textBox.fitToBBox(true);              //テキストボックスのサイズを文字列の BBox に合わせるかの設定
```

```
textBox.setBackColor(backColor);        //矩形の背景色の指定
textBox.setOutlineColor(outlineColor);  //矩形の縁取りの色の指定
textBox.setOpacity(opacity);            //テキストボックスの不透明度
```

PtlParamWriteStringTextBox もまた上記以外に様々なパラメータを指定可能

```
PtlParamFont font = new PtlParamFont(); //フォント指定用クラス。詳細は「テキストのフォントを指定」を参照
PtlColorRGB outlineColor = new PtlColorRGB(red, green, blue);
PtlColorRGB textColor = new PtlColorRGB(red, green, blue);

textParam.setFont(font);                    //パラメータにフォントを設定
textParam.setOutlineColor(outlineColor);    //文字の背景色の指定
textParam.setTextColor(textColor);          //文字の色指定
textParam.setOpacity(opacity);              //文字の透明度の指定
textParam.setFormat(format);                //文字描画の体裁
textParam.setStrikeOut(true);               //打消し線の有無
textParam.setUnderline(true);               //下線の有無
```

ページ上に画像を描画

```
//入力 PDF の取得は「PDF を開く」を参照
//ページコンテナ・ページ・ページコンテント取得は「ページコンテントを操作する」を参照
PtlParamInput insertImage = new PtlParamInput(imagePath);   //画像のパス指定用パラメータ
PtlParamDrawImage paramDrawImage = new PtlParamDrawImage(); //出力画像描画パラメータ
PtlRect outputRect = new PtlRect(left, bottom, right, top);
paramDrawImage.setImageStream(insertImage);
content.drawImage(outputRect, PtlContent.ALIGN.ALIGN_CENTER, paramDrawImage);

// 出力 PDF の保存は「PDF の保存」を参照
```

PtlParamDrawImage は上記以外に以下のように様々なパラメータを指定可能

```
paramDrawImage.setAngle(angle); //角度指定（任意の数値）
paramDrawImage.setDPI(dpi); //描画時の DPI を指定
```

```
paramDrawImage.setUseOriginalDPI(false);//画像が元々持つ DPI を使うか否かの設定
paramDrawImage.setImagePageNumber(imagePageNumber);    //マルチ Tiff だった場合のページ番号を指定
paramDrawImage.setOpacity(opacity);    //透明度を指定
//描画画像にかけるソフトマスクとして画像 fileSoftMask を指定
paramDrawImage.setMaskImageStream(fileSoftMask, PtlParamDrawImage.MASK_TYPE.MASK_SOFT);
```

別 PDF のページを貼り付け

```
//入力 PDF の取得は「PDF を開く」を参照
//ページコンテナ・ページ・ページコンテント取得は「ページコンテントを操作する」を参照

PtlParamInput insertPdf = new PtlParamInput(insertPdfPath); //挿入する PDF 指定用クラス
PtlPDFDocument docToInsert = new PtlPDFDocument();          //挿入 PDF 用クラス
docToInsert.load(insertPdf);
PtlPages pagesToInsert = docToInsert.getPages();            //挿入する PDF のページコンテナ
PtlPage pageToInsert = pagesToInsert.get(insertPageNum);    //挿入する PDF のページ
PtlParamDrawForm formInsert = new PtlParamDrawForm();       //挿入するページ描画用パラメータ
formInsert.setPage(pageToInsert);

// 出力 PDF の保存は「PDF の保存」を参照
```

PtlParamDrawForm は以下のパラメータを指定可能

```
formInsert.setAngle(float angle);       //任意の傾きを設定。

formInsert.setOpacity(float opacity)    //不透明度を設定
```

ページ上に図形を描画

```
//入力 PDF の取得は「PDF を開く」を参照
//ページコンテナ・ページ・ページコンテント取得は「ページコンテントを操作する」を参照

PtlPoint startPoint = new PtlPoint(startX, startY);
```

```
PtlPoint endPoint = new PtlPoint(endX, endY);
PtlParamDrawShape paramDrawShape = new PtlParamDrawShape();
PtlColorDeviceRGB colorRGB = new PtlColorDeviceRGB(colorR, colorG, colorB);

paramDrawShape.setLineColor(colorRGB);
paramDrawShape.setOpacity(opacity);
content.drawLine(startPoint, endPoint, paramDrawShape);

// 出力 PDF の保存は「PDF の保存」を参照
```

上記はページ上に線を描画した例

PtlParamDrawShape は以下のパラメータを指定可能

```
PtlColorDeviceRGB colorFill = new PtlColorDeviceRGB(red, green, blue);
PtlColorDeviceRGB colorStroke = new PtlColorDeviceRGB(red, green, blue);

paramDrawShape.setFillColor(colorFill);          //塗りつぶし色を設定。
paramDrawShape.setLineColor(colorStroke);        //線の色を設定。
paramDrawShape.setLineStyle(lineStyle);          //線スタイルを設定。
paramDrawShape.setLineWidth(lineWidth);          //線幅を設定。
paramDrawShape.setOpacity(float opacity);        //不透明度を設定。
```

また、PtlParamDrawShape を用いて以下の図形も描画可能

```
PtlPoint center = new PtlPoint(startX, startY);              //円の中心
content.drawCircle(center, radius, paramDrawShape);          //円を描画

PtlRect rect = new PtlRect(left, bottom, right, top);        //矩形の描画範囲
content.drawRect(PtlRect rectMM, PtlParamDrawShape paramDrawShape); //矩形を描画

PtlRect roundRect = new PtlRect(left, bottom, right, top);          //丸角矩形の描画範囲
content.drawRoundRect(roundRect, widthRound, heightRound, paramDrawShape);  //丸角矩形を描画
```

参考：

『PDF Tool API』の解説本『PDF CookBook』では、以下のようなプログラム例を提示しています。

・ページ上にテキストを追加

『PDF CookBook（第 1 巻）2.1.1 本文テキストの追加』

https://www.antenna.co.jp/ptl/cookbook/vol1/i02-0003.html

・ページ上に画像を描画

『PDF CookBook（第 1 巻）2.2.1 ページ上に画像を描画』

https://www.antenna.co.jp/ptl/cookbook/vol1/i03-0002.html

・画像描画の際に角度を指定する

『PDF CookBook（第 5 巻）5.1.1 画像を任意の角度で描画する』

https://www.antenna.co.jp/ptl/cookbook/vol5/i03-0032.html

・別 PDF のページを貼り付け

『PDF CookBook（第 1 巻）2.3.1 PDF のページを貼り付け』

https://www.antenna.co.jp/ptl/cookbook/vol1/i05-0002.html

・ページ上に図形を描画

『PDF CookBook（第 1 巻）2.4.1 パスで直線の描画』

https://www.antenna.co.jp/ptl/cookbook/vol1/i05-0007.html

10-5　注釈

概説：

『PDF Tool API』では注釈の作成の他、注釈の削除、注釈のエクスポート・インポートが可能です。

『PDF Tool API』で実施可能な加工例：

- 注釈コンテナの取得
- ラバースタンプ注釈の作成
- カスタムスタンプ注釈の作成
- リンク注釈の追加
- テキスト注釈の作成
- ファイル添付注釈の作成
- ハイライト注釈の作成
- テキスト検索してハイライト注釈を付ける
- 注釈タイプの判別をして情報を取得する
- 注釈の削除
- 注釈情報のエクスポート：FDF ファイル
- 注釈情報のインポート：FDF ファイル
- 注釈情報のインポート：PDF ファイル

注釈の基本要素

注釈コンテナの取得と注釈の追加

```
//入力 PDF の取得は「PDF を開く」を参照
//ページコンテナ・ページ取得は「ページ情報を操作する」を参照
PtlAnnots annots = page.getAnnots();        //注釈コンテナの取得

//ここには追加する予定の注釈を表すクラス newAnnot パラメータを設定する記述が入る
annots.append(newAnnot);                    //注釈コンテナに注釈を追加する

// 出力 PDF の保存は「PDF の保存」を参照
```

ポップアップ注釈の指定

マークアップ注釈の子クラスである注釈を作成する場合、それが持つポップアップ注釈のパラメータを指定できる。

```
//入力 PDF の取得は「PDF を開く」を参照
//ページコンテナ・ページ取得は「ページ情報を操作する」を参照
//注釈コンテナの取得は「注釈コンテナの取得と注釈の追加」を参照
//注釈を表すクラス newAnnot の取得はそれぞれの注釈の解説を参照
PtlAnnotPopup annotPopup = new PtlAnnotPopup();             //ポップアップ注釈
PtlRect rectPopup = new PtlRect(left, bottom, right, top); //ポップアップを開く先の矩形
annotPopup.setRect(rectPopup);          //矩形の設定
annotPopup.setOpenState(true);          //PDF を開いた際にポップアップが開いているか否か

newAnnot.setMarkUpTitle(title);         //ポップアップウィンドウのタイトル文字列
newAnnot.setMarkUpSubj(subject);        //サブジェクトの短い説明
newAnnot.setTextContents(textContents); //ポップアップ内の本文指定
newAnnot.setAnnotPopUp(annotpopup);     //ポップアップ注釈の設定
//注釈の追加は「注釈コンテナの取得と注釈の追加」を参照
```

注釈の応用

ラバースタンプ注釈の作成

```
//入力 PDF の取得は「PDF を開く」を参照
//ページコンテナ・ページ取得は「ページ情報を操作する」を参照
```

```
//注釈コンテナの取得は「注釈コンテナの取得と注釈の追加」を参照
PtlAnnotStamp stampAnnot = new PtlAnnotStamp();                //スタンプ注釈
PtlRect outputRect = new PtlRect(left, bottom, right, top);     //注釈の矩形
stampAnnot.setRect(rect);                                      //矩形の設定
stampAnnot.setIconType(PtlAnnotStamp.ICON_TYPE.ICON_APPROVED); //スタンプの模様の指定

//注釈の追加は「注釈コンテナの取得と注釈の追加」を参照
// 出力 PDF の保存は「PDF の保存」を参照
```

カスタムスタンプ注釈の追加

```
//入力 PDF の取得は「PDF を開く」を参照
//ページコンテナ・ページ取得は「ページ情報を操作する」を参照
//注釈コンテナの取得は「注釈コンテナの取得と注釈の追加」を参照
//スタンプ注釈用クラスの作成は「ラバースタンプ注釈の作成」を参照
stampAnnot.setIconType(PtlAnnotStamp.ICON_TYPE.ICON_CUSTOM); //スタンプの模様の指定
PtlPage pageCustomStamp; //カスタムスタンプにするための何らかのページを用意する
stampAnnot.setPage(pageCustomStamp);

//注釈の追加は「注釈コンテナの取得と注釈の追加」を参照
// 出力 PDF の保存は「PDF の保存」を参照
```

リンク注釈の追加

```
//入力 PDF の取得は「PDF を開く」を参照
//ページコンテナ・ページ取得は「ページ情報を操作する」を参照
//注釈コンテナの取得は「注釈コンテナの取得と注釈の追加」を参照
PtlAnnotLink annotLink = new PtlAnnotLink();       //リンク注釈
PtlActionGoTo acttiongoto = new PtlActionGoTo();    //GoTo アクション
PtlDestFit destfit = new PtlDestFit();             //アクションの宛先用クラス
PtlRect outputRect = new PtlRect(left, bottom, right, top); //リンク注釈の指定範囲

destfit.setPageNumber(pages.getCount() - 1);//ここでは最終ページを宛先に指定
acttiongoto.setDest(destfit);              //指定した宛先をアクションに設定
annotLink.setAction(acttiongoto);           //リンク注釈にアクションを設定
annotLink.setRect(rect);                   //リンク注釈の矩形指定
```

```
//注釈の追加は「注釈コンテナの取得と注釈の追加」を参照
// 出力 PDF の保存は「PDF の保存」を参照
```

テキスト注釈の作成

```
//入力 PDF の取得は「PDF を開く」を参照
//ページコンテナ・ページ取得は「ページ情報を操作する」を参照
//注釈コンテナの取得は「注釈コンテナの取得と注釈の追加」を参照
PtlAnnotText annotText = new PtlAnnotText();              //テキスト注釈
PtlRect rectAnnot = new PtlRect(left, bottom, right, top); //テキスト注釈アイコンの位置
annotText.setRect(rectAnnot);                             //アイコンの矩形を指定
annotText.setIconType(PtlAnnotText.ICON_TYPE.ICON_COMMENT); //テキスト注釈用アイコン形式指定
//ポップアップ注釈のパラメータ指定は「ポップアップ注釈の指定」を参照
//（テキスト注釈の中身は「ポップアップ注釈の指定」の setTextContents()で指定される

//注釈の追加は「注釈コンテナの取得と注釈の追加」を参照
// 出力 PDF の保存は「PDF の保存」を参照
```

ファイル添付注釈の作成

```
//入力 PDF の取得は「PDF を開く」を参照
//ページコンテナ・ページ取得は「ページ情報を操作する」を参照
//注釈コンテナの取得は「注釈コンテナの取得と注釈の追加」を参照
PtlAnnotFileAttachment annotFileAttachment = new PtlAnnotFileAttachment(); //ファイル添付注釈
PtlParamInput attachmentFile = new PtlParamInput(attachmentFilePath);      //添付ファイルのパス
PtlRect rectAnnot = new PtlRect(left, bottom, right, top);                 //注釈アイコンの矩形

annotFileAttachment.setFileName(attachmentFileName);                       //表示されるファイル名
annotFileAttachment.readFile(attachmentFile);                              //添付するファイル指定
annotFileAttachment.setRect(rectAnnot);                                    //アイコンの矩形を指定
annotFileAttachment.setIconType(PtlAnnotFileAttachment.ICON_TYPE.ICON_PUSHPIN); //アイコン形式指定
//ポップアップ注釈のパラメータ指定は「ポップアップ注釈の指定」を参照

//注釈の追加は「注釈コンテナの取得と注釈の追加」を参照
// 出力 PDF の保存は「PDF の保存」を参照
```

ハイライト注釈の作成

```
//入力 PDF の取得は「PDF を開く」を参照
//ページコンテナ・ページ取得は「ページ情報を操作する」を参照
//注釈コンテナの取得は「注釈コンテナの取得と注釈の追加」を参照
PtlAnnotHighlight annotHighlight = new PtlAnnotHighlight();     //ハイライト注釈
PtlColorDeviceRGB color = new PtlColorDeviceRGB(colorR, colorG, colorB);
annotHighlight.setColor(color);                 //ハイライト注釈の色指定
annotHighlight.setMarkUpCA(opacity);            //ハイライト注釈の不透明度指定

PtlQuadPoints annotQuadPoints = annotHighlight.getQuadPoints(); //注釈が持つ QuadPoint コンテナ
PtlPoint topLeft = new PtlPoint(leftX, topY);                   //QuadPoint の四隅（上左）
PtlPoint topRight = new PtlPoint(rightX, topY);                 //QuadPoint の四隅（上右）
PtlPoint bottomLeft = new PtlPoint(leftY, bottomY);             //QuadPoint の四隅（下左）
PtlPoint bottomRight = new PtlPoint(rightX, bottomY);           //QuadPoint の四隅（下右）
PtlQuadPoint outputQuadPoint = new PtlQuadPoint(topLeft, topRight, bottomLeft, bottomRight); //注釈用 QuadP
oint
annotQuadPoints.append(outputQuadPoint);                        //QuadPoint の追加

//ポップアップ注釈のパラメータ指定は「ポップアップ注釈の指定」を参照
//注釈の追加は「注釈コンテナの取得と注釈の追加」を参照
// 出力 PDF の保存は「PDF の保存」を参照
```

文字列検索してハイライト注釈を付ける

```
//入力 PDF の取得は「PDF を開く」を参照
//ページコンテナ・ページ取得は「ページ情報を操作する」を参照

//検索設定指定用クラス
PtlParamSearchTextAndHighlight paramSearchTextHighlight = new PtlParamSearchTextAndHighlight();
PtlColorDeviceRGB color = new PtlColorDeviceRGB(colorR, colorG, colorB);
paramSearchTextHighlight.appendText(textToSearch);  //ハイライト注釈に指定したいキーワードを指定
paramSearchTextHighlight.setColor(color);           //ハイライト注釈の色指定
paramSearchTextHighlight.setOpacity(opacity);       //ハイライト注釈の不透明度指定

//ドキュメント全体を検索し、ヒットした部分にハイライト挿入実行
```

```
doc.searchTextAndDoProcess(paramSearchTextHighlight);
```

指定したページだけ検索することもできます。

```
//ページを検索し、ヒットした部分にハイライト挿入実行
page.searchTextAndDoProcess(paramSearchTextHighlight);
```

注釈タイプの判別をして情報を取得する

```
//入力 PDF の取得は「PDF を開く」を参照
//ページコンテナ・ページ取得は「ページ情報を操作する」を参照
//注釈コンテナの取得は「注釈コンテナの取得と注釈の追加」を参照

PtlAnnot annot = annots.get(annotNumToGetInfo);     //注釈の取得
annot.getTextContents();
annot.getRect();
annot.getDate();
annot.getAnnotFlags();
annot.getBorderStyle();
annot.getBorderWidth();

annot.isMarkup();           //マークアップ注釈であるか否かの判別
PtlAnnotMarkup annotMarkup = (PtlAnnotMarkup)annot;
annotMarkup.getMarkUpTitle();
annotMarkup.getMarkUpSubj();
annotMarkup.getMarkUpDate();

annotMarkup.hasAnnotPopup(); //ポップアップ注釈が指定済みか否かの判別
PtlAnnotPopup popup = annotMarkup.getAnnotPopup();
popup.getOpenState();

annot.getType();            //注釈種別の取得

//テキスト注釈の場合
PtlAnnotText annotText = (PtlAnnotText)annot;
annotText.getIconType();
```

```
annotText.getIconName();

//リンク注釈の場合
PtlAnnotLink annotLink = (PtlAnnotLink)annot;
annotLink.getHighlightMode();
annotLink.getDest();
annotLink.getAction();

//スタンプ注釈の場合
PtlAnnotStamp annotStamp = (PtlAnnotStamp)annot;
annotStamp.getIconType();
annotStamp.getIconName();

//ファイル添付注釈の場合
PtlAnnotFileAttachment annotFileAttachment = (PtlAnnotFileAttachment)annot;
annotFileAttachment.getIconType();
annotFileAttachment.getIconName();
annotFileAttachment.getFileName();
PtlParamOutput outputAttach = new PtlParamOutput(attachmentOutputPath);
annotFileAttachment.writeFile(outputAttach);
```

注釈の削除

```
//入力 PDF の取得は「PDF を開く」を参照
//ページコンテナ・ページ取得は「ページ情報を操作する」を参照
//注釈コンテナの取得は「注釈コンテナの取得と注釈の追加」を参照
annots.remove(annotNumToRemove);

// 出力 PDF の保存は「PDF の保存」を参照
```

注釈情報のエクスポート：FDF ファイル

```
//入力 PDF の取得は「PDF を開く」を参照
PtlParamOutput outputFile = new PtlParamOutput(outputFDFPath);  //エクスポート先のパス
doc.exportAnnotsToFDF(outputFile);                              //パスを指定してエクスポート
```

注釈情報のインポート：FDF ファイル

```
//入力 PDF の取得は「PDF を開く」を参照
PtlParamInput FdfToImport = new PtlParamInput(inportFDFPath);   //インポート元のパス
doc.importAnnotsFromFDF(FdfToImport);                          //FDF ファイルから注釈取り込み

// 出力 PDF の保存は「PDF の保存」を参照
```

注釈情報のインポート：PDF ファイル

```
//入力 PDF の取得は「PDF を開く」を参照
PtlParamInput PdfToImport = new PtlParamInput(importPDFPath);   //インポート元のパス
doc.importAnnotsFromPDF(PdfToImport);                          //PDF ファイルから注釈取り込み

// 出力 PDF の保存は「PDF の保存」を参照
```

参考：

『PDF Tool API』の解説本『PDF CookBook』では、以下のようなプログラム例を提示しています。

・ラバースタンプ注釈の作成

『PDF CookBook（第 2 巻）1.1.1 ラバースタンプ注釈の作成』

https://www.antenna.co.jp/ptl/cookbook/vol4/i02-0004.html

・カスタムスタンプ注釈の作成

『PDF CookBook（第 2 巻）1.1.2 カスタムスタンプ注釈作成（画像・PDF 設定）』

https://www.antenna.co.jp/ptl/cookbook/vol4/i02-0005.html

・リンク注釈の追加

『PDF CookBook（第 2 巻）1.1.4 リンク注釈の追加』

https://www.antenna.co.jp/ptl/cookbook/vol4/i02-0007.html

・テキスト注釈の作成

『PDF CookBook（第 2 巻）1.1.5 テキスト注釈の作成』

https://www.antenna.co.jp/ptl/cookbook/vol4/i02-0008.html

・ファイル添付注釈の作成

『PDF CookBook（第 2 巻）1.1.7 ファイル添付注釈の作成』

https://www.antenna.co.jp/ptl/cookbook/vol4/i02-0010.html

・ハイライト注釈の作成

『PDF CookBook（第 2 巻）1.1.8 ハイライト注釈の作成』

https://www.antenna.co.jp/ptl/cookbook/vol4/i02-0011.html

・テキスト検索してハイライト注釈を付ける

『PDF CookBook（第 2 巻）1.1.9 テキスト検索してハイライト注釈』

https://www.antenna.co.jp/ptl/cookbook/vol4/i02-0012.html

・注釈タイプの判別をして情報を取得する

『1.2.1 注釈の個数・全ての注釈に共通した項目の情報を取得』

https://www.antenna.co.jp/ptl/cookbook/vol4/i02-0015.html

『1.2.2 マークアップ注釈の情報を取得』

https://www.antenna.co.jp/ptl/cookbook/vol4/i02-0016.html

『PDF CookBook（第 2 巻）1.2.3 注釈タイプの判別をして特有の情報を取得する』

https://www.antenna.co.jp/ptl/cookbook/vol4/i02-0017.html

・注釈の削除

『PDF CookBook（第 2 巻）1.3.1 注釈の削除』

https://www.antenna.co.jp/ptl/cookbook/vol4/i02-0019.html

・注釈情報のエクスポート：FDF ファイル

『PDF CookBook（第 2 巻）2.1.1 注釈情報を FDF ファイルにエクスポート』

https://www.antenna.co.jp/ptl/cookbook/vol4/i02-0022.html

・注釈情報のインポート：FDF ファイル

『PDF CookBook（第 2 巻）2.1.2 FDF ファイルの注釈情報をインポート』

https://www.antenna.co.jp/ptl/cookbook/vol4/i02-0023.html

・注釈情報のインポート：PDF ファイル

『PDF CookBook（第 2 巻）2.2.1 PDF ファイルからの注釈データインポート』

https://www.antenna.co.jp/ptl/cookbook/vol4/i02-0025.html

10-6　しおり

概説：

- しおり情報の取得
- しおりの追加
- しおりの個別削除
- しおりの一括削除

しおりの基本要素

ルートしおりの取得

しおりに関する処理を実行する場合、最初に文書からルートしおりを取得する必要があります。

```
//入力 PDF doc の取得は「PDF を開く」を参照
PtlOutline rootOutline = doc.getRootOutline();      //ルートしおりを PtlOutline クラスの形で取得
```

子しおりの取得

子しおりを持っている場合は、子しおりを取得して関係をたぐっていくことができます。

```
//入力 PDF の取得は「PDF を開く」を参照
```

```
//ルートしおりの取得は「ルートしおりの取得」を参照
parentOutline.hasChild();
PtlOutline outline = parentOutline.getFirstChild(); //最初の子しおりを PtlOutline クラスの形で取得
```

兄弟しおりの取得

しおりが、それに続く兄弟しおりを持っている場合はその兄弟しおりを取得して関係をたぐっていくことができます。

```
//入力 PDF doc の取得は「PDF を開く」を参照
//ルートしおりやその子しおりの取得は「ルートしおりの取得」「子しおりの取得」を参照
outline.hasNextSibling();                    //兄弟しおりを持っているか否かの判別
PtlOutline nextSibling = outline.getNextSibling(); //兄弟しおりを PtlOutline の形で取得
```

しおりの応用

しおり情報の取得

```
//入力 PDF doc の取得は「PDF を開く」を参照
//ルートしおりと、それに連なる目的の子しおり outline の取得は「ルートしおりの取得」「子しおりの取得」「兄弟しおりの取得」を参照
outline.getTitle();     //しおりのタイトル取得
outline.isOpen();       //しおりが開いているか否かを取得
outline.getFlags();     //しおりが持つフラグを取得
outline.getColor();     //しおりの色を取得
outline.getAction();    //しおり押下時のアクションを取得
```

しおりの追加

```
//入力 PDF の取得は「PDF を開く」を参照
//ルートしおり outlineRoot の取得は「ルートしおりの取得」を参照
PtlOutline outlineActionGoToTop = new PtlOutline();        //設定するしおり
PtlColorDeviceRGB color =  new PtlColorDeviceRGB(r, g, b);
PtlActionGoTo actionGoTo = new PtlActionGoTo();            //しおり押下時のアクション
```

```
PtlDestFit destFit = new PtlDestFit();                    //しおり押下時の宛先
destFit.setPageNumber(pageNum);                           //アクションの宛先ページ
actionGoTo.setDest(destFit);                              //アクションへの宛先設定

outlineActionGoToTop.setAction(actionGoTo);               //しおり押下時のアクションを設定
outlineActionGoToTop.setColor(color);                     //しおりの色を設定
outlineActionGoToTop.setTitle(title);                     //しおりのタイトルを設定
outlineActionGoToTop.setFlags(PtlOutline.FLAG_BOLD);      //しおりのフラグを設定
outlineActionGoToTop.setOpen(isOpen);                     //しおりが開いているか否かを設定

outlineRoot.appendLastChild(outlineActionGoToTop);        //ルートしおりの最後の子しおりとして追加

// 出力 PDF の保存は「PDF の保存」を参照
```

しおりの個別削除

```
//入力 PDF doc の取得は「PDF を開く」を参照
//ルートしおり及び目的の子しおり outline の取得は「ルートしおりの取得」「子しおりの取得」「兄弟しおりの取得」を参照
outline.destroy();              //自分及び自分につらなる子しおりの全てを削除

// 出力 PDF の保存は「PDF の保存」を参照
```

しおり自体を削除する他、しおりの子だけを削除する、しおりのアクションを削除するなども可能

```
outline.removeAction();         //アクションの削除
outline.removeChildren();       //自分につらなる子しおりを削除
```

しおりの一括削除

```
//入力 PDF doc の取得は「PDF を開く」を参照
PtlParamOptimize paramOptimize = new PtlParamOptimize();    //最適化オプション設定用クラス
paramOptimize.setRemoveOutlines(true);                      //しおりの削除設定をオンにする
doc.optimize(paramOptimize);                                //最適化を実行する

// 出力 PDF の保存は「PDF の保存」を参照
```

参考：

・しおり情報の取得

『PDF CookBook（第 2 巻）3.1.1 しおり情報の取得』

https://www.antenna.co.jp/ptl/cookbook/vol2/i04-0003.html

・しおりの追加

『PDF CookBook（第 2 巻）3.1.2 しおりの追加』

https://www.antenna.co.jp/ptl/cookbook/vol2/i04-0003.html

・しおりの個別削除

『PDF CookBook（第 2 巻）3.1.3 しおりの削除』

https://www.antenna.co.jp/ptl/cookbook/vol2/i04-0003.html

・しおりの一括削除

『PDF CookBook（第 3 巻）5.2.2 しおりの削除』

https://www.antenna.co.jp/ptl/cookbook/vol3/i01-0048.html

10-7　フォームデータのインポート / エクスポート

概説：

- フォームデータの FDF へのエクスポート
- フォームデータの FDF からのインポート
- フォームデータの XFDF へのエクスポート
- フォームデータの XFDF からのインポート

フォームデータの FDF へのエクスポート

```
//入力 PDF doc の取得は「PDF を開く」を参照
PtlParamOutput outputFDF = new PtlParamOutput(exportFDFPath);   //エクスポート先のパス
doc.exportFormFieldsToFDF(outputFDF);                          //パスを指定してエクスポート
```

フォームデータの FDF からのインポート

```
//入力 PDF doc の取得は「PDF を開く」を参照
PtlParamInput FdfToImport = new PtlParamInput(importFDFPath);   //インポート元のパス
doc.importFormFieldsFromFDF(FdfToImport);                       //FDF ファイルからインポート

// 出力 PDF の保存は「PDF の保存」を参照
```

なお、何らかの理由でインポート時に設定できなかったフォームフィールドがあった場合、その情報を取得することができます。

```
//フォームデータインポート時のエラー情報コンテナを取得
PtlFormFieldValues failedFormFieldValues = doc.importFormFieldsFromFDF(FdfToImport);
//エラー情報コンテナからインデックス番号を指定して取り出し
PtlFormFieldValue formFieldValue = formFieldValues.get(failedFormFieldNum);
```

```
formFieldValue.getFieldName();      //エラー番号
formFieldValue.getValue();          //エラー値
```

フォームデータの XFDF へのエクスポート

```
//入力 PDF doc の取得は「PDF を開く」を参照
PtlParamOutput outputXFDF = new PtlParamOutput(exportXFDFPath); //エクスポート先のパス
doc.exportFormFieldsToXFDF(outputXFDF);                         //パスを指定してエクスポート
```

フォームデータの XFDF からのインポート

```
//入力 PDF doc の取得は「PDF を開く」を参照
PtlParamInput XFdfToImport = new PtlParamInput(importXFDFPath); //インポート元のパス
doc.importFormFieldsFromXFDF(XFdfToImport);                     //XFDF ファイルからインポート

// 出力 PDF の保存は「PDF の保存」を参照
```

参考：

・フォームデータの FDF へのエクスポート

『PDF CookBook（第 5 巻）7.1.1 FDF を用いた PDF フォームデータのエクスポート』

https://www.antenna.co.jp/ptl/cookbook/vol5/i03-0048.html

・フォームデータの FDF からのインポート

『PDF CookBook（第 5 巻）7.1.2 FDF を用いた PDF フォームデータのインポート』

https://www.antenna.co.jp/ptl/cookbook/vol5/i03-0049.html

・フォームデータの XFDF へのエクスポート

『PDF CookBook（第 5 巻）7.2.1 XFDF を用いた PDF フォームデータのエクスポート』

https://www.antenna.co.jp/ptl/cookbook/vol5/i03-0051.html

・フォームデータの XFDF からのインポート

『PDF CookBook（第 5 巻）7.2.2 XFDF を用いた PDF フォームデータのインポート』

https://www.antenna.co.jp/ptl/cookbook/vol5/i03-0052.html

10-8　要素の抽出

概説：

- ページ全体からのテキスト抽出
- 矩形を指定してのテキスト抽出
- 特定文字の置換をしたテキスト抽出
- 重なり合う文字の制御をしたテキスト抽出
- 画像の抽出
- 埋め込まれた画像の画質のままに抽出

ページ全体からのテキスト抽出

```
//入力 PDF の取得は「PDF を開く」を参照
//ページコンテナ・ページ・ページコンテント取得は「ページコンテントを操作する」を参照
PtlParamExtractText plainParam = new PtlParamExtractText(); //抽出時のパラメータ
String TextFromPdf = content.extractText(plainParam);       //コンテントから文字列を取得
```

矩形を指定してのテキスト抽出

```
//入力 PDF の取得は「PDF を開く」を参照
//ページコンテナ・ページ・ページコンテント取得は「ページコンテントを操作する」を参照
//パラメータクラスの準備は「ページ全体からのテキスト抽出」を参照
PtlParamExtractText paramExtractText = new PtlParamExtractText(); //抽出時のパラメータ
PtlRect extractRect = new PtlRect(left, bottom, right, top);      //抽出範囲の矩形
paramExtractText.appendRect(extractRect);                         //矩形を設定
paramExtractText.setTextOverlapRatio(overlapRatio);     //矩形の境界にある文字の重なり具合指定
paramExtractText.setIgnoreActualText(ignoreActualText); //ActualText を無視するか否かの指定

String TextFromPdf = content.extractText(paramExtractText); //コンテントから文字列を取得
```

特定文字のスペース置換・削除をしたテキスト抽出

特定の特殊文字はスペースに置き換える、もしくは削除することができます。

```
//入力 PDF の取得は「PDF を開く」を参照
//ページコンテナ・ページ・ページコンテント取得は「ページコンテントを操作する」を参照
//テキスト抽出パラメータクラスの準備は「矩形を指定してのテキスト抽出」を参照

int unicodeCharFlag = 0;       //置き換えたい文字指定用変数
//Space(U+0020)を指定に追加
unicodeCharFlag = unicodeCharFlag | PtlParamExtractText.AHEXTRACTTEXT_UNI_SPACE;
//NO-BREAK SPACE(U+00A0)を指定に追加
unicodeCharFlag = unicodeCharFlag | PtlParamExtractText.AHEXTRACTTEXT_UNI_NO_BREAK_SPACE ;
paramExtractText.setUnicodeToSpace(unicodeCharFlag);    //指定したフラグの特殊文字を Space に置換え

//テキスト抽出は「ページ全体からのテキスト抽出」を参照
```

削除したい場合は以下の関数を用います。

```
paramExtractText.setUnicodeToRemove(unicodeCharFlag);    //指定したフラグの特殊文字を削除
```

重なり合う文字の制御をしたテキスト抽出

同じ文字が重なっていたときに二重に読むのを防ぐ指定が可能です。

```
//入力 PDF の取得は「PDF を開く」を参照
//ページコンテナ・ページ・ページコンテント取得は「ページコンテントを操作する」を参照
//テキスト抽出パラメータクラスの準備は「矩形を指定してのテキスト抽出」を参照
paramExtractText.setSameTextOmitRatio(sameTextOmitRatio);    //一定以上重なった同じ文字を除外

//テキスト抽出は「ページ全体からのテキスト抽出」を参照
```

異なる文字が重なっていた時に二重に読むのを防ぐ指定も可能です。色が異なる場合だけ除外することもできます。

```
//一定以上重なった異なる文字を除外（onlyDiffColor が true だと色が同じ場合は除外しない）
paramExtractText.setDifferentTextOmitRatio(diffTextOmitRatio, onlyDiffColor);
```

画像の抽出

```
//入力 PDF の取得は「PDF を開く」を参照
//ページコンテナ・ページ・ページコンテント取得は「ページコンテントを操作する」を参照

//画像だけを指定したエレメントコンテナを取得
PtlEditElements elems = content.getEditElements(PtlContent.GET_IMAGE);
PtlEditElement elem = elems.get(imageIndexNum); //インデックス番号の画像エレメントを取得
PtlEditImage elemImage = (PtlEditImage)elem;    //画像エレメントにリキャスト
PtlParamOutput outputImage = new PtlParamOutput(outputImagePath);       //出力先のパス
//画像の出力フォーマット指定
PtlEditImage.OUTPUT_FORMAT format = PtlEditImage.OUTPUT_FORMAT.FORMAT_AUTO;

elemImage.writeFile(outputImage, format);   //画像の出力処理
```

出力する画像形式を BMP, JPEG, PNG から選択することもできます。

以下は PNG を指定した例です。

```
PtlEditImage.OUTPUT_FORMAT format = PtlEditImage.OUTPUT_FORMAT.FORMAT_PNG;
```

埋め込まれた画像の画質のままに抽出

```
//入力 PDF の取得は「PDF を開く」を参照
//ページコンテナ・ページ・ページコンテント取得は「ページコンテントを操作する」を参照
//画像エレメントの取得・出力画像の指定は「画像の抽出」を参照
elemImage.setPassThrough(true);              //パススルー設定をオンに指定

//画像の出力処理は「画像の抽出」を参照
```

特定条件を満たして PDF に埋め込まれていた画像（※）の場合、埋め込まれた画像をそのまま抽出することができます。

それ以外の場合は新規画像として出力されます。

（※）

- カラースペースが、DeviceRGB, DeviceGray, 無指定(=PDF_EMPTY_NAME)のいずれかで指定されていること
- マスクが指定されていないこと
- Decode の値がデフォルトであること
- グラフィックステートパラメータ辞書の『ExtGState:TR』が指定されていないこと

参考：

・ページ全体からのテキスト抽出

『PDF CookBook（第 3 巻）1.1.1 ページから全テキスト抽出』

https://www.antenna.co.jp/ptl/cookbook/vol3/i01-0009.html

・矩形を指定してのテキスト抽出

『PDF CookBook（第 3 巻）1.1.2 指定矩形からテキストを抽出』

https://www.antenna.co.jp/ptl/cookbook/vol3/i01-0010.html

・特定文字の置換をしたテキスト抽出

『PDF CookBook（第 5 巻）9.1 特殊文字の処理』

https://www.antenna.co.jp/ptl/cookbook/vol5/i03-0060.html

・重なり合う文字の制御をしたテキスト抽出

『PDF CookBook（第 5 巻）9.2 抽出における重なり合う文字の制御』

https://www.antenna.co.jp/ptl/cookbook/vol5/i03-0066.html

・画像の抽出

『PDF CookBook（第 3 巻）2.1.2 指定した画像を抽出』

https://www.antenna.co.jp/ptl/cookbook/vol3/i01-0014.html

『PDF CookBook（第 3 巻）2.1.3 出力画像形式の指定』

https://www.antenna.co.jp/ptl/cookbook/vol3/i01-0015.html

・埋め込まれた画像の画質のままに抽出

『PDF CookBook（第 5 巻）5.2.1 PDF に Exif 付き JPEG 画像を埋め込み、元の画像のまま抽出する』

https://www.antenna.co.jp/ptl/cookbook/vol5/i03-0034.html

10-9　文字列検索

概説：

- 文字列検索の基礎
- 文字列検索の細かいオプション
- 文字列検索をした部分に墨消しをして削除する
- 文字列検索をしてハイライト注釈をかける

文字列検索の基礎

文字検索

```
//入力 PDF の取得は「PDF を開く」を参照
PtlParamSearchText paramSearchText = new PtlParamSearchText(); //文字列検索設定用クラス

paramSearchText.appendText(textToSearch1);          //検索したい文字列その 1
paramSearchText.appendText(textToSearch2);          //検索したい文字列その 2

//文書全体から検索結果の取得
PtlSearchTextResults resultsContainer = doc.searchText(paramSearchText);
```

ページを指定しての文字列検索も可能です

```
//ページから検索結果の取得
PtlSearchTextResults resultsContainer = page.searchText(paramSearchText);
```

検索結果の取得

```
//入力 PDF の取得は「PDF を開く」を参照
//ページコンテナ・ページ取得は「ページ情報を操作する」を参照
//検索結果コンテナ resultsContainer の取得は「文字検索」を参照
//インデックス番号を指定して個別の検索結果を取得
PtlSearchTextResult searchResult = resultsContainer.get(resultNum);
searchResult.getPageNumber();    //ページ番号の取得
searchResult.getKeyword();       //キーワードの取得
```

検索結果の詳細座標取得

```
//入力 PDF の取得は「PDF を開く」を参照
//ページコンテナ・ページ取得は「ページ情報を操作する」を参照
//検索結果コンテナ resultsContainer の取得は「文字検索」を参照
//検索結果 searchResult の取得は「検索結果の取得」を参照

//検索結果の詳細コンテナ
PtlSearchTextResultDetails resultDetails = searchResult.getResultDetails();
//検索結果詳細の個別内容を取得
PtlSearchTextResultDetail resultDetail = resultDetails.get(detailNum);
PtlQuadPoint quadPoint = resultDetail.getQuadPoint();    //個別の詳細結果の QuadPoint を取得
```

文字列検索の細かいオプション

検索用パラメータ PtlParamSearchText は細かいオプションが設定可能である。

大文字・小文字を判別した上での文字列検索

```
paramSearchText.setCompareCase(compareCase);    //大文字・小文字を区別するか否かを指定
```

ActuralText の無視を指定した文字列検索

```
paramSearchText.setIgnoreActualText(ignoreActualText);  //ActualText を無視するか否かを指定
```

座標順にテキスト取得する文字列検索

```
//座標順にテキストを指定して検索
paramSearchText.setTextType(PtlParamSearchText.TEXT_TYPE.TEXT_SORT);
```

座標順にテキストを取得する場合、どの程度ずれた行まで同一行とみなすかを指定可能

```
//どの程度ずれた行まで同一行とみなすかを指定
paramSearchText.setOverlapAsLine(overlapRate);
```

文字列検索をした部分に墨消しをして削除する

ドキュメント全体に検索・墨消しをかける場合

```
//入力 PDF の取得は「PDF を開く」を参照
//検索するテキストやマスク処理のパラメータ
PtlParamSearchTextAndSetMask paramSearchTextSetMask = new PtlParamSearchTextAndSetMask();
PtlColorDeviceRGB color = new PtlColorDeviceRGB(red, green, blue);

paramSearchTextSetMask.setColor(color);              //マスクの色
paramSearchTextSetMask.setOpacity(opacity);          //マスクの不透明度
paramSearchTextSetMask.appendText(textToSearch1);    //検索する文字列その 1
paramSearchTextSetMask.appendText(textToSearch2);    //検索する文字列その 2

//ドキュメント全体を検索し、ヒットした部分をマスクする
doc.searchTextAndDoProcess(paramSearchTextSetMask);
```

関数 searchTextAndDoProcess()はページオブジェクトに対して指定することも可能です。

```
//ページを検索し、ヒットした部分をマスクする
page.searchTextAndDoProcess(paramSearchTextSetMask);
```

墨消し機能の詳細は「墨消し機能（データの削除）」の項を参照してください。

文字列検索をしてハイライト注釈を付与する

「注釈」の項の「文字列検索してハイライト注釈を付ける」を参照してください。

参考：

・文字列検索の基礎

『PDF CookBook（第 3 巻）1.2.1 キーワードの指定による検索』

https://www.antenna.co.jp/ptl/cookbook/vol3/i01-0004.html

・文字列検索の細かいオプション

『PDF CookBook（第 3 巻）1.2.2 検索オプションの指定：検索対象文字列のオプション』

https://www.antenna.co.jp/ptl/cookbook/vol3/i01-0005.html

『PDF CookBook（第 3 巻）1.2.3 検索オプションの指定：取得順序』

https://www.antenna.co.jp/ptl/cookbook/vol3/i01-0006.html

『PDF CookBook（第 3 巻）1.2.4 検索オプションの指定：同一行とみなす文字の重なり』

https://www.antenna.co.jp/ptl/cookbook/vol3/i01-0006.html

・文字列検索をした部分に墨消しをして削除する

『PDF CookBook（第 3 巻）3.3.1 テキスト検索とマスク処理の組み合わせ』

https://www.antenna.co.jp/ptl/cookbook/vol3/i01-0039.html

・文字列検索をしてハイライト注釈をかける

『PDF CookBook（第 4 巻）1.1.9 テキスト検索してハイライト注釈』

https://www.antenna.co.jp/ptl/cookbook/vol4/i02-0012.html

10-10　PDF 属性の確認・設定

概説：

- プロパティの確認・設定
  - 文書情報の確認
  - 文書情報の設定
  - カスタムプロパティの設定
  - カスタムプロパティの削除
  - 文書のフォント情報の確認
  - 開き方の確認
  - PDF のバージョンを確認
  - リニアライズの有無の確認
- PDF/X かどうかの確認
- PDF/A かどうかの確認
- 電子署名の有無の確認

プロパティの確認・設定

プロパティの基本

```
//入力 PDF の取得は「PDF を開く」を参照
PtlDocProperty docProperty = doc.getDocProperty();  //PDF の文書プロパティ用クラスを取得
```

文書情報の確認

```
//入力 PDF の取得は「PDF を開く」を参照
//プロパティ用クラスの取得は「プロパティの基本」を参照
PtlDocInfo docinf = docProperty.getDocInfo();       //PDF の文書情報用クラスを取得
docinf.getTitle();          //「タイトル」の値を取得
docinf.getAuthor();         //「作成者」の値を取得
docinf.getSubject();        //「サブタイトル」の値
docinf.getKeywords();       //「キーワード」の値を取得
docinf.getCreator();        //「アプリケーション」(元ドキュメントの作成ソフト名) の値を取得
docinf.getProducer();       //「PDF 変換」(変換ソフト名) の値を取得
PtlDate dateCreate = docinf.getCreationDate();  //作成日時を取得
PtlDate dateMod = docinf.getModDate();          //更新日時を取得

PtlCustomProperties customProperties = docProperty.getCustomProperties();   //カスタムプロパティ情報取得
customProperties.isEmpty(); //カスタムプロパティのコンテナに値が存在するか否か
//インデックス番号を指定してカスタムプロパティを取得
PtlCustomProperty customProperty = customProperties.get(customPropertyNum);
customProperty.getName();   //カスタムプロパティの「名前」の値を取得
customProperty.getValue();  //カスタムプロパティの「値」の値を取得
```

文書情報の設定

```
//入力 PDF の取得は「PDF を開く」を参照
//プロパティ用クラスの取得は「プロパティの基本」を参照
PtlDocInfo docinf = docProperty.getDocInfo();       //PDF の文書情報用クラスを取得

docinf.setTitle(title);                     //タイトルを設定
```

```
docinf.setAuthor(author);                    //作成者を設定
docinf.setCreationDate(dateCreateNew);       //作成日時を設定
docinf.setModDate(dateModNew);               //更新日時を設定

// 出力 PDF の保存は「PDF の保存」を参照
```

カスタムプロパティの設定

```
//入力 PDF の取得は「PDF を開く」を参照
//プロパティ用クラスの取得は「プロパティの基本」を参照
//カスタムプロパティ用コンテナクラスを取得
PtlCustomProperties customProperties = docProperty.getCustomProperties();
//新規カスタムプロパティ用クラス
PtlCustomProperty newCustomProp = new PtlCustomProperty(name, value);
customProperties.append(newCustomProp);              //コンテナクラスにカスタムプロパティを追加

// 出力 PDF の保存は「PDF の保存」を参照
```

カスタムプロパティの削除

```
//入力 PDF の取得は「PDF を開く」を参照
//プロパティ用クラスの取得は「プロパティの基本」を参照
//カスタムプロパティ用コンテナクラスを取得
PtlCustomProperties customProperties = docProperty.getCustomProperties();
customProperties.remove(numToRemoveCustomProp); //指定したカスタムプロパティを削除

// 出力 PDF の保存は「PDF の保存」を参照
```

カスタムプロパティは一括削除をすることも可能

```
customProperties.removeAll();   //コンテナクラス内の全てのカスタムプロパティを削除
```

文書のフォント情報の確認

```
//入力 PDF の取得は「PDF を開く」を参照
//プロパティ用クラスの取得は「プロパティの基本」を参照
PtlFontInfos fontoInfos= docProperty.getFontInfos();    //フォント情報コンテナを取得
PtlFontInfo fontInfo = fontInfos.get(fontInfoNum);      //フォント情報を取得

fontInfo.getEncodingName();     //エンコーディング名を取得
fontInfo.getFontName();         //フォント名を取得
fontInfo.isEmbedded();          //埋め込みを許可されているか否かを取得
```

開き方の確認

```
//入力 PDF の取得は「PDF を開く」を参照
//プロパティ用クラスの取得は「プロパティの基本」を参照
PtlOpenMode openmode = docProperty.getOpenMode();   //オープンモードの取得
openmode.getOpenAction();       //オープンアクションを取得
openmode.getOpenDest();         //オープン時の宛先を取得
```

開き方の設定は「開き方」の項を参照してください

PDF のバージョンを確認

```
//入力 PDF の取得は「PDF を開く」を参照
//プロパティ用クラスの取得は「プロパティの基本」を参照
PtlDocProperty.PDF_VERSION version = docProperty.getVersion();      //PDF のバージョンを取得
```

リニアライズの有無の確認

```
//入力 PDF の取得は「PDF を開く」を参照
//プロパティ用クラスの取得は「プロパティの基本」を参照
docProperty.isLinearized();     //リニアライズされているか否かを取得
```

PDF/X かどうかの確認

```
//入力 PDF の取得は「PDF を開く」を参照
doc.isPDFX();       //PDF/X か否かの取得
```

PDF/A かどうかの確認

「PDF/A」の項の「PDF/A の種類の取得」を参照してください。

電子署名の有無の確認

```
//入力 PDF の取得は「PDF を開く」を参照
doc.isSignatured(); //電子署名をされているか否かの取得
```

参考：

・プロパティの確認・設定

・文書情報の確認

『PDF CookBook（第 4 巻）7.1.1 文書情報の取得』

https://www.antenna.co.jp/ptl/cookbook/vol4/i02-0049.html

・文書情報の設定

『PDF CookBook（第 4 巻）7.1.2 文書情報の設定』

https://www.antenna.co.jp/ptl/cookbook/vol4/i02-0050.html

・カスタムプロパティの設定

『PDF CookBook（第 4 巻）7.2.1 独自の項目名を持つカスタムプロパティの設定』

https://www.antenna.co.jp/ptl/cookbook/vol4/i02-0052.html

・カスタムプロパティの削除

『PDF CookBook（第 4 巻）7.2.2 カスタムプロパティの削除』

https://www.antenna.co.jp/ptl/cookbook/vol4/i02-0053.html

・PDF のバージョンを確認

『PDF CookBook（第 5 巻）1.1.1 PDF 2.0 ファイルの読み込み』

https://www.antenna.co.jp/ptl/cookbook/vol5/i03-0004.html

・リニアライズの有無の確認

『PDF CookBook（第 4 巻）7.3.5 Web 表示用に最適化されているか否かを判定する』

https://www.antenna.co.jp/ptl/cookbook/vol4/i02-0059.html

・PDF/X かどうかの確認

『PDF CookBook（第 4 巻）7.3.2 PDF/X かどうかを取得』

https://www.antenna.co.jp/ptl/cookbook/vol4/i02-0057.html

・PDF/A かどうかの確認

『PDF CookBook（第 4 巻）7.3.2 PDF/A かどうかを取得』

https://www.antenna.co.jp/ptl/cookbook/vol4/i02-0056.html

・電子署名の有無の確認

『PDF CookBook（第 4 巻）7.3.1 署名付き PDF かどうかを取得』

https://www.antenna.co.jp/ptl/cookbook/vol4/i02-0055.html

10-11　ページ属性情報の取得

概説：

- 境界値の取得
- ページサイズの取得

境界値の取得

```
//入力 PDF の取得は「PDF を開く」を参照
//ページコンテナ・ページ取得は「ページ情報を操作する」を参照
PtlRect artBox = page.getArtBox();     // 読み込んだページの ArtBox 取得
PtlRect bleedBox = page.getBleedBox(); // 読み込んだページの BleedBox 取得
PtlRect cropBox = page.getCropBox();   // 読み込んだページの CropBox 取得
PtlRect mediaBox = page.getMediaBox(); // 読み込んだページの MediaBox 取得
PtlRect trimBox = page.getTrimBox();   // 読み込んだページの TrimBox 取得

// 出力 PDF の保存は「PDF の保存」を参照
```

ページサイズの取得

```
//入力 PDF の取得は「PDF を開く」を参照
//ページコンテナ・ページ取得は「ページ情報を操作する」を参照
PtlSize size = page.getSize();      // 読み込んだページのサイズ取得

// 出力 PDF の保存は「PDF の保存」を参照
```

参考：

・境界値の取得

『PDF CookBook（第 1 巻）1.2.1 ページ境界値の取得』

https://www.antenna.co.jp/ptl/cookbook/vol1/i01-0002.html

10-12　開き方

概要：

- 開き方の基本
- PDF を開いたときのアクションを指定する
- PDF のページモードの指定
- PDF のページレイアウトの指定
- PDF の UI の指定
- PDF のウィンドウオプションの指定
- タイトルバーにおける文書タイトルの表示・非表示
- 開き方設定の削除

開き方の基本：

```
//入力 PDF の取得は「PDF を開く」を参照
PtlDocProperty docProperty = doc.getDocProperty(); //プロパティの取得
PtlOpenMode openmode = docProperty.getOpenMode();  //開き方の取得
```

PDF を開いたときのアクションを指定する

```
//入力 PDF の取得は「PDF を開く」を参照
//開き方用クラスの取得は「開き方の基本」を参照
PtlActionGoTo act = new PtlActionGoTo();   // 設定する GoTo アクション
PtlDestFit dst = new PtlDestFit();         //設定する宛先

dst.setPageNumber(pageNum);                //宛先としてページを設定
act.setDest(dst);                          //GoTo アクションに宛先を設定
openmode.setOpenAction(act);               //オープンアクションとして GoTo アクションを設定

// 出力 PDF の保存は「PDF の保存」を参照
```

この例では GoTo アクションの宛先に PtlDestFit を指定している。setDest()は PtlDest の子クラスであればどの宛先クラスも指定できる。

PDF のページモードの指定

```
//入力 PDF の取得は「PDF を開く」を参照
//開き方用クラスの取得は「開き方の基本」を参照

//ページモードの「アウトラインパネルとページ」を表すクラス
PtlOpenMode.PAGE_MODE pageModeType = PtlOpenMode.PAGE_MODE.PAGE_MODE_USE_OUTLINES;
openmode.setPageMode(pageModeType); //ページモードを「アウトラインパネルとページ」に指定

// 出力 PDF の保存は「PDF の保存」を参照
```

PDF のページレイアウトの指定

```
//入力 PDF の取得は「PDF を開く」を参照
//開き方用クラスの取得は「開き方の基本」を参照

//ページレイアウトの「単一ページ」を表すクラス
PtlOpenMode.PAGE_LAYOUT pageLayoutType = PtlOpenMode.PAGE_LAYOUT.PAGE_LAYOUT_SINGLE_PAGE;
openmode.setPageLayout(pageLayoutType); //ページレイアウトを「単一ページ」に指定

// 出力 PDF の保存は「PDF の保存」を参照
```

PDF の UI の指定

```
//入力 PDF の取得は「PDF を開く」を参照
//開き方用クラスの取得は「開き方の基本」を参照
int UIOption = PtlOpenMode.UI_OPTION_NONE;      //フラグ用変数

//「メニューバーを隠す」「ツールバーを隠す」「ウィンドウ UI を隠す」を同時指定する
UIOption = UIOption | PtlOpenMode.UI_OPTION_HIDE_MENUBAR;
UIOption = UIOption | PtlOpenMode.UI_OPTION_HIDE_TOOLBAR;
UIOption = UIOption | PtlOpenMode.UI_OPTION_HIDE_WINDOWUI;

openmode.setUIOption(UIOption);     //指定した UI のフラグを UI オプションに設定する

// 出力 PDF の保存は「PDF の保存」を参照
```

PDF のウィンドウオプションの指定

```
//入力 PDF の取得は「PDF を開く」を参照
//開き方用クラスの取得は「開き方の基本」を参照
int windowOption = PtlOpenMode.WINDOWS_OPTION_NONE; //フラグ用変数
```

```
//「ページにウィンドウサイズを合わせる」「ウィンドウを画面中央に配置」「フルスクリーンモードで開く」を同時指定する
windowOption = windowOption | PtlOpenMode.WINDOWS_OPTION_FIT_WINDOW;
windowOption = windowOption | PtlOpenMode.WINDOWS_OPTION_CENTER_WINDOW;
windowOption = windowOption | PtlOpenMode.WINDOWS_OPTION_FULL_SCREEN;

openmode.setWindowOption(windowOption); //指定した UI のフラグをウィンドウオプションに設定する

// 出力 PDF の保存は「PDF の保存」を参照
```

タイトルバーにおける文書タイトルの表示・非表示

```
//入力 PDF の取得は「PDF を開く」を参照
//開き方用クラスの取得は「開き方の基本」を参照
openmode.setDisplayDocTitle(flagDisplayDocTitle);    //タイトルバーに文書タイトルを表示するかを指定
// 出力 PDF の保存は「PDF の保存」を参照
```

開き方設定の削除

```
//入力 PDF の取得は「PDF を開く」を参照
//開き方用クラスの取得は「開き方の基本」を参照
openmode.removeOpenAction();                    //オープンアクションの削除
// 出力 PDF の保存は「PDF の保存」を参照
```

参考：

・PDF を開いたときのアクションを指定する

『PDF CookBook（第 4 巻）5.1.1 PDF を開いた時の動作』

https://www.antenna.co.jp/ptl/cookbook/vol4/i02-00610.html

・PDF を開いたときのページモードの指定

・PDF を開いたときのページレイアウトの指定

『PDF CookBook（第 4 巻）5.1.2 ページモード・ページレイアウトの指定』

https://www.antenna.co.jp/ptl/cookbook/vol4/i02-0062.html

・PDF の UI の指定

『PDF CookBook（第 4 巻）5.1.3 UI（ユーザーインターフェイス）オプションの設定』

https://www.antenna.co.jp/ptl/cookbook/vol4/i02-0063.html

・PDF のウィンドウオプションの指定

『PDF CookBook（第 4 巻）5.1.4 ウィンドウオプションの設定』

https://www.antenna.co.jp/ptl/cookbook/vol4/i02-0064.html

・タイトルバーにおける文書タイトルの表示・非表示

『PDF CookBook（第 4 巻）5.1.5 タイトルバーへの文書タイトル表示の有無を選択する』

https://www.antenna.co.jp/ptl/cookbook/vol4/i02-0065.html

・開き方設定の削除

10-13　PDF の最適化

概要：

- PDF 内の画像の最適化
  - 画像の最適化の基礎
  - 画像種別ごとの最適化パラメータの指定
  - カラー画像とグレースケール画像に適用できる関数
  - カラー画像とグレースケール画像、モノクロ 2 値画像に適用できる関数
  - PtlParamOptimizeImage を用いたパラメータの指定
- 不要なデータの削除

PDF 最適化の基礎

```
//入力 PDF  doc の取得は「PDF を開く」を参照
PtlParamOptimize paramOptimize = new PtlParamOptimize();    //最適化用パラメータクラス
doc.optimize(paramOptimize);                                //最適化の実行

// 出力 PDF の保存は「PDF の保存」を参照
```

本項で扱う最適化用パラメータはすべて paramOptimize に設定する。

PDF 内の画像の最適化

画像の最適化の基礎

```
//入力 PDF doc の取得は「PDF を開く」を参照
//最適化用パラメータ paramOptimize の取得は「PDF 最適化の基礎」を参照

//画像の最適化用パラメータクラスの取得
PtlParamOptimizeImage paramOptimizeImage = paramOptimize.getParamOptimizeImage();

//ここで具体的な画像の最適化パラメータを設定する

//最適化実行は「PDF 最適化の基礎」を参照
// 出力 PDF の保存は「PDF の保存」を参照
```

画像の最適化はこの PtlParamOptimizeImage を取得することで行います。

画像種別ごとの最適化パラメータの指定

```
//入力 PDF doc の取得は「PDF を開く」を参照
//最適化用パラメータ paramOptimize の取得は「PDF 最適化の基礎」を参照
//画像最適化用パラメータ paramOptimizeImage の取得は「画像の最適化の基礎」を参照

//カラー画像用最適化パラメータクラスの取得
PtlParamOptimizeImageColor paramOptimizeImageColor = paramOptimizeImage.getParamOptimizeImageColor()

//ここでパラメータクラスの内容を指定する

//最適化実行は「PDF 最適化の基礎」を参照
// 出力 PDF の保存は「PDF の保存」を参照
上記はカラー画像の最適化パラメータを指定して最適化を実行した例です。
```

グレースケール画像の最適化の場合は以下のパラメータを用います。

```
//グレースケール画像最適化パラメータクラスの取得
PtlParamOptimizeImageGrayScale paramOptimizeImageGrayScale = paramOptimizeImage.getParamOptimizeImage
```

```
GrayScale();
```

モノクロ 2 値画像の最適化の場合は以下のパラメータを用います

```
//モノクロ 2 値画像最適化パラメータクラスの取得
PtlParamOptimizeImageMono paramOptimizeImageMono = paramOptimizeImage.getParamOptimizeImageMono();
```

これらパラメータに適用できる内容は以下のものです。

カラー画像とグレースケール画像に適用できる関数

```
//圧縮タイプを JPEG に指定
paramOptimizeImageColor.setCompress(PtlParamOptimizeImageColor.COMPRESS_TYPE.COMPRESS_JPEG);
//圧縮品質を「中」に指定
paramOptimizeImageColor.setQuality(PtlParamOptimizeImageColor.QUALITY_TYPE.QUALITY_MIDDLE);
```

本例はカラー画像を最適化している関数ですが、グレースケール画像にも適用できます。

カラー画像とグレースケール画像、モノクロ 2 値画像に適用できる関数

```
//ダウンサンプリング方法を「バイキュービック法」で指定
paramOptimizeImageColor.setDownSampling(PtlParamOptimizeImageDownSampling.DOWNSAMPLING_TYPE.DOW
NSAMPLING_BICUBIC);
//ダウンサンプリング率の下限値を指定
paramOptimizeImageColor.setMinDownSamplingRate(minDownSamplingRate);
//ダウンサンプリングを行う基準の PPI を指定。これより大きいとダウンサンプリングする
paramOptimizeImageColor.setSourcePPI(sourcePPI);
//ダウンサンプリング後の PPI を指定
paramOptimizeImageColor.setTargetPPI(targetPPI);
```

本例はカラー画像を最適化している関数ですが、グレースケール画像・モノクロ 2 値画像にも適用できます。

PtlParamOptimizeImage を用いたパラメータの指定

```
//入力 PDF doc の取得は「PDF を開く」を参照
//最適化用パラメータ paramOptimize の取得は「PDF 最適化の基礎」を参照
//画像最適化用パラメータ paramOptimizeImage の取得は「画像の最適化の基礎」を参照

int filterFlag = PtlParamOptimizeImage.FILTER_NONE;     //フラグ用変数
//「ASCII85Decode」「DCTDecode」「FlateDecode」を同時指定するフラグ
filterFlag = filterFlag | PtlParamOptimizeImage.FILTER_ASCII85Decode;
filterFlag = filterFlag | PtlParamOptimizeImage.FILTER_DCTDecode;
filterFlag = filterFlag | PtlParamOptimizeImage.FILTER_FlateDecode;

paramOptimizeImage.setValidFilter(filterFlag);  //フィルターの設定
//最適化処理は「PDF 最適化の基礎」を参照
// 出力 PDF の保存は「PDF の保存」を参照
```

本例はフィルターを指定した画像の最適化を例示しているが、他にも以下のような関数がある

```
//ダウンサンプリングをする画像の最小サイズを指定
paramOptimizeImage.setMinSampleSize(minSampleSize);
```

minSampleSize より縦・横の画素数が共に大きい場合にダウンサンプリングを行う

不要なデータの削除

```
//入力 PDF doc の取得は「PDF を開く」を参照
//最適化用パラメータ paramOptimize の取得は「PDF 最適化の基礎」を参照
paramOptimize.setRemoveOutlines(true); //しおりを削除するよう指定

//最適化実行は「PDF 最適化の基礎」を参照
// 出力 PDF の保存は「PDF の保存」を参照
```

本例はしおりを削除するよう指定しているが、以下の関数でそれぞれの要素を削除指定できる

```
paramOptimize.setRemoveAnnots(true);        //注釈・フォームを削除するよう指定
paramOptimize.setRemoveArticles(true);      //アーティクルを削除するよう指定
paramOptimize.setRemoveThumbnails(true);    //サムネールを削除するよう指定
paramOptimize.setRemoveJavaScripts(true);   //JavaScript を削除するよう指定
```

フォントの統合

```
//入力 PDF doc の取得は「PDF を開く」を参照
//最適化用パラメータ paramOptimize の取得は「PDF 最適化の基礎」を参照
paramOptimize.setMergeFonts(true);
//最適化処理は「PDF 最適化の基礎」を参照
// 出力 PDF の保存は「PDF の保存」を参照
```

参考：

PDF 内の画像の最適化

・画像種別ごとの最適化パラメータの指定

『PDF CookBook（第 3 巻）2.2.1 カラー画像最適化オプションの取得・指定』

https://www.antenna.co.jp/ptl/cookbook/vol3/i01-0018.html

『PDF CookBook（第 3 巻）2.2.2 グレースケール画像最適化オプションの取得・指定』

https://www.antenna.co.jp/ptl/cookbook/vol3/i01-0019.html

『PDF CookBook（第 3 巻）2.2.3 モノクロ画像最適化オプションの取得・指定』

https://www.antenna.co.jp/ptl/cookbook/vol3/i01-0020.html

・カラー画像とグレースケール画像に適用できる関数

『PDF CookBook（第 3 巻）2.3.1 JPEG 圧縮設定』

https://www.antenna.co.jp/ptl/cookbook/vol3/i01-0028.html

・カラー画像とグレースケール画像、モノクロ 2 値画像に適用できる関数

『PDF CookBook（第 3 巻）2.2.4 ダウンサンプリング方法の指定』

https://www.antenna.co.jp/ptl/cookbook/vol3/i01-0021.html

『PDF CookBook（第 3 巻）2.2.7 ダウンサンプリング率の下限値の指定』

https://www.antenna.co.jp/ptl/cookbook/vol3/i01-0024.html

『PDF CookBook（第 3 巻）2.2.8 ダウンサンプリング対象の画像を PPI で絞り込む』

https://www.antenna.co.jp/ptl/cookbook/vol3/i01-0025.html

『PDF CookBook（第 3 巻）2.2.9 ダウンサンプリング後の PPI を指定する』

https://www.antenna.co.jp/ptl/cookbook/vol3/i01-0026.html

『PDF CookBook（第 3 巻）2.2.9 ダウンサンプリング後の PPI を指定する』

https://www.antenna.co.jp/ptl/cookbook/vol3/i01-0026.html

・PtlParamOptimizeImage を用いたパラメータの指定

『PDF CookBook（第 3 巻）2.2.5 最適化を行う画像の対象とするフィルターの指定』

https://www.antenna.co.jp/ptl/cookbook/vol3/i01-0022.html

『PDF CookBook（第 3 巻）2.2.6 ダウンサンプリングする画素数の指定』

https://www.antenna.co.jp/ptl/cookbook/vol3/i01-0023.html

不要なデータの削除

『PDF CookBook（第 3 巻）5.2.1 オープンアクションの削除』

https://www.antenna.co.jp/ptl/cookbook/vol3/i01-0047.html

『PDF CookBook（第 3 巻）5.2.2 しおりの削除』

https://www.antenna.co.jp/ptl/cookbook/vol3/i01-0048.html

『PDF CookBook（第 3 巻）5.2.3 注釈・フォームの削除』

https://www.antenna.co.jp/ptl/cookbook/vol3/i01-0049.html

『PDF CookBook（第 3 巻）5.2.4 アーティクルの削除』

https://www.antenna.co.jp/ptl/cookbook/vol3/i01-0050.html

『PDF CookBook（第 3 巻）5.2.5 サムネールの削除』

https://www.antenna.co.jp/ptl/cookbook/vol3/i01-0051.html

『PDF CookBook（第 5 巻）10.2.1 PDF に設定された JavaScript の一括削除』

https://www.antenna.co.jp/ptl/cookbook/vol5/i03-0074.html

10-14　レイヤー

概要：

- レイヤーとして PDF 文書ページを挿入する
- レイヤーのパラメータを指定する

レイヤーとして PDF 文書ページを挿入する

```
//入力 PDF の取得は「PDF を開く」を参照
//ページコンテナ・ページ・ページコンテント取得は「ページコンテントを操作する」を参照
PtlPDFDocument docToInsert = new PtlPDFDocument();          //挿入する PDF 用クラス
PtlParamInput inputFileToInsert = new PtlParamInput(insertPDFPath); //挿入する PDF のパス
docToInsert.load(inputFileToInsert);                        //挿入する PDF を取得
PtlPages pagesToInsert = docToInsert.getPages();            //挿入する PDF のページコンテナを取得
PtlPage pageInsert = pagesToInsert.get(insertPageNum);      //挿入するページを取得

PtlParamDrawLayer paramLayer = new PtlParamDrawLayer();     //レイヤーの描画用パラメータ
paramLayer.setPage(pageInsert);                             //レイヤーに挿入するページを指定
PtlRect rectLayer = new PtlRect(left, bottom, right, top);  //挿入するレイヤーの矩形
//コンテントにレイヤーを挿入する
content.drawLayer(rectLayer, PtlContent.ALIGN.ALIGN_CENTER, paramLayer);

// 出力 PDF の保存は「PDF の保存」を参照
```

レイヤーのパラメータを指定する

```
//入力 PDF の取得は「PDF を開く」を参照
//ページコンテナ・ページ・ページコンテント取得は「ページコンテントを操作する」を参照
//レイヤーPDF の指定、レイヤー描画パラメータ paramLayer の取得は「レイヤーとして PDF 文書ページを挿入する」
を参照
paramLayer.setName(layerName);          //レイヤーの名前を指定

//レイヤーの設定は「レイヤーとして PDF 文書ページを挿入する」を参照
// 出力 PDF の保存は「PDF の保存」を参照
```

本例はレイヤーの名前を指定しているが、以下の関数でそれぞれの要素を指定できる

```
paramLayer.setOpacity(opacity);         //レイヤーの不透明度を指定
```

```
paramLayer.setRotate(angle);            //レイヤーの回転を指定（90 度単位）
paramLayer.setAngle(angle);             //レイヤーの回転を指定（任意の数値）
//レイヤーの Z オーダーを指定（本例は前面で指定）
paramLayer.setZorder(PtlparamLayerDrawLayer.ZORDER.ZORDER_FRONT);
//レイヤーの表示・非表示を指定（本例は非表示で指定）
paramLayer.setShow(PtlparamLayerDrawLayer.SHOW.SHOW_OFF);
```

参考：

・レイヤーとして PDF 文書ページを挿入する

『PDF CookBook（第 3 巻）6.1.1 レイヤーに使用する PDF 文書ページを設定』

https://www.antenna.co.jp/ptl/cookbook/vol3/i01-0056.html

・レイヤーのパラメータを指定する

『PDF CookBook（第 3 巻）6.1.2 レイヤーの名前の指定』

https://www.antenna.co.jp/ptl/cookbook/vol3/i01-0057.html

『PDF CookBook（第 3 巻）6.1.3 レイヤーの不透明度の指定』

https://www.antenna.co.jp/ptl/cookbook/vol3/i01-0058.html

『PDF CookBook（第 3 巻）6.1.4 回転角度の設定』

https://www.antenna.co.jp/ptl/cookbook/vol3/i01-0059.html

『PDF CookBook（第 5 巻）10.5.1 角度設定（レイヤー描画)』

https://www.antenna.co.jp/ptl/cookbook/vol3/i01-0059.html

『PDF CookBook（第 3 巻）6.1.5 レイヤーの Z オーダーの指定』

https://www.antenna.co.jp/ptl/cookbook/vol3/i01-0059.html

『PDF CookBook（第 3 巻）6.1.6 レイヤーの表示/非表示の指定』

https://www.antenna.co.jp/ptl/cookbook/vol3/i01-0061.html

10-15　墨消し機能（データの削除）

概要：

- 墨消し機能の基礎
- 墨消し箇所の見た目を設定する
- 削除対象を指定する
- テキスト削除時のオプションを指定する

墨消し機能の基礎

```
//入力 PDF の取得は「PDF を開く」を参照
//ページコンテナ・ページ取得は「ページ情報を操作する」を参照
PtlParamSetMask paramSetMask = new PtlParamSetMask();       //マスク処理のパラメータ用クラス
PtlRect rectMask = new PtlRect(left, bottom, right, top);   //マスクの矩形

paramSetMask.appendRect(rectMask);      //マスクの矩形を指定
page.setMask(paramSetMask);             //マスクをページに設定
// 出力 PDF の保存は「PDF の保存」を参照
```

墨消し箇所の見た目を設定する

```
//入力 PDF の取得は「PDF を開く」を参照
//ページコンテナ・ページ取得は「ページ情報を操作する」を参照
```

```
//マスク用パラメータ paramSetMask の取得は「マスク機能の基礎」を参照
PtlColorDeviceRGB colorRGB = new PtlColorDeviceRGB(red, green, blue);   //マスクの色
paramSetMask.setMaskColor(colorRGB);            //マスクの色を設定
paramSetMask.setOpacity(opacity);               //マスクの不透明度を設定

//マスク範囲の指定、マスクの適用は「マスク機能の基礎」を参照
// 出力 PDF の保存は「PDF の保存」を参照
```

墨消し箇所の削除対象

```
//入力 PDF の取得は「PDF を開く」を参照
//ページコンテナ・ページ取得は「ページ情報を操作する」を参照
//マスク用パラメータ paramSetMask の取得は「マスク機能の基礎」を参照
paramSetMask.setRemoveElement(PtlParamSetMask.REMOVE_TEXT);

//マスク範囲の指定、マスクの適用は「マスク機能の基礎」を参照
// 出力 PDF の保存は「PDF の保存」を参照
```

本例はテキストを指定した場合

フラグによる論理和で複数種類を同時に指定することも可能

```
int removeElement = PtlParamSetMask.REMOVE_NONE;    //フラグ用変数
//フラグにテキストと画像を同時指定する
removeElement = removeElement | PtlParamSetMask.REMOVE_TEXT;
removeElement = removeElement | PtlParamSetMask.REMOVE_IMAGE;

paramSetMask.setRemoveElement(removeElement);       //削除対象をフラグで指定する
```

テキストに関しては、矩形とどの程度重なっていたら削除するのかを overlapRatio で指定できる

```
paramSetMask.setRemoveElement(PtlParamSetMask.REMOVE_TEXT); //削除対象をテキストに指定
paramSetMask.setTextOverlapRatio(overlapRatio);             //削除する重なり具合を指定
```

テキスト検索をして墨消し処理をする

「文字列検索」の項の「文字列検索をした部分に墨消しをして削除する」を参照してください。

参考：

マスク機能

・マスク機能の基礎

・マスクの見た目を設定する

『PDF CookBook（第 3 巻）3.1.1 マスクの色』

https://www.antenna.co.jp/ptl/cookbook/vol3/i01-0031.html

・マスクの見た目を設定する

『PDF CookBook（第 3 巻）3.1.1 マスクの色』

https://www.antenna.co.jp/ptl/cookbook/vol3/i01-0031.html

『PDF CookBook（第 3 巻）3.1.2 マスクの不透明度』

https://www.antenna.co.jp/ptl/cookbook/vol3/i01-0032.html

・マスクの削除対象

『PDF CookBook（第 3 巻）3.2.1 テキスト：矩形内の文字を削除』

https://www.antenna.co.jp/ptl/cookbook/vol3/i01-0034.html

『PDF CookBook（第 3 巻）3.2.3 画像：矩形内の画像データを部分削除』

https://www.antenna.co.jp/ptl/cookbook/vol3/i01-0036.html

『PDF CookBook（第 3 巻）3.2.4 図形：矩形内にパスデータ全体が含まれる場合に削除』

https://www.antenna.co.jp/ptl/cookbook/vol3/i01-0037.html

『PDF CookBook（第 3 巻）3.2.2 テキスト：削除時オプションの指定』

https://www.antenna.co.jp/ptl/cookbook/vol3/i01-0035.html

・テキスト検索をしてマスク処理をする

『PDF CookBook（第 3 巻）3.3.1 テキスト検索とマスク処理の組み合わせ』

https://www.antenna.co.jp/ptl/cookbook/vol3/i01-0039.html

10-16　暗号化

概要：

- パスワードによる暗号化
  - ユーザーパスワードによる暗号化
  - オーナーパスワードによる暗号化
  - オーナーパスワードの権限設定
  - パスワード付き PDF を開く
  - パスワードによる暗号化の解除

- 電子証明書を用いた暗号化
  - 証明書による暗号化（証明書セキュリティ）
  - 証明書セキュリティの権限設定
  - 証明書セキュリティ PDF を開く
  - 証明書セキュリティの削除

- セキュリティ情報の読み取り
  - 暗号化付き PDF の情報取得
  - パスワード付き PDF の情報取得
  - 証明書セキュリティ PDF の情報取得

パスワードによる暗号化

```
//入力 PDF の取得は「PDF を開く」を参照
//暗号化パラメータ設定用クラス
encPtlEncryptStandard256AES enc256 = new PtlEncryptStandard256AES();

//ユーザーパスワード設定やオーナーパスワードの設定は「セキュリティハンドラの設定項目」を参照

doc.setEncrypt(enc256);     //ドキュメントに対して暗号化をする

// 出力 PDF の保存は「PDF の保存」を参照
```

本例では 256bit AES 暗号化用パラメータクラスを用いています。

その他の暗号化形式に合わせたパラメータ設定用クラスを用いることも可能です。

セキュリティハンドラの設定項目

```
enc256.setUserPassword(outUserPass);      //ユーザーパスワードの設定
enc256.setOwnerPassword(outOwnerpass);   //オーナーパスワードの設定
//暗号化する対象の設定
enc256.setEncryptComponent(PtlEncrypt.ENCRYPT_COMPONENT.ENCRYPT_EXCEPT_METADATA);
```

オーナーパスワードの権限設定

```
//入力 PDF の取得は「PDF を開く」を参照
```

```
//標準セキュリティハンドラ enc256 の取得は「オーナーパスワードによる暗号化」を参照
//オーナーパスワードの設定は「セキュリティハンドラの設定項目」を参照

//ユーザーアクセス許可フラグ用クラス
PtlEncryptPermissionType2 perms2 = new PtlEncryptPermissionType2();
//プリント権限を「不可」に設定
perms2.setPrint(PtlEncryptPermissionType2.PERMISSION_PRINT.PERM_PRINT_NOT_ALLOWED);
//編集権限を「不可」に設定
perms2.setModify(PtlEncryptPermissionType2.PERMISSION_MODIFY.PERM_MODIFY_NOT_ALLOWED);
perms2.setCopy(false);              //コピーを禁止
perms2.setAccessibility(true);      //アクセシビリティ関連のアクセスを許可

enc256.setPermission(perms2);       //セキュリティハンドラにユーザーアクセス許可フラグを設定
//暗号化実行は「パスワードによる暗号化」を参照
//出力 PDF の保存は「PDF の保存」を参照
```

パスワード付き PDF を開く

```
//入力ファイルパスの設定・入力 PDF 用クラス doc の作成は「PDF を開く」を参照
doc.setPassword(password);      //PDF を開くためのパスワードを設定
doc.load(inputFile);            //PDF を開く
```

パスワードはユーザーパスワード・オーナーパスワードで異なる場合がある。

開いた後の用途によってはユーザーパスワードだけでは情報取得や加工ができない点に注意。

パスワードによる暗号化の解除

```
//パスワード付き PDF の取得は「パスワード付き PDF を開く」を参照
doc.isEncrypted();          //暗号化されているか否かの確認
doc.hasOwnerAuthority();    //オーナー権限を持っているか否かの確認

doc.removeEncrypt();        //暗号化を解除
// 出力 PDF の保存は「PDF の保存」を参照
```

電子証明書を用いた暗号化

証明書による暗号化（証明書セキュリティ）

```
//入力 PDF の取得は「PDF を開く」を参照
PtlParamInput x509Pubkey = new PtlParamInput(X509FilePath); //証明書のパスを指定
PtlRecipient recipient = new PtlRecipient();                //証明書の受信者設定用クラス
recipient.setX509(x509Pubkey);                              //パスの先の証明書を受信者に設定

//証明書セキュリティ用セキュリティハンドラクラス
PtlEncryptPubKey256AES encPubkey256AES = new PtlEncryptPubKey256AES();
//セキュリティハンドラが持つ受信者コンテナを取得
PtlRecipients recipients = encPubkey256AES.getRecipients();
recipients.append(recipient);                               //受信者コンテナに受信者を追加

//暗号化する対象を指定
encPubkey256AES.setEncryptComponent(PtlEncrypt.ENCRYPT_COMPONENT.ENCRYPT_ALL);
doc.setEncrypt(encPubkey256AES);                            //暗号化の実行

// 出力 PDF の保存は「PDF の保存」を参照
```

上記は X.509 ファイルによる暗号化の例

証明書セキュリティの権限設定

```
//入力 PDF の取得は「PDF を開く」を参照
//受信者の取得は「証明書による暗号化」を参照
//証明書セキュリティにおけるユーザーアクセス許可フラグ設定用クラス
PtlEncryptPermissionPubKey permission = new PtlEncryptPermissionPubKey();
permission.setFullPermission(false);    //暗号化の解除を含む全権限を持たせないよう設定
permission.setCopy(false);              //コピー禁止
permission.setAccessibility(true);      //アクセシビリティ関連のアクセスを許可
//プリント権限を「不可」に設定
permission.setPrint(PtlEncryptPermissionType2.PERMISSION_PRINT.PERM_PRINT_NOT_ALLOWED);
//編集権限を「不可」に設定
permission.setModify(PtlEncryptPermissionType2.PERMISSION_MODIFY.PERM_MODIFY_NOT_ALLOWED);
```

```
recipient.setPermission(permission);    //受信者に対してユーザーアクセス許可フラグを設定

//受信者の設定・証明書による暗号化は「証明書による暗号化」を参照
// 出力 PDF の保存は「PDF の保存」を参照
```

証明書セキュリティ PDF を開く

```
//入力ファイルパスの設定・入力 PDF 用クラス doc の作成は「PDF を開く」を参照
PtlParamInput pkcs12File = new PtlParamInput(pkcs12FilePath);   //PKCS#12 ファイルのパスを指定
doc.setPKCS12(pkcs12File);                          //PDF ドキュメントに PKCS#12 ファイルを指定
doc.load(inputFile);                                //PDF を開く
```

証明書セキュリティの削除

```
//証明書セキュリティ付き PDF doc の取得は「証明書セキュリティ PDF を開く」を参照
//ドキュメントの暗号化情報を取得（証明書セキュリティ用のクラスにリキャスト）
PtlEncryptPubKey encryptPubkey = (PtlEncryptPubKey)doc.getEncrypt();
//ユーザーアクセス許可フラグを取得（証明書セキュリティ用のクラスにリキャスト）
PtlEncryptPermissionPubKey permsPubkey = (PtlEncryptPermissionPubKey)encryptPubkey.getPermission();
permsPubkey.hasFullPermission();    //全ての権限を持ったユーザーアクセス許可フラグであるかを確認

doc.removeEncrypt();                //暗号化を解除

// 出力 PDF の保存は「PDF の保存」を参照
```

セキュリティ情報の読み取り

暗号化付き PDF の共通情報取得

```
//入力 PDF の取得は「PDF を開く」を参照
PtlEncrypt encrypt = doc.getEncrypt();  //ドキュメントの暗号化情報を取得

encrypt.getEncryptComponent();          //暗号化の対象が何かの情報を取得
encrypt.getKeyLength();                 //キー長を取得
```

```
encrypt.getFilterType();                //セキュリティハンドラのタイプを取得
```

パスワード付き PDF の情報取得

```
//パスワード付き PDF の取得は「パスワード付き PDF を開く」を参照
//PDF の暗号化情報 encrypt の取得・共通の情報取得は「暗号化付き PDF の情報取得」を参照

//セキュリティハンドラのタイプに合わせてリキャスト
encPtlEncryptStandard256AES enc256 = (encPtlEncryptStandard256AES)encrypt;
//ユーザーアクセス許可フラグを取得（標準セキュリティ用のクラスにリキャスト）
PtlEncryptPermissionType2 perms2 = (PtlEncryptPermissionType2)enc256.getPermission();
perms2.getPrint();              //印刷許可に関する権限情報を取得
perms2.getModify();              //編集許可に関する権限情報を取得
perms2.getCopy();               //コピー可否についての情報を取得
perms2.getAccessibility();      //アクセシビリティ関連のアクセス可否を取得
```

上記は 256 ビット AES 暗号化で暗号化された PDF だった場合の例

証明書セキュリティ PDF の情報取得

```
//パスワード付き PDF の取得は「パスワード付き PDF を開く」を参照
//PDF の暗号化情報 encrypt の取得・共通の情報取得は「暗号化付き PDF の情報取得」を参照

//セキュリティハンドラのタイプに合わせてリキャスト
PtlEncryptPubKey encryptPubkey = (PtlEncryptPubKey)encrypt;
//ユーザーアクセス許可フラグを取得（証明書セキュリティ用のクラスにリキャスト）
PtlEncryptPermissionPubKey permsPubkey = (PtlEncryptPermissionPubKey)encryptPubkey.getPermission();
permsPubkey.hasFullPermission();    //全ての権限を持っているか
permsPubkey.getPrint();             //印刷許可に関する権限情報を取得
permsPubkey.getModify();             //編集許可に関する権限情報を取得
permsPubkey.getCopy();              //コピー可否についての情報を取得
permsPubkey.getAccessibility();     //アクセシビリティ関連のアクセス可否を取得
```

上記は 256 ビット AES 暗号化で暗号化されていた PDF の例

参考：

パスワードを用いた暗号化

・ユーザーパスワードによる暗号化

『PDF CookBook（第 2 巻）1.1.2 ユーザーパスワードによるセキュリティの設定』

https://www.antenna.co.jp/ptl/cookbook/vol2/i02-0004.html

『PDF CookBook（第 2 巻）1.1.3 暗号化の対象』

https://www.antenna.co.jp/ptl/cookbook/vol2/i02-0010.html

・オーナーパスワードによる暗号化

『PDF CookBook（第 2 巻）1.2.1 オーナーパスワードの設定』

https://www.antenna.co.jp/ptl/cookbook/vol2/i02-0005.html

『PDF CookBook（第 2 巻）1.1.3 暗号化の対象』

https://www.antenna.co.jp/ptl/cookbook/vol2/i02-0010.html

・オーナーパスワードの権限設定

『PDF CookBook（第 2 巻）1.2.2 印刷不可セキュリティの設定』

https://www.antenna.co.jp/ptl/cookbook/vol2/i02-0006.html

『PDF CookBook（第 2 巻）1.2.3 変更不可セキュリティの設定』

https://www.antenna.co.jp/ptl/cookbook/vol2/i02-0007.html

『PDF CookBook（第 2 巻）1.2.4 内容のコピーの制限』

https://www.antenna.co.jp/ptl/cookbook/vol2/i02-0008.html

『PDF CookBook（第 2 巻）1.2.5 アクセシビリティのための内容の抽出の制限』

https://www.antenna.co.jp/ptl/cookbook/vol2/i02-0009.html

・パスワード付き PDF を開く

『PDF CookBook（第 2 巻）1.1.1 セキュリティ情報の取得』

https://www.antenna.co.jp/ptl/cookbook/vol2/i02-0003.html

・パスワードによる暗号化の解除

『PDF CookBook（第 2 巻）1.1.4 セキュリティの解除』

https://www.antenna.co.jp/ptl/cookbook/vol2/i02-0011.html

電子証明書を用いた暗号化

・証明書による暗号化（証明書セキュリティ）

『PDF CookBook（第 5 巻）6.1.1 証明書によるセキュリティ設定(X.509 形式)』

https://www.antenna.co.jp/ptl/cookbook/vol5/i03-0039.html

『PDF CookBook（第 5 巻）6.1.2 証明書によるセキュリティ設定(PKCS#12 形式)』

https://www.antenna.co.jp/ptl/cookbook/vol5/i03-0040.html

・証明書セキュリティの権限設定

『PDF CookBook（第 5 巻）6.1.3 すべての権限を許可する(setFullPermission)』

https://www.antenna.co.jp/ptl/cookbook/vol5/i03-0041.html

・証明書セキュリティ PDF を開く

『PDF CookBook（第 5 巻）6.2.1 証明書セキュリティ付き PDF から権限情報を取得する』

https://www.antenna.co.jp/ptl/cookbook/vol5/i03-0043.html

・証明書セキュリティの削除

『PDF CookBook（第 5 巻）6.3.1 証明書セキュリティの解除』

https://www.antenna.co.jp/ptl/cookbook/vol5/i03-0045.html

セキュリティ情報の読み取り

・暗号化付き PDF の情報取得

『PDF CookBook（第 2 巻）1.1.1 セキュリティ情報の取得』

https://www.antenna.co.jp/ptl/cookbook/vol2/i02-0003.html

・パスワード付き PDF の情報取得

『PDF CookBook（第 2 巻）1.1.1 セキュリティ情報の取得』

https://www.antenna.co.jp/ptl/cookbook/vol2/i02-0003.html

・証明書セキュリティ PDF の情報取得

『PDF CookBook（第 5 巻）6.2.1 証明書セキュリティ付き PDF から権限情報を取得する』

https://www.antenna.co.jp/ptl/cookbook/vol5/i03-0043.html

10-17　PDF/A

概要：

- PDF/A 変換
- PDF/A 準拠チェック
- PDF/A のエラー読み取り
- PDF/A の種類の取得

PDF/A 変換

```
PtIParamInput inputFile = new PtIParamInput(inputPDFPath);      //入力 PDF のパスを設定
PtIParamOutput outputFile = new PtIParamOutput(outputPDFAPath); //出力 PDF のパスを設定
PtIPDFFixUp fixUp = new PtIPDFFixUp();                          //PDF/A 変換用クラス

fixUp.fixUpPDFA(PtIPDFFixUp.PDFA_TYPE.PDFA_2B, inputFile);      //入力 PDF を PDF/A-2b に変換する
fixUp.save(outputFile);                                         //PDF/A 変換後のファイルを出力する
```

本例では PDF/A-2b ファイルへの変換操作を行っている

PDF/A 変換時はオプションとしてリニアライズをするかどうかを指定できる

```
fixUp.setSaveOption(PtIPDFFixUp.SAVE_OPTION.SAVE_LINEARIZE);    //出力ファイルをリニアライズする
```

PDF/A 準拠チェック

```
PtlParamInput inputFile = new PtlParamInput(inputPDFPath);      //入力 PDF のパスを設定
PtlPDFFixUp fixUp = new PtlPDFFixUp();                          //PDF/A 用クラス

fixUp.validatePDFA(PtlPDFFixUp.PDFA_TYPE.PDFA_2B, inputFile);   //入力ファイルの PDF/A の準拠チェック
```

本例では PDF/A-2b に対する準拠チェックを行っている

PDF/A のエラー読み取り

PDF/A 変換、及び PDF/A 準拠チェック時に読み取ったエラーは以下で確認できる

```
//入力ファイル名の設定・PDF/A 変換は「PDF/A 変換」を参照

//エラーコンテナに変換時のエラーを取得
PtlPDFFixUpErrors fixUpErrors = fixUp.getErrors();
//インデックス番号を指定してエラーを取得
PtlPDFFixUpError thisFixUpError = fixUpErrors.get(errorNum);
thisFixUpError.getErrorMessageJP();                         //日本語でエラーメッセージを取得
```

PDF/A の種類の取得

```
//入力 PDF の取得は「PDF を開く」を参照
doc.isPDFA();                                               //PDF/A であるか否かを取得
PtlPDFDocument.PDFA_TYPE pdfAType = doc.getPDFAType();      //PDF/A タイプを取得
```

参考：

・PDF/A 変換

『PDF CookBook（第 5 巻）2.2.1 PDF/A-1b、PDF/A-2b への変換』

https://www.antenna.co.jp/ptl/cookbook/vol5/i03-0007.html

『PDF CookBook（第 5 巻）2.2.2 保存オプションの指定(PDF/A への変換)』

https://www.antenna.co.jp/ptl/cookbook/vol5/i03-0008.html

・PDF/A 準拠チェック

『PDF CookBook（第 5 巻）2.3.1 PDF/A-1b,PDF/A-2b の準拠確認』

https://www.antenna.co.jp/ptl/cookbook/vol5/i03-0010.html

・PDF/A のエラー読み取り

『PDF CookBook（第 5 巻）2.2.1 PDF/A-1b、PDF/A-2b への変換』

https://www.antenna.co.jp/ptl/cookbook/vol5/i03-0007.html

『PDF CookBook（第 5 巻）2.3.1 PDF/A-1b,PDF/A-2b の準拠確認』

https://www.antenna.co.jp/ptl/cookbook/vol5/i03-0010.html

・PDF/A の種類の取得

『PDF CookBook（第 5 巻）2.3.2 PDF/A の種類の取得』

https://www.antenna.co.jp/ptl/cookbook/vol5/i03-0011.html

10-18　透かし機能

概要：

- 透かしの挿入
  - テキスト透かしの挿入
  - 画像透かしの挿入
  - PDF 透かしの挿入
  - 色透かしの挿入
- 透かしの設定
  - 透かしの共通設定
  - 透かしを Acrobat©で加工可能な形式にする
- 透かしの削除

テキスト透かしの挿入

```
//入力 PDF の取得は「PDF を開く」を参照

//テキスト透かしのパラメータ設定用クラス
PtlParamWaterMarkText waterMarkText = new PtlParamWaterMarkText();
//テキスト透かしの色設定用クラス
PtlColorDeviceRGB colorText = new PtlColorDeviceRGB(red, green, blue);
PtlParamFont font = new PtlParamFont(); //フォント指定用クラス

//フォント用クラス font の内容設定は「テキストのフォントを指定」を参照

waterMarkText.setString(contentOfWaterMark);    //テキスト透かしの文字列を指定
waterMarkText.setTextColor(colorText);          //テキスト透かしの色を指定
waterMarkText.setFont(font);                    //テキスト透かしのフォントを指定

doc.appendWaterMark(waterMarkText);             //テキスト透かしをドキュメントに挿入
// 出力 PDF の保存は「PDF の保存」を参照
```

画像透かしの挿入

```
//入力 PDF の取得は「PDF を開く」を参照

//画像透かしのパラメータ設定用クラス
PtlParamWaterMarkImage waterMarkImage = new PtlParamWaterMarkImage();
PtlParamInput inputImage = new PtlParamInput(pathImage);    //入力画像のパス
waterMarkImage.setImageStream(inputImage);                  //入力画像を画像透かしに設定

doc.appendWaterMark(waterMarkImage);                        //画像透かしをドキュメントに挿入
// 出力 PDF の保存は「PDF の保存」を参照
```

PDF 透かしの挿入

```
//入力 PDF の取得は「PDF を開く」を参照
PtlPDFDocument docWatermark = new PtlPDFDocument();     //透かし用 PDF 用クラス
PtlParamInput inputPdf = new PtlParamInput(pathPdf);    //透かし用 PDF のパス
docWatermark.load(inputPdf);                            //透かし用 PDF の読み込み
PtlPages pages = docWatermark.getPages();               //透かし用 PDF のページコンテナを取得
PtlPage pageOfWatermark = pages.get(pageNum);           //透かし用 PDF のページを取得

//PDF 透かしのパラメータ用クラス
PtlParamWaterMarkPDF waterMarkPdf = new PtlParamWaterMarkPDF();
waterMarkPdf.setPage(pageOfWatermark);                  //PDF 透かしにページを設定

doc.appendWaterMark(waterMarkPdf);                      //PDF 透かしをドキュメントに挿入
// 出力 PDF の保存は「PDF の保存」を参照
```

色透かしの挿入

```
//入力 PDF の取得は「PDF を開く」を参照
//色透かしのパラメータ設定用クラス
PtlParamWaterMarkColor waterMarkColor = new PtlParamWaterMarkColor();
```

```
PtlColorDeviceRGB color = new PtlColorDeviceRGB(red, green, blue); //色透かしの色
waterMarkColor.setColor(color);             //色透かしの色を設定

doc.appendWaterMark(waterMarkColor);        //色透かしをドキュメントに挿入
// 出力 PDF の保存は「PDF の保存」を参照
```

透かしの設定

透かしの共通設定

```
//入力 PDF の取得は「PDF を開く」を参照
//テキスト透かし waterMarkText の取得は「テキスト透かしの挿入」を参照

waterMarkText.setAlign(PtlParamWaterMark.ALIGN.ALIGN_CENTER);   //透かしの位置を中央に指定
waterMarkText.setAngle(15.0f);                                  //角度を 15 度に指定
//挿入ページをカスタムで指定
waterMarkText.setPageRange(PtlParamWaterMark.PAGE_RANGE.PAGE_RANGE_CUSTOM);
waterMarkText.setCustomPageRange(pageRange);                    //挿入ページを指定
waterMarkText.setDipslayWaterMark(false)                        //ビューワ上での透かし表示をオフ
waterMarkText.setPrintWaterMark(true);                          //プリント時の透かし表示をオン
waterMarkText.setName("waterMarkText");                         //透かしの名前を設定
waterMarkText.setOpacity(0.7f);                                 //透かしの不透明度を設定
waterMarkText.setTiling(false);                                 //タイリング設定をオフ
waterMarkText.setZorder(PtlParamWaterMark.ZORDER.ZORDER_FRONT); //透かしを文書前面に挿入

//テキスト透かしの挿入は「テキスト透かしの挿入」を参照
// 出力 PDF の保存は「PDF の保存」を参照
```

上記設定は全透かしに共通の設定です。

本例ではテキスト透かしに値を設定しています。

透かしを Acrobat©で加工可能な形式にする

```
//入力 PDF の取得は「PDF を開く」を参照
//テキスト透かし waterMarkText の取得は「テキスト透かしの挿入」を参照
```

```
waterMarkText.setAcrobatCompatible(setAcrobatCompatible);    //透かしを Acrobat 互換に設定

//テキスト透かしの挿入は「テキスト透かしの挿入」を参照
// 出力 PDF の保存は「PDF の保存」を参照
```

本例ではテキスト透かしを Acrobat©で加工可能に設定しています。

Acrobat©対応透かしと併用できない設定が存在します。例えば、setTiling は false である必要があります。詳細はリファレンスをご確認ください。

透かしの削除

```
//入力 PDF の取得は「PDF を開く」を参照
doc.removeWaterMark(watermarkname);                    //指定した名前の透かしを削除

// 出力 PDF の保存は「PDF の保存」を参照
```

参考：

・テキスト透かしの挿入

『PDF CookBook（第 2 巻）2.2 テキスト透かし』

https://www.antenna.co.jp/ptl/cookbook/vol2/i03-0012.html

・画像透かしの挿入

『PDF CookBook（第 2 巻）2.3.1 画像透かしの挿入』

https://www.antenna.co.jp/ptl/cookbook/vol2/i03-0022.html

・PDF 透かしの挿入

『PDF CookBook（第 2 巻）2.3.2 PDF 透かしの挿入』

https://www.antenna.co.jp/ptl/cookbook/vol2/i03-0023.html

・色透かしの挿入

『PDF CookBook（第 2 巻）2.4.1 色透かしの挿入』

https://www.antenna.co.jp/ptl/cookbook/vol2/i03-0028.html

透かしの設定

・透かしの共通設定

『PDF CookBook（第 2 巻）2.1 透かし(共通)』

https://www.antenna.co.jp/ptl/cookbook/vol2/i03-0002.html

・透かしを Acrobat©で加工可能な形式にする

『PDF CookBook（第 5 巻）8.1.1 透かしを Acroabt で加工可能にする』

https://www.antenna.co.jp/ptl/cookbook/vol5/i03-0055.html

・透かしの削除

『PDF CookBook（第 2 巻）2.1.10 透かしの削除』

https://www.antenna.co.jp/ptl/cookbook/vol2/i03-0011.html

10-19　各要素の情報取得・判定

概要：

- テキストが持つ情報の取得
- 画像の持つ情報の取得
- パスが持つ情報の取得

各要素が共通で持つ情報の取得

```
//入力 PDF doc の取得は「PDF を開く」を参照
//ページコンテナ・ページ・ページコンテント取得は「ページコンテントを操作する」を参照
PtlEditElements elems = content.getEditElements(PtlContent.GET_ALL);
PtlEdt Element element = elems.get(elementNum);     //インデックス番号を指定してエレメントを取得
PtlRect bBox = element.getBBox();                   //エレメントが持つ BBox を取得
PtlQuadPoint quadPoint = element.getQuadPoint();    //エレメントが持つ QuadPoint を取得
element.getType();                                  //エレメントのタイプを取得
```

テキストが持つ情報の取得

```
//入力 PDF doc の取得は「PDF を開く」を参照
//ページコンテナ・ページ・ページコンテント取得は「ページコンテントを操作する」を参照
//コンテントからテキストを指定してエレメントコンテナに取得
PtlEditElements elems = content.getEditElements(PtlContent.GET_TEXT);
//PtlEditText にリキャストしてテキストエレメントを取得
PtlEditText textElement = (PtlEditText)elems.get(elementNum);
PtlEditTextItems textItems = textElement.getTextItems();    //テキストアイテムコンテナを取得
PtlEditTextItem textItem = textItems.get(itemNum);          //テキストアイテムを取得
PtlRect textItemBBox =textItem.getBBox();                   //テキストアイテムの BBox を取得
textItem.getText();                                 //テキストを取得
textItem.getPaintFlags();                           //ペイントフラグを取得
```

```
PtlColor fillColor = textItem.getFillColor();       //塗りつぶしカラーを取得
PtlColor strokeColor = textItem.getStrokeColor();   //ストロークカラーを取得
```

PtlEditTextItem からはフォント情報も読み取ることができる。

```
//入力 PDF doc の取得は「PDF を開く」を参照
//ページコンテナ・ページ・ページコンテント取得は「ページコンテントを操作する」を参照
//テキストアイテム textItem の取得は「テキストが持つ情報の取得」を参照
PtlFontInfo fontInfo = textItem.getFontInfo(); //フォント情報を取得
fontInfo.getFontName();                        //フォント名を取得
fontInfo.getFontType();                        //フォントタイプを取得
fontInfo.getEncodingType();                    //エンコーディングタイプを取得
fontInfo.getEncodingName();                    //エンコーディング名を取得
```

画像が持つ情報の取得

```
//コンテントから画像を指定してエレメントコンテナに取得
PtlEditElements elems = content.getEditElements(PtlContent.GET_IMAGE);
//PtlEditImage にリキャストして画像エレメントを取得
PtlEditImage elemImage = (PtlEditImage)elem;
elemImage.getHeight();      //画像の高さを取得
elemImage.getWidth();       //画像の幅を取得
elemImage.getPPI();         //画像の PPI を取得
```

パスが持つ情報の取得

```
//入力 PDF doc の取得は「PDF を開く」を参照
//ページコンテナ・ページ・ページコンテント取得は「ページコンテントを操作する」を参照

//コンテントからパスを指定してエレメントコンテナに取得
PtlEditElements elems = content.getEditElements(PtlContent.GET_PATH);
//PtlEditPath にリキャストしてパスエレメントを取得
PtlEditPath pathElement = (PtlEditPath)elems.get(elementNum);
```

```
pathElement.getPaintFlags();        //ペイントフラグを取得
pathElement.getFillColor();         //塗りつぶしカラーを取得
pathElement.getStrokeColor();       //ストロークカラーを取得

PtlEditPathItems pathItems = pathElement.getPathItems();    //パスアイテムコンテナを取得
PtlEditPathItem pathItem = pathItems.get(pathItemNum);      //パスアイテムを取得
pathItem.getType();                                         //パスアイテムの種別を取得
PtlEditPathLine pathLine = (PtlEditPathLine)pathItem;       //パスアイテムをリキャスト
PtlPoint start = pathLine.getStartPoint();                  //線の始点を取得
PtlPoint end = pathLine.getEndPoint();                      //線の終点を取得
```

参考：

・テキストが持つ情報の取得

『PDF CookBook（第 5 巻）4.1.1 4.1.2 PtlEditText の情報を取得』

https://www.antenna.co.jp/ptl/cookbook/vol5/i03-0027.html

・画像の持つ情報の取得

『PDF CookBook（第 3 巻）2.1.1 画像個数の取得』

https://www.antenna.co.jp/ptl/cookbook/vol3/i01-0013.html

『PDF CookBook（第 3 巻）2.1.4 画像の大きさ、解像度を取得』

https://www.antenna.co.jp/ptl/cookbook/vol3/i01-0016.html

・パスが持つ情報の取得

『PDF CookBook（第 5 巻）4.1.1 PtlEditPath の情報を取得』

https://www.antenna.co.jp/ptl/cookbook/vol5/i03-0026.html

10-20　フォントの統合・埋め込み

概要：

- フォントの埋め込み
- フォントの統合

フォントの埋め込み

```
//入力 PDF  doc の取得は「PDF を開く」を参照
doc.embedFonts(); // フォント埋め込みを設定する
// 出力 PDF の保存は「PDF の保存」を参照
```

フォントの統合

「PDF の最適化」の項の「フォントの統合」を参照してください。

10-21　C++における文字列クラスの作成

概要：

・PtlParamString クラス（C++ API）の定義

PtlParamString クラス（C++ API）の定義

```
// char* から構築する PtlParamString の定義
const char* str = "example";
PtlParamString p1(str);

// std::string から char* 経由で行う PtlParamString の定義
std::string stdstr = "hello";
PtlParamString p2(stdstr.c_str()); //

// CP_UChar* にキャストして構築する PtlParamString の定義
std::wstring stdwstr = L"アンテナハウス";
PtlParamString p3((CP_UChar*)stdwstr.c_str());
```

第11章 仕様変更について

11-1　V7.0 との違い

11-1-1　　V8.0 で削除された要素

V7.0 の PtlColor クラスにある「getCSType()」を、V8.0 で廃止しました。

該当する関数は下記の C++ API と Java API にあります。.NET API には存在しません。

[C++ API]

PtlColorSpace::COLOR_SPACE_TYPE PdfTk::PtlColor::getCSType()const

[Java API]

public PtlColorSpace.COLOR_SPACE_TYPE getCSType()　throws PtlException

11-1-2　　PDF 規格ファイルに対する処理

PDF/A、PDF/X、PDF/E の PDF 規格ファイルに対して処理を行ったとき、文書情報にある PDF 規格の情報を削除するようにしました。

V7.0 では、PDF/A の情報はそのまま残り、PDF/X の情報は削除していました。
V8.0 では、PDF 規格ファイルに対する処理オプションを指定するインターフェースを新たに設けました。

11-1-3　　文字列取得に関する戻り値の変更

V7.0 にて「PtlParamString」クラスを戻り値として受け取っていた各種関数に関して、
V8.0 では「const PtlParamString&」の形でポインタを戻り値として受け取るように変更しました。

例：「PtlEditTextItem:: getText ()」のリファレンス表記

V8.0	V7.0
const PtlParamString& getText ()	PtlParamString getText()

11-1-4　　綴りの修正

V8.0	V7.0	
PtlEditImage::setPassThrough	PtlEditImage::setPathThrough	PtlEditImage クラス
-extractImage -passThrough	-extractImage -pathThrough	コマンドライン

11-2　V6.0 / V5.0 / V4.0 との違い

11-2-1　　セキュリティ設定の仕様変更

[40bit RC4 の非対応]

V8.0 では、40bit RC4 のセキュリティ設定は非対応です。処理はエラーになります。

40bit RC4 のセキュリティ設定された PDF ファイルの読み込みやセキュリティの削除の処理には対対応しています。

[256bit AES 設定の変更点]

V8.0 では、256bit AES のセキュリティ設定は「R（リビジョン）6」で行います。

V6.0/ V5.0 / V4.0 では、「R5」で行っていました。

11-2-2　　文書情報の Metadata 作成

V8.0 では、入力ファイルに Metadata が存在しない場合、『PDF Tool API』が新たに Metadata を作成します。

V6.0 / V5.0 / V4.0 では、入力ファイルに Metadata が存在しない場合、『PDF Tool API』が新たに Metadata を作成することはありません。文書情報設定を行ったときは Document Information 辞書を更新します。

11-2-3　　文字列取得に関する戻り値の変更

過去バージョンにて「PtlParamString」クラスを戻り値として受け取っていた各種関数に関して、
V8.0 では「const PtlParamString&」の形でポインタを戻り値として受け取るように変更しました。

例：「PtlEditTextItem:: getText ()」のリファレンス表記

V8.0	V6.0 / V5.0 / V4.0
const PtlParamString& getText ()	PtlParamString getText()

11-2-4　　綴りの修正

V8.0	V5.0 / V4.0	
PtlEncryptPermissionType2::setAccessibility	PtlEncryptPermissionType2::setAccesibility	PtlEncryptPermissionType2 クラス
PtlEncryptPermissionType2::getAccessibility	PtlEncryptPermissionType2::getAccesibility	PtlEncryptPermissionType2 クラス
PERM_MODIFY_ASSEMBLEANDFORM	PERM_MODIFY_ASSENBLEANDFORM	enum PtlEncryptPermissionType1::PERMISSION_MODIFY のメンバー
PERM_MODIFY_ASSEMBLEDOC	PERM_MODIFY_ASSENBLEDOC	enum PtlEncryptPermissionType2::PERMISSION_MODIFY のメンバー
PtlParamWaterMark::setDisplayWaterMark	PtlParamWaterMark::setDipslayWaterMark	PtlParamWaterMark クラス

第12章 評価版の仕様について

- Windows 版ではインストール後 30 日間、Linux 版ではインストール後『PDF Tool API』を初めて実行した日から 30 日間、使用できます。期間を過ぎると使用できません。

評価版は以下の動作仕様があります。

- 透かし文字列：「Antenna House PDF Tool」を各ページに挿入
- 画像を抽出する操作： 1 個のみ抽出
- 文字を抽出する操作：1 ページのみ抽出
- 添付ファイルを抽出する操作： 1 個のみ抽出

第13章 バージョンアップについて

『PDF Tool API』のバージョンは、「Vn.m」形式で表記されます。「Vn.m」の n をメジャーバージョンといい、m をマイナーバージョンといいます。
たとえば、『PDF Tool API V8.0』のメジャーバージョンは 8、マイナーバージョンは 0 です。

「Vn.m」の n の増加をメジャーバージョンアップ、m の増加をマイナーバージョンアップといいます。

本章ではバージョンアップに関する手続きや仕様を解説します。

13-1　保守契約期限内のバージョンアップについて

保守契約期限内であれば、無償または購入価格との差額で『PDF Tool API』のバージョンアップが可能です。

- バージョンアップする場合は新しいバージョンのインストーラを入手・実行する必要があります。
  ただし、1 つのライセンスで運用可能なバージョン数や複数バージョンの同時インストールに関して注意点がございます。詳細は『13-2 バージョンアップとライセンスに関する制限事項』、『13-3 複数バージョンのインストールについて』を参照してください。
- バージョンアップする場合はインストーラの入手・実行の他、該当バージョンに合わせたライセンスファイルの入手と入れ替えも必要になります。
  - 保守契約期限内であれば、新しいライセンスファイルは保守契約窓口(hosyu@antenna.co.jp)に請求することで入手できます。
  - 具体的な入れ替え操作については、『5-4 ライセンスファイルの入れ替えについて』を参照してください。
- バージョンアップに際して、バージョン間の価格改定があった場合に差額が必要となる場合があります。
  具体的な差額についての詳細は弊社 web ページ『バージョンアップ』のページを参照してください。
  (www.antenna.co.jp/ptl/versionup.html)

13-2　バージョンアップとライセンスに関する制限事項

バージョンアップとライセンスに関する制限事項は以下の通りです。

- １つのライセンスで同時に２つのバージョンを運用することは許諾されていません。
  ライセンスの新規購入をせずにバージョンアップして新バージョンのみを運用する場合は旧バージョンのアンインストールが必要です。
- 旧バージョンと新バージョンの『PDF Tool API』を両方運用で使う場合は旧バージョンのライセンスをそのまま保有し、新バージョンのライセンスを新規購入して頂く必要があります。
- アップグレードを検討される際に２つのバージョンを同時に運用されたい場合は保守契約窓口(hosyu@antenna.co.jp)にご相談ください。

13-3　複数バージョンのインストールについて

複数バージョンの『PDF Tool API』を同一マシンへインストールすることへの対応は Windows 版と Linux 版で異なります。

> 注意：
> １つのライセンスで２つのバージョンを運用することは許諾されていません。そのため、同時稼働させるバージョンの数だけライセンスを購入して頂く必要があります。詳細は前述の『13-2 バージョンアップとライセンスに関する制限事項』を参照してください。

[Windows 版の場合]

Windows 版の場合、異なるバージョンの『PDF Tool API』を 1 台の PC 上にインストールすることができます。

[Linux 版の場合]

Linux 版の場合は、旧バージョンをアンインストールしてから新しいバージョンをインストールすることをお勧めします。なお、Linux 版でも旧バージョンとは別フォルダに新規インストールすること自体は可能です。

第14章 改訂版について

『PDF Tool API』はメジャーバージョン、マイナーバージョンが同一のまま更新が加えられる「改訂版」がリリースされることがあります。
より具体的には、改訂版とはバージョン番号（「Vn.m」の n と m）が同一でリリース日のみが新しくなっている製品のことです。メンテナンスリリース（MR）と呼ぶこともあります。改訂版の場合、バージョン番号の後に MR 番号が表記されます。

「メジャーバージョンアップ」「マイナーバージョンアップ」に関する詳細は『第 13 章 バージョンアップについて』を参照してください。

本章では改訂版のアップデート方法の詳細について説明します。

14-1　保守契約期限内の改訂版へのアップデートについて

- 改訂版へアップグレードする場合は改訂版のインストーラを入手・実行する必要があります。
- バージョンアップとは異なり、保守契約期限内にリリースされた改訂版であればアップデート後も現在のライセンスファイルをご利用いただけます。詳細は『5-6 保守契約期限と改訂版アップデートにおける『PDF Tool API』の動作について』を参照してください。
- 上書きインストールの可否が OS により異なるため、改訂版のインストール手順も OS ごとに異なります。詳細は後述の『14-2 改訂版のインストール方法について』を参照してください。

14-2　改訂版のインストール方法について

『PDF Tool API』の改訂版は Windows では上書きインストールが可能ですが、Linux 版では必要な操作が異なります。

[Windows 版]

改訂版（MR）にアップデートする場合は上書きインストールが可能です。
そのため、次の手順でアップデートできます。順序は 1 と 2 のどちらが先でも構いません。

1. 旧評価版をインストールしたフォルダに改訂版（最新版）を上書きインストールする。
2. 必要性な場合はインストール先のライセンスファイルを正規版のライセンスファイルに入れ替える[*1]。

[*1]：

ライセンスファイルの入れ替え作業に関する詳細は『5-4 ライセンスファイルの入れ替えについて』を参照してください。

[Linux 版]

Linux 版インストーラは、上書きインストールはできません。
つまり、改訂前の『PDF Tool API』を一旦アンインストールしてから改訂版を改めてインストールする必要があります。

Linux 版のインストールに関する詳細は『4-2-1 インストール方法』を参照してください。

Linux 版におけるライセンス用フォルダの扱いについて：

- アンインストール実行時、インストールフォルダ内のファイルはすべて削除されます。
  それに伴い、インストールフォルダ内のライセンスファイル用フォルダも削除されます。
- インストールフォルダ内にライセンスファイルを配置していた場合は、改訂版のインストール後に正規版ライセンスファイルの再配置が必要になります。

第15章 エラー処理について

15-1　エラー発生時の挙動

実行中に『PDF Tool API』ライブラリ上でエラーが発生した場合は、エラー内容を示す PtlException クラスが返されます。

- 各エラーコードの詳細は『15-2 エラーコード一覧』を参照してください。

エラーを受け取るためのコード例

```
// PDF Tool API を使った何らかの処理
catch (PtlException pex)
{
    Console.WriteLine(pex.getErrorCode() + " : " + pex.getErrorMessageJP());
    pex.Dispose();
}
```

15-2　エラーコード一覧

エラーコード	エラーメッセージ
0	正常終了
10	Cannot find license file. ライセンスファイルが見つかりません。
11	License file is expired. 評価版ライセンスの有効期限が切れています。
12	License file is invalid. ライセンスファイルが無効です。
13	License file is for other platform. ライセンスファイルが他プラットフォーム用です。
14	License file is for other product. ライセンスファイルが他製品用です。
100	Invalid PDF file.

	PDF が異常です。
101	Cannot read PDF. PDF の読み込みができません。
102	Cannot write PDF. PDF の書き出しができません。
103	Cannot write too large PDF. 大きすぎる PDF の書き出しができません。
110	Invalid user password. ユーザーパスワード不正です。
111	Invalid owner password. オーナーパスワード不正です。
112	Invalid password. パスワード不正です。
113	Has not authority. 処理権限がありません。
114	Is not encrypted. 暗号化されていません。
115	Unsupported security handler. 未対応のセキュリティハンドラです。
116	Unsupported security algorithm. 未対応のセキュリティアルゴリズムです。
117	Is signatured. 電子署名されています。
118	Has XFA(XML Form). XFA(XML Form) を持っています。
120	Can not certificate. 認証できません。
200	Invalid parameter value. パラメータに問題があります。
201	Invalid page number. ページの指定が間違っています。
202	Has no text. テキストの設定がありません。
203	Has no font. フォントの設定がありません。
204	Has no valid data. 有効なデータの設定がありません。
205	Cannot use this function.

	この関数は使えません。
210 [*1]	Need password. パスワードが必要です。
211 [*2]	Need user password. ユーザーパスワードが必要です。
212 [*3]	Need owner password. オーナーパスワードが必要です。
213	Invalid encrypt key length. 暗号化キー長が間違っています。
214	Invalid encrypt permission. 権限が間違っています。
215	Invalid encrypt component. 暗号化する文書コンポーネントに誤りがあります。
216	Invalid encrypt method. 暗号化メソッドが間違っています。
217	Need PKCS12. PKCS12 が必要です。
220	Can not read attached file. 添付ファイルの読み込みが出来ません。
221	Can not write attached file. 添付ファイルの書き出しが出来ません。
222	No attached file. 添付ファイルがありません。
223	Attached file has no name. 添付ファイルに名前がありません。
230	Can not read image file. 画像ファイルの読み込みが出来ません。
231	Can not write image file. 画像ファイルの書き出しが出来ません。
232	Unsupported image. 未対応の画像です。
233	Unsupported image for stencil mask. ステンシルマスクとしてサポートしていない画像です。
234	Image is not single. ステンシルマスクがモノクロ画像でありません。
235	Unsupported image for color key mask. カラーキーマスクとしてサポートしていない画像です。
236	Unsupported image for explicit mask.

	明示マスクとしてサポートしていない画像です。
237	Image is not single. 明示マスクがモノクロ画像でありません。
238	Image is not gray scale. ソフトマスクとしてサポートしていない画像です。
240	Image processing error. イメージ処理で問題が発生しました。
241	Can not read ICC Profile. ICC プロファイルの読み込みが出来ません。
245	Font processing error. フォント処理で問題が発生しました。
246 ※V8.0 追加	Cannot find glyph Glyph が見つからない
250	Can not insert page. ページの挿入が出来ません。
251	Can not delete page. ページの削除が出来ません。
252	Has no pages. ページが存在しません。
260 *4	Free docproperty error. DocProperty がドキュメントからフリーです。
261 *4	Free openmode error. OpenMode がドキュメントからフリーです。
262 *4	Free embeddedfiles error. EmbeddedFiles がドキュメントからフリーです。
263 *4	Free pages error. Pages がドキュメントからフリーです。
264 *4	Free page error. Page がドキュメントからフリーです。
265 5* ※V8.0 追加	Free fields error. Fields がドキュメントからフリーです。
270	Can not set to root outline. ルートアウトラインには設定出来ません。
271 *6	Can not set to free outline. フリーアウトラインには設定出来ません。
280	Invalid FDF file. FDF が異常です。

281	Cannot read FDF. FDF の読み込みができません。
282	Cannot write FDF. FDF の書き出しができません。
290	Cannot read PKCS12. PKCS12 の読み込みができません。
291	Cannot read X509. X509 の読み込みができません。
300 [7*] ※V8.0 追加	Cannot read XMP. XMP の読み込みができません。
301 [7*] ※V8.0 追加	Invalid XMP file. XMP が異常です。
302 [8*] ※V8.0 追加	Cannot write XMP. XMP の書き出しができません。
310 ※V8.0 追加	Cannot edit locked layer. ロックされているレイヤーは編集できません。
500 [*9]	Linearize processing error. 線形化処理で問題が発生しました。
700	null value. null 値です。
800	No Object. オブジェクトが存在しません。
900	Not enough memory. メモリが不足しています。
901	Internal error. 内部エラーです。
902	Other error. その他のエラーです。
999	Sorry, not implemented. 未実装です。

※備考

*1[210] PDF の暗号化設定処理においてパスワードの設定が行われていない場合に発生します。

*2[211] 添付ファイルのみの暗号化設定処理において添付ファイルを開くためのパスワードが設定されていない場合に発生します。

*3[212] PDF のセキュリティ権限フラグ設定処理において権限パスワードの設定が行われていない場合に発生します。

*4[260/261/262/263/264] 取得されたオブジェクトが PDF とは紐づいていない場合に発生します。

*5[265] PtlAcroForm から取得していないフォームフィールドのコンテナを扱おうとした際に発生します。

*6[271] しおりが作成できないことを示します。

*7[300/301] メタデータの設定に使う XMP ファイルに何らかのエラーがある場合に発生します。

*8[302] 取得したメタデータを書き出せない場合に発生します。

*9[500] 線形化処理＝Web 表示用に最適化する処理で問題が発生したことを意味します。

第16章 商標・著作権情報

16-1　商標

Microsoft、Windows、OpenType は米国 Microsoft Corporation の米国およびその他の国における登録商標または商標です。

Linux は、Linus Torvalds 氏の日本およびその他の国における登録商標または商標です。

Adobe Acrobat、および Adobe Acrobat Reader は、Adobe Inc.の米国ならびに他の国における登録商標または商標です。

TrueType は米国その他の国で登録された米国 Apple Inc.の商標です。

その他記載されている全ての会社名および製品名は、個々の所有者の登録商標または商標です。

16-2　第三者ライブラリー著作権情報

jpeg

Copyright (C) 1994-1998, Thomas G. Lane.

This file is part of the Independent JPEG Group's software.

(http://www.ijg.org/).

libtiff

This product includes software developed by Sam Leffler and Silicon Graphics, Inc (http://www.libtiff.org/).

Copyright (c) 1988-1996 Sam Leffler

Copyright (c) 1991-1996 Silicon Graphics, Inc.

zlib

This product includes software developed by Jean-loup Gailly and Mark Adler

(https://www.zlib.net/).

Copyright (C) 1995-2024 Jean-loup Gailly and Mark Adler

libpng

This code is released under the libpng license.

This product includes software developed by Glenn Randers-Pehrson and many other contributers (http://www.libpng.org/pub/png/libpng.html).

Copyright (c) 2004, 2006-2011 Glenn Randers-Pehrson

Copyright (c) 2000-2002 Glenn Randers-Pehrson

Copyright (c) 1998, 1999 Glenn Randers-Pehrson

Copyright (c) 1996, 1997 Andreas Dilger

Copyright (c) 1995, 1996 Guy Eric Schalnat, Group 42, Inc.

j2k(jasper)

JasPer License Version 2.0

Copyright (c) 2001-2016 Michael David Adams

Copyright (c) 1999-2000 Image Power, Inc.

Copyright (c) 1999-2000 The University of British Columbia

All rights reserved.

LittleCMS

Little Color Management System

Copyright (c) 1998-2020 Marti Maria Saguer

Permission is hereby granted, free of charge, to any person obtaining

a copy of this software and associated documentation files (the "Software"),

to deal in the Software without restriction, including without limitation

the rights to use, copy, modify, merge, publish, distribute, sublicense,

and/or sell copies of the Software, and to permit persons to whom the Software

is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in

all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,

EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO

THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND

NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE

LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION

OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION

WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

ICU

Unicode and the Unicode Logo are registered trademarks of Unicode, Inc. in the United States and other countries. (https://icu.unicode.org/)

NOTICE TO USER: Carefully read the following legal agreement. BY DOWNLOADING, INSTALLING, COPYING OR OTHERWISE USING DATA FILES, AND/OR SOFTWARE, YOU UNEQUIVOCALLY ACCEPT, AND AGREE TO BE BOUND BY, ALL OF THE TERMS AND CONDITIONS OF THIS AGREEMENT. IF YOU DO NOT AGREE, DO NOT DOWNLOAD, INSTALL, COPY, DISTRIBUTE OR USE THE DATA FILES OR SOFTWARE.

Permission is hereby granted, free of charge, to any person obtaining a copy of data files and any associated documentation (the "Data Files") or software and any associated documentation (the "Software") to deal in the Data Files or Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, and/or sell copies of the Data Files or Software, and to permit persons to whom the Data Files or Software are furnished to do so, provided that either (a) this copyright and permission notice appear with all copies of the Data Files or Software, or (b) this copyright and permission notice appear in associated Documentation.

THE DATA FILES AND SOFTWARE ARE PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OF THIRD PARTY RIGHTS.

IN NO EVENT SHALL THE COPYRIGHT HOLDER OR HOLDERS INCLUDED IN THIS NOTICE BE LIABLE FOR ANY CLAIM, OR ANY SPECIAL INDIRECT OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THE DATA FILES OR SOFTWARE.

Except as contained in this notice, the name of a copyright holder shall not be used in advertising or otherwise to promote the sale, use or other dealings in these Data Files or Software without prior written authorization of the copyright holder.

FreeType

This product includes softwares developed by:

Portions of this software are copyright 2024 The FreeType Project (www.freetype.org).

All rights reserved.

brotli

Copyright (c) 2009, 2010, 2013-2016 by the Brotli Authors.

(https://brotli.org/)

OpenSSL

OpenSSL is licensed under the Apache License 2.0, which means that you are free to get and use it for commercial and non-commercial purposes as long as you fulfill its conditions.

Copyright (c) 1998-2025 The OpenSSL Project

Copyright (c) 1995-1998 Eric A. Young, Tim J. Hudson

All rights reserved.

索引

改訂履歴

日付	内容
2025/07/31	初版発行
2025/09/30	MR2 に改訂 誤字・脱字の修正
2025/11/07	MR2a に改訂 対応 Java バージョンの表記を変更

奥付

Antenna House PDF Tool API V8.0 コマンドライン説明書 2025.11.7