.NET 開発者向け帳票作成ツール

Reports.NET

プログラマーズマニュアル

第 18 版

2024 年 5 月

Pao@Office

本書は、有限会社パオ・アット・オフィスが開発したソフトウエア「Reports.net」についての説明を行うものです。

利用者は本書のいかなる部分も、発行者の許可なく、複製を行ってはいけません。

有限会社パオ・アット・オフィスは、本書の内容に起因する一切の結果に関して、いかなる責任も負いません。

有限会社パオ・アット・オフィスは、本書の内容、または Reports.net の仕様を予告なく改訂、あるいは、内容変更する権利を有します。また、それらの行為を行った場合においても、利用者への通知の義務を負いません。

有限会社パオ・アット・オフィスは、Reports.net の仕様に起因する結果にたいして、いかなる責任も負いません。

マニュアル中での画像は、説明のため見やすく編集している箇所があります。利用者の皆様の画面とは一致しない場合がございますので、あらかじめご了承ください。

有限会社パオ・アット・オフィス
郵便番号 275-0026
千葉県習志野市谷津 3-29-2-401

http://www.pao.ac/

目次

はじめに

.NET 開発環境下で開発を行っているプログラマの皆様、こんにちは、お疲れ様です。
Reports.net の クラスとしてのインターフェースは非常にシンプルです。
クラスやメソッドの数も少ししかありません。

(メソッド)

IReports インターフェース ・・・ 印刷又はプレビューを行うための共通インターフェース

- LoadDefFile メソッド ・・・ デザインファイル(デザイン)を読み込む
- ChangeDefFile メソッド ・・・ デザインファイル(デザイン)変更
- PageStart メソッド ・・・ ページの開始を宣言する
- Write メソッド ・・・ 印刷データを書き込む
- PageEnd メソッド ・・・ ページの終了を宣言する
- Output メソッド ・・・ 印刷／プレビューを指示する
- GetPrintDocument メソッド ・・・ 独自プレビュー用 PrintDocument 取得
- SavePDF メソッド ・・・ PDF 形式の印刷データを書き出す
- SaveXPS メソッド ・・・ XPS 形式の印刷データを書き出す
- SaveXMLFile メソッド ・・・ 印刷データファイルを書き出す
- LoadXMLFile メソッド ・・・ 印刷データファイルを読み込む
- SaveData メソッド ・・・ 圧縮した印刷バイナリデータを返す
- LoadData メソッド ・・・ 圧縮した印刷バイナリデータを読み込む
  （上記２つは WEB サービスとの転送フォーマット）
- SaveSVGFile メソッド ・・・ SVG 形式の帳票画像データを書き出す
- SaveSVGZFile メソッド ・・・ SVGZ 形式の帳票画像データを書き出す

(インスタンス作成メソッド)

ReportCreator クラス ・・・上記の IReports 型で、
印刷又は、プレビュー等のインスタンス(オブジェクト)を返す

- GetPreview メソッド ・・・ プレビューオブジェクトを返す
- GetReport メソッド ・・・ 印刷オブジェクトを返す
- GetPdf メソッド ・・・ PDF オブジェクトを返す
- GetImagePdf メソッド ・・・ イメージ PDF オブジェクトを返す

※ReportStartImpl クラスも見えると思いますが、これはプレビュー単体起動用なので気にしないでください。

(プロパティ)

IReports インターフェース　･･･印刷又はプレビューを行うための共通インターフェース

- int AllPage　･･･全ページ数
- bool AccessFile　･･･ファイルにアクセスする事(ファイルに保存等)を許可する・しない
- bool AcceptDragDrop　･･･プレビュー画面へのファイルのドラッグ＆ドロップを許可する･しない
- bool CutByPage　･･･1 ページずつ用紙カットする・しない(ラベルプリンタ用)
- bool DisplayDialog　･･･印刷ダイアルログの表示する・しない
- bool DisplayPrinting　･･･印刷中(ページ数)の表示する・しない
- float MarginTop　･･･上部余白(印刷・プレビュー時のみ有効) mm 単位で指定
- float MarginLeft　･･･左側余白(印刷・プレビュー時のみ有効) mm 単位で指定
- bool PreviewDialog　･･･プレビューをダイアログ表示する・しない
- bool PreviewInTaskbar ･･･プレビューをタスクバーに表示する・しない
- bool SwapPdfImage　･･･PDF 出力中、画像データをスワップする・しない
- IObjects　z_Objects　･･･各印刷オブジェクトのプロパティ変更用静的クラス
  ※帳票デザイン時、各オブジェクトのプロパティ通りです。
  説明は割愛します。
- IPreviewWindow　z_PreviewWindow　･･･プレビュー画面情報。
  次ページで説明します。

それでは本書内ではコーディング例等を用いながら、各クラスやメソッドについてもう少し細かく書いていくことにします。
皆様が楽しんで楽にプログラミングできることを心から願います。

作者

Ps.
開発当初より、メソッドやプロパティが追加されております。
全てお客様のご要望により、都度実装していった機能でございます。
少し複雑になってきており、申し訳ございません。
ただし、基本的な印刷やプレビュー・PDF 出力といった機能を実現するロジックはいたってシンプルに作成することができます。

試用版インストーラにサンプルが付属しております。
是非、お試しください。

2014/11/4 作者

IPreviewWindow インターフェース(プレビュー画面情報)のプロパティ一覧

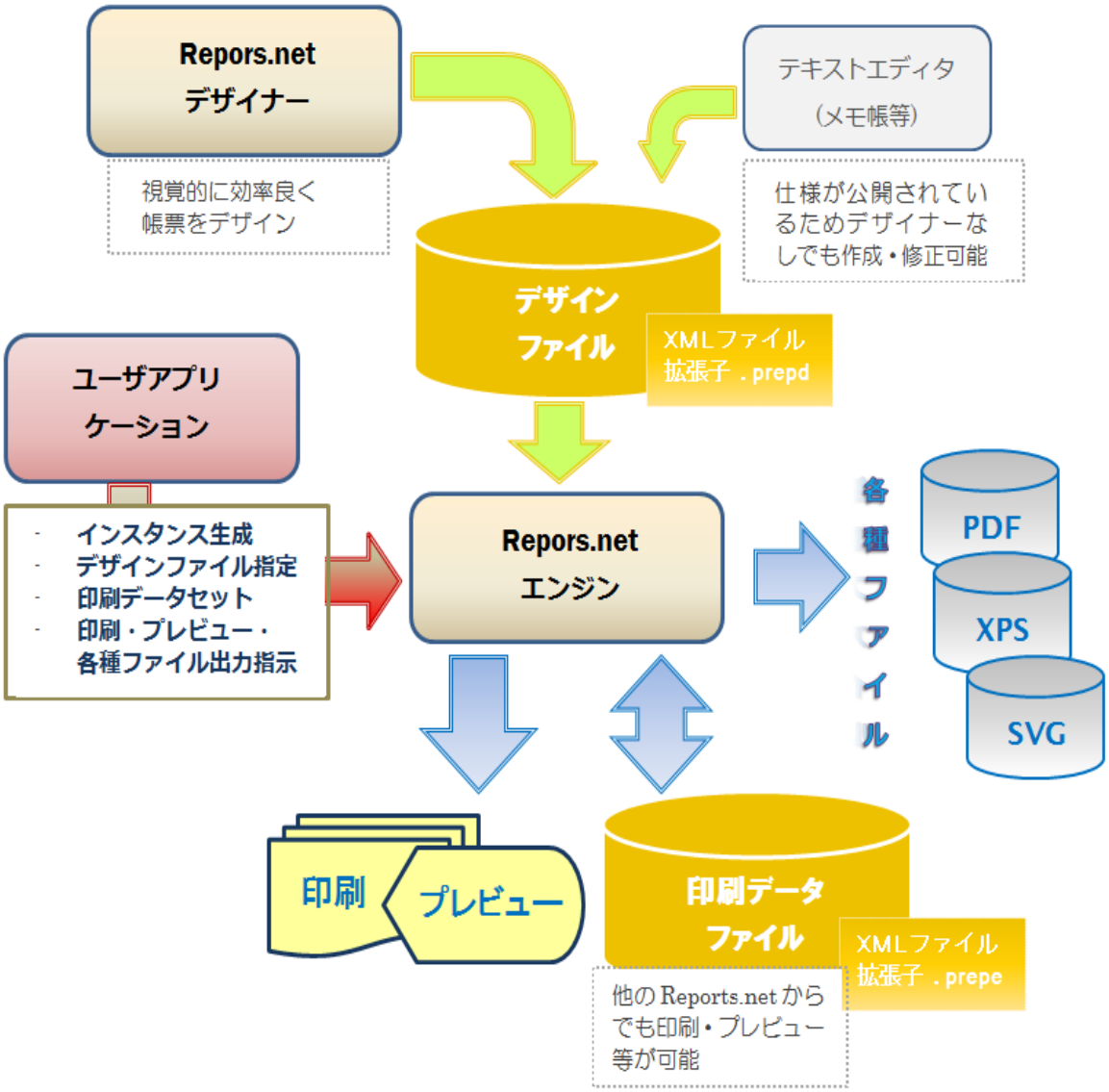
```
/// プレビュー画面タイトル
string z_TitleText

/// プレビュー画面アイコン
Icon z_Icon

/// プレビュー画面上位置(Y 座標)
int z_Top

/// プレビュー画面左位置(X 座標)
int z_Left

/// プレビュー画面幅
int z_Width

/// プレビュー画面高さ
int z_Height

/// プレビューウィンドウの最大化表示
bool z_Maximum

/// PDF 保存先
string z_SavePdfPath

/// 印刷データ保存先
string z_SaveXmlPath

/// プレビュー画面 開くボタン表示
bool z_VisibleOpenButton

/// プレビュー画面 保存ボタン表示
bool z_VisibleSaveButton

/// プレビュー画面 印刷ボタン表示
bool z_VisiblePrintButton

/// プレビュー画面 メニュー表示
bool z_VisibleMenu

/// バージョンウィンドウ
IVersionWindow z_VersionWindow

/// プレビュー表示倍率
double z_Zoom

/// プレビュー画面のツールバーに小さいアイコンを表示する場合、true を指定。
/// 既定値： false (大きいアイコン)
bool z_SmallToolbarIcon

/// プレビュー画面のツールバーに、テキストを表示する場合、true。
/// アイコンのみ出力して、テキストを表示しない場合、false。
/// 既定値： true (テキストを表示)
bool z_DispToolbarText
```

機能概要

【単体機能】

Reports.net の核となるのは、エンジンと呼ばれる部分です。エンジンは、.NET アプリケーションに対し、「デザインファイル」により定義された帳票を作成するための機能を提供します。利用者様は任意のアプリケーションからエンジンを制御し、帳票の印刷・プレビューや「印刷データ」の書き出しを行う事が可能です。

また、PDF や XPS(Microsoft Document Writer)、SVG 形式のファイルに出力することが可能で、ブラウザでプレビュー・印刷を行うことができます。



【Azure / AWS / GCP /、他 WEB サーバ Linux / Windows サーバ 対応】

Azure / AWS / GCP といった各種クラウドサービスや、オンプレミスを含むその他の WEB サーバに、帳票出力 WEB アプリケーションや帳票データ作成 WEB API を配置してご利用いただけます。Ver 9.0 で、.NET 5 / .NET 6 に対応したことにより、Linux サーバ上で動作するように対応いたしました。

お客様の開発実装とエンドユーザ様にご利用いただくまでの大まかな手順をその構成と共に以下に記載します。

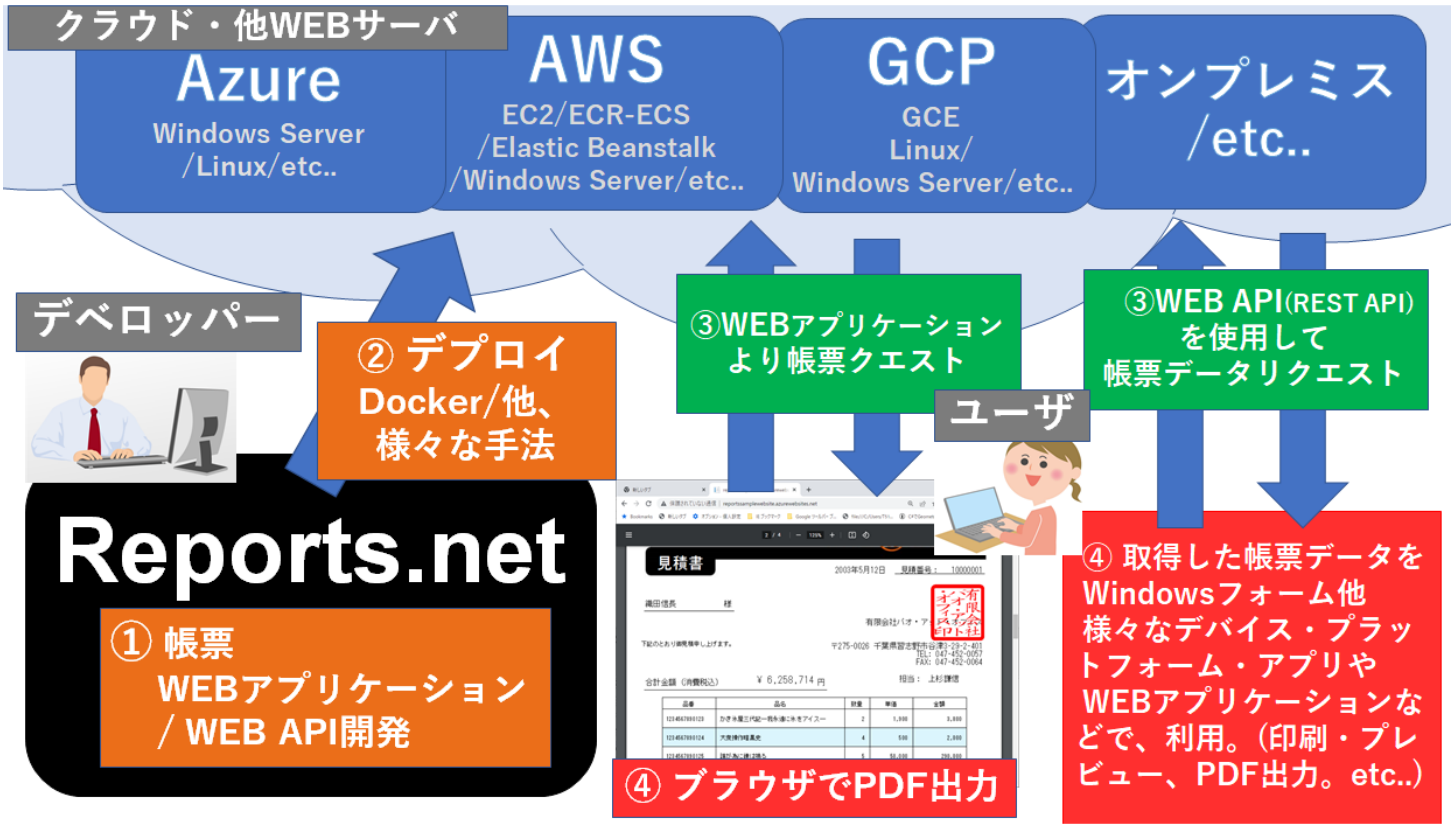
(1) Reports.net で、帳票出力 WEB アプリケーションや、帳票作成 WEB API を開発。

(2) 開発した帳票 WEB アプリケーションや WEB API を各種クラウドやその他 WEB サーバにデプロイ(Docker を含む様々な手順)して配置
※これらの開発手法やデプロイの手順については、本章の末尾に手順を解説した動画の URL を記載しておきます。

(3) WEB アプリケーションの場合、ユーザはブラウザより帳票をリクエスト。

(4) ブラウザに PDF 帳票が出力される。

(5) WEB API の場合、Windows フォームを含む、様々なデバイス・プラットフォーム・アプリ、または、WEB アプリケーションより、WEB API(WEB メソッド)を Call。
WEB API(WEB メソッド)の戻り値として取得した印刷データを、印刷・プレビュー・PDF 出力する等して利用する。



Azure / AWS / GCP /他 Linux / Windows サーバ用 開発・デプロイ手順動画

【WEB アプリケーション編】

帳票 WEB アプリケーションの作成方法や各種デプロイ手順など動画にまとめてございます。用途に応じてご覧ください。

1. WEB アプリで PDF 帳票出力開発手法紹介 / Azure への WEB アプリをデプロイ手順 / Azure SQL Server 使用
   https://youtu.be/6UI_pP-ws3c
2. Linux 上で動作する.NET5/6(C#)-帳票出力 WEB アプリ作成手順 - [WSL2 & Azure-Linux 編]
   https://youtu.be/OF3y7875BGo
3. 帳票出力 WEB アプリを AWS Linux Elastic Beanstalk へデプロイ - AWS Toolkit for Visual Studio 使用 - DynamoDB 使用- .NET6
   https://youtu.be/1wTuV2ffATg
4. .NET5 Docker の最も単純な方法で AWS-EC2 上で帳票出力 WEB アプリを動作させる手順
   https://youtu.be/0y3K3CW7DRM
5. .NET6 Docker の最も単純な方法で AWS-EC2 上で帳票出力 WEB アプリを動作させる手順
   https://youtu.be/UnPXcadLwFY
6. AWS ECS/ECR で帳票出力 WEB アプリを Docker で動作させる手順
   https://youtu.be/TQpeQGwGNmM
7. Docker の最も単純な方法で GCP 上で帳票出力 WEB アプリを動作させる手順
   https://youtu.be/YFdjUg9KgFo
8. 帳票出力 WEB アプリを複数クラウドにマルチデプロイ - Azure / AWS / GCP
   https://youtu.be/igApoNMri7k
9. 超簡単、帳票出力 WEB アプリをフォルダデプロイ方式で AWS EC2 へ
   https://youtu.be/3SE7hLNcOo8

- 動画チャンネルトップ

https://www.youtube.com/channel/UCYKjmyVrFhW_w_WhLU-wdqA?sub_confirmation=1

※これからも Reports.net を使った技術動画を随時追加していく予定です。

【WEB API 編】

帳票 WEB API の作成方法や各種デプロイ手順など動画にまとめてございます。
用途に応じてご覧ください。

1. 最短/最速 REST API(WEB API)実装 GET 編 / IIS+Windows Form クライアント編 / .NET5/6(.NET Framework 4.x も可
   https://youtu.be/cYEtHFpa8G4
2. 最短/最速 REST API(WEB API)実装 POST 編 / IIS+Windows Form クライアント編 / .NET5/6(.NET Framework 4.x も可
   https://youtu.be/EfIMRmMYU4A
3. Visual Studio から WEB API を IIS へデプロイ手順。.NET 5/.NET Framework 4.5 Roboto
4. [.NET5/6] REST API(WEB API)で帳票作成 / Windows Form クライアントで帳票出力
   https://youtu.be/BoIfww56aWY
5. [.NET5/6] REST API(WEB API)で帳票作成-2 SQL Server 編/Windows Form クライアントで出力
   https://youtu.be/VNeD7w3LdV0
6. 帳票出力 WEB API を複数クラウドにマルチデプロイ - Azure / AWS / GCP
   https://youtu.be/KW_RK8PmXro

- 動画チャンネルトップ
https://www.youtube.com/channel/UCYKjmyVrFhW_w_WhLU-wdqA?sub_confirmation=1

※これからも Reports.net を使った技術動画を随時追加していく予定です。

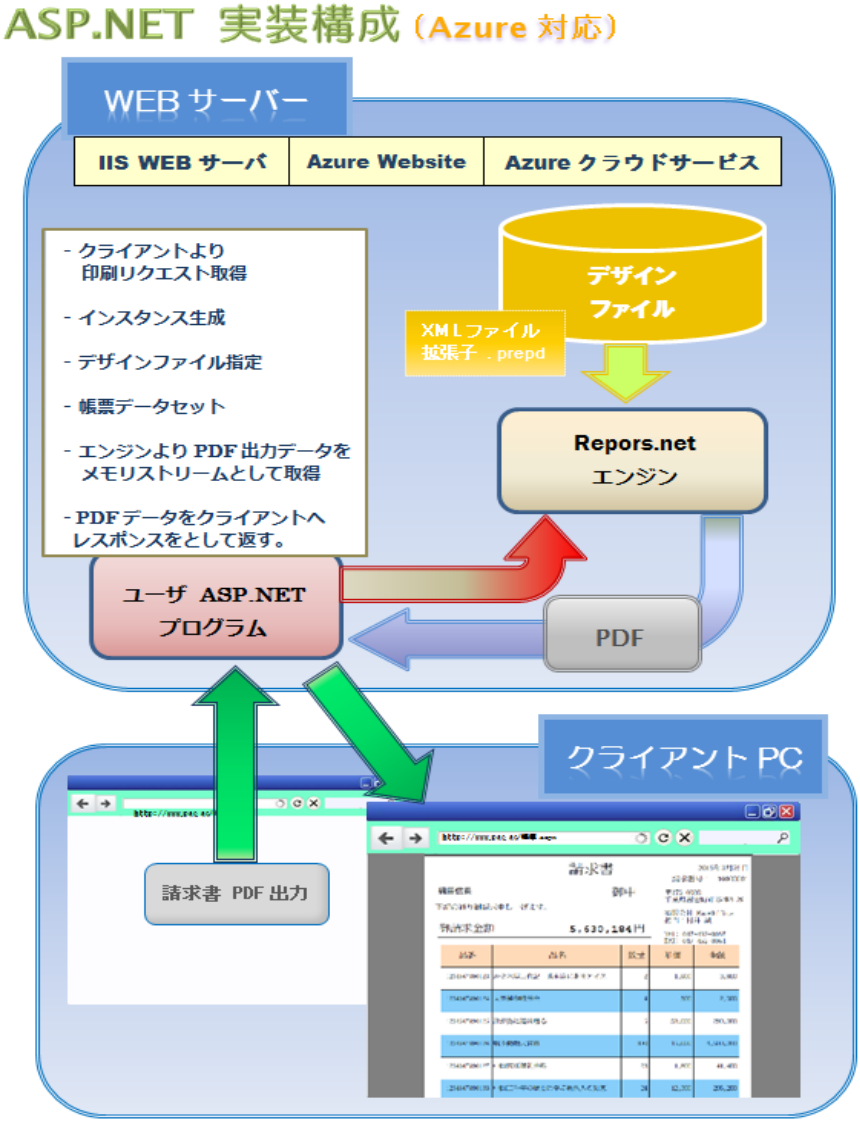
【ASP.NET で PDF 出力（Azure 対応)】- 旧.net framework 用

本機能は、Reports.net において互換のために実装してありますが、旧アーキテクチャになります。最新のアーキテクチャをご利用いただく場合、前章の .NET5 / .NET 6 により、Azure / AWS / GCP といったクラウドサービスの Linux / Windows サーバでの動作を可能としたアーキテクチャを参照してください。ただし、アーキテクチャが異なるだけで、概要としては、.NET5/.NET6 版と変わるところはございません。

.net framework 版の説明を続けさせていただきます。

ブラウザからの PDF 出力要求に対して、PDF データを生成しブラウザに PDF 出力を行うことができます。内部的にはリクエストに対するレスポンスに PDF データをセットします。ブラウザへの出力だけでなく PDF ファイルをダウンロードさせることも可能です。

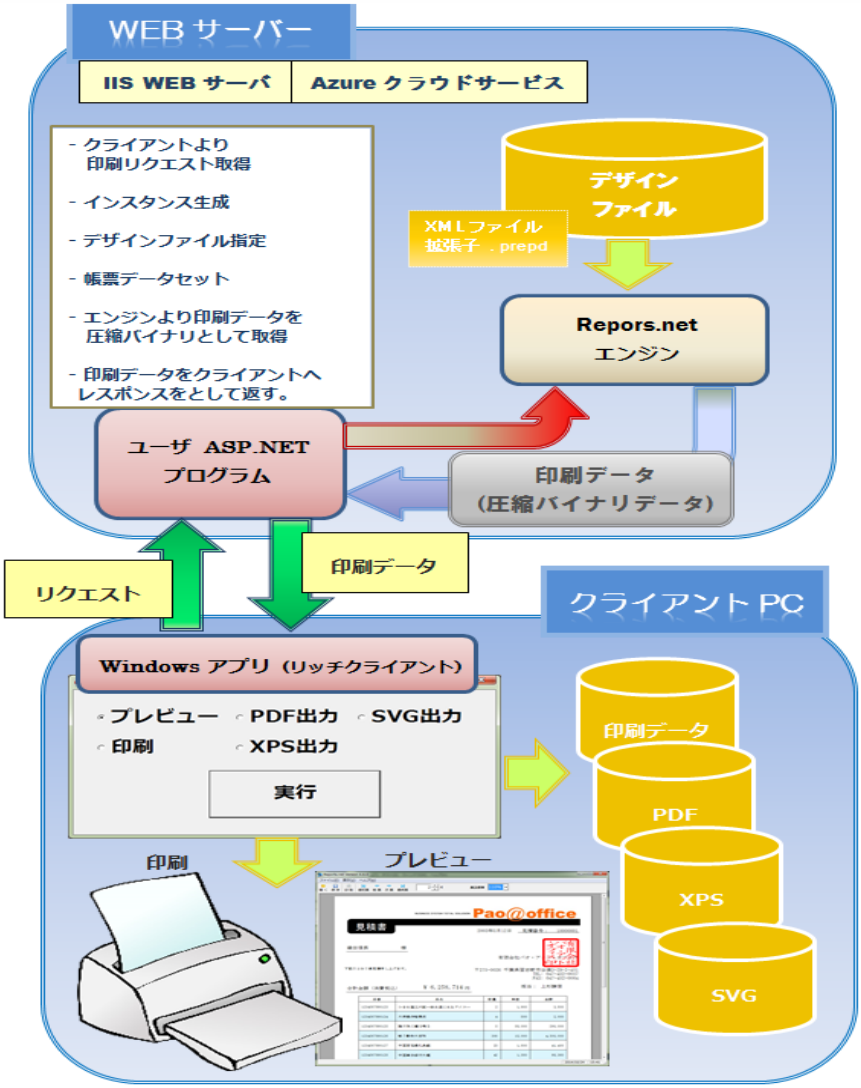
用意しているサンプルプログラムでは、PDF のブラウザ出力／PDF ファイルダウンロードのどちらも可能です。



【Azure クラウドと Windows アプリとの連携】 -旧.net framework 用

本機能は、Reports.net において互換のために実装してありますが、旧アーキテクチャになります。最新のアーキテクチャをご利用いただく場合、.NET5 / .NET 6 により、Azure / AWS / GCP といったクラウドサービスの Linux / Windows サーバ上での動作を可能としたアーキテクチャを参照してください。ただし、アーキテクチャが異なるだけで、概要としては、.NET5/.NET6 版と変わるところはございません。

.net framework 版の説明を続けさせていただきます。Windows をプラットフォームとしたリッチクライアントから、Asure や IIS、または、UNIX(Linux 等)サーバ上の WEB サービス(Azure クラウド /.NET WebService / axis 等)に対して 1 つの命令を下す(メソッドを呼び出す)だけで、WEB サーバから印刷データを圧縮したバイナリデータを取得し、印刷を行うことが可能です。クライアントから命令がきたら(メソッドが呼び出されたら)サーバ側のみでデータベース等にアクセスして印刷データを作成し、バイナリデータ(byte[]型変数)として、クライアント側に返し、クライアント側でそれを印刷するという仕組みです。



動作条件

本製品を使用するためには、以下の条件を満たす環境のパソコンが必要です。

開発環境	Windows 7 / 8 / 8.1 / 10 / 11 Windows Server 2008 / 2012 / 2013 / 2016 / 2019 / 2022 Microsoft Visual Studio 2005 / 2008 / 2010 / 2012 / 2013 / 2015 / 2017 / 2019 / 2022
開発言語	開発言語: VB.NET／C#／他 .net 用言語 ※.NET 5 / .NET 6 の WEB アプリケーションや WEB API の開発には、VB.NET はご利用いただけなくなりました。
実行環境	Windows 7 / 8 / 8.1 / 10 / 11 Windows Server 2008 / 2012 / 2013 / 2016 / 2019 / 2022 ※AnyCpu(32bit・64bit 共用) / 64bit 専用 別に製品ご提供。 Azure Linux / Azure Windows Server / AWS 各種 Linux(EC2 / ECR-EC/ Elastic Beans / 他) / AWS Windows Server / GCP GCE 各種 Linux / GCP GCE Windows Server ※Linux 上で WEB アプリケーションから帳票を出力する場合、通常の PDF 出力のみ対応しております。イメージ PDF には対応しておりません。 ※Linux 上に WEB API を配置した場合、WEB API を CALL する呼び出し元では、Reports.net の全ての機能をご利用いただけます。印刷・プレビュー・SVG 出力・XPS 出力・イメージ PDF 等
.Net Framework	.NET 5 / .NET 6 / .NET 7 .ner framework: 2.0 / 3.0 / 3.5 / 4.0 / 4.5 / 4.5.1 / 4.5.2 / 4.6 / 4.6.1 / 4.6.2 / 4.7 / 4.7.1 / 4.7.2 / 4.8 / 4.8.1 .NET / .net framework のバージョン別に製品ご提供。下位互換あり。

Reports.net 通常版 / Linux 版 / Azure 版利用方法

Reports.net のインストールを行うと次の３種類のアセンブリがインストールされます。

- Pao.Reports.dll
- Pao.Reports.Linux.dll
- Pao.Reports.Azure.dll

Reports.net を Windows でのみご利用いただく場合は、オンプレミスの WEB サーバを含めて、通常版の Pao.Reports.dll をプロジェクトに参照追加してご利用ください。

Reports.net を Linux の実行環境でご利用いただく場合、Pao.Reports.Linux.dll をプロジェクトに参照追加してご利用ください。

Azure Windows サーバ等につきましては、Pao.Reports.Linux.dll をご利用いただいても問題ございませんが、各種クラウドの Windows サーバが実行環境の場合、Pao.Reports.Azure.dll をご利用いただくと、Linux 版で出力できない制限であるイメージ PDF が出力可能となります。

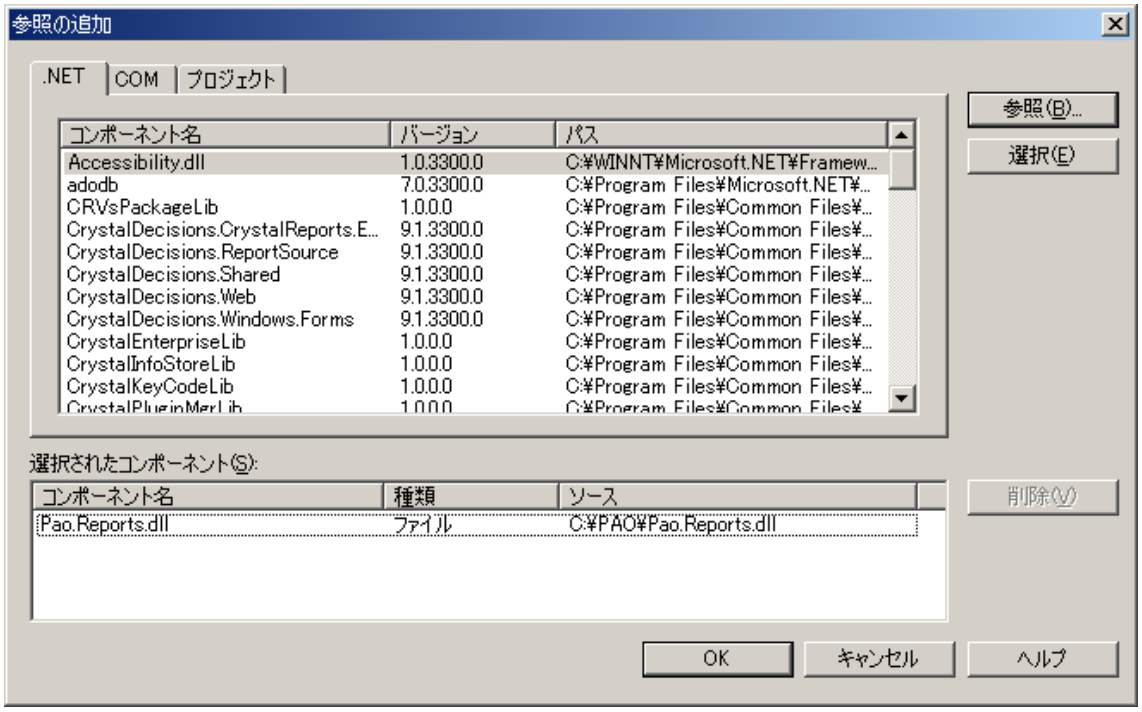
※イメージ PDF について
横道にそれてしまい申し訳ございませんが・・・
イメージ PDF は、通常 PDF の制限となっているハンコ画像の印影などの透過 Gif/Png の問題が解決されている他、フォントの種類の制限もございません。ただし、印刷イメージ画像を PDF 化している都合上､PDF 上で表示されている文字列は実際のテキストではないため、多少文字がギザギザになるなどの弊害もございます。弊社におきましては、実際の見積書や請求書などはイメージ PDF を使用しております。

使用方法

1. Reports.net をインストールしてください。最新版は常に製品サイトにご用意させて頂いております。

   http://www.pao.ac/reports.net/#download

   Reports.net をご利用になりたいプログラムのプロジェクトに、「Pao.Reports.dll」への参照を追加して下さい。



   「Pao.Reports.dll」「Pao.Reports.Linux.dll」「Pao.Reports.Azure.dll」は、以下のいずれかのインストール先フォルダにございます。

   C:¥Program Files (x86)¥Pao@Office¥Reports.net

   C:¥Program Files¥Pao@Office¥Reports.net

2. 必要に応じて C#の場合は「using」を、VB.NET の場合は「Imports」を定義して下さい。

   C#の場合

   ```
   using Pao.Reports;
   ```

   VB.NET の場合

   ```
   Imports Pao.Reports
   ```

3. Linux 版の注意点

前置きになりますが・・・
.NET 5 版 / .NET 6 版をインストールされた場合、付属のサンプルプログラムの
「09. .NET5 WEB アプリ -Azure,AWS,GCP,オンプレ」
「10. .NET5 WEB API - Azure,AWS,GCP,オンプレ」
のソースコード内に記述がございますので、是非、ご覧になってください。

その記述通り、Linux 上では、System.Drawing が利用できません。
従って、
using System.Drawing;
で、Color.Red などを使用して、帳票に色を指定している場合、これが使えません。
この場合、Pao.Reports.Linux に System.Drawing の代わりの定義などを入れてありますので、
using System.Drawing;
を
using Pao.Reports.Linux;
に変更してお使いください。
プログラム内の別の個所で、System.Drawing をお使いの場合は、
xxx.Color = Pao.Reports.Linux.Color.Red; として、使用してください。

繰り返しになりますが、
Linux 上では、System.Drawing を利用できないため、
例えば、Color の型は、System.Drawing.Color の型ではなくなっております。
Pao.Linux.Color 型になっております。

なお、Doker ビルド時の注意点がございます。
Doker ですので、Linux に限った話ではございませんが、Visual Studio では、DockerFile を右クリックしてビルドを行うことができます。ただし、不具合とも思われますが、参照アセンブリが遠い Path に存在すると、Dcoker コンテナの中に入れてくれません。
従って・・・
「09. .NET5 WEB アプリ -Azure,AWS,GCP,オンプレ」
「10. .NET5 WEB API - Azure,AWS,GCP,オンプレ」
のサンプルプログラムでも、Pao.Reports.Linux.dll 等のアセンブリは、近い Path にも配置して Dcoker ビルドが成功するようにしてあります。
Visual Studio で Docker ビルドを行う際には、アセンブリに限らず、Docker コンテナに入れるものは近い Path においておくことをお勧めします。

アプリケーションプログラムからの Reports.net 使用方法

例題サンプルプログラムの紹介

ここで説明に上げるサンプルプログラムは、インストールフォルダ内の
Samle¥1.programers(PDF・SZG 出力)
フォルダに納められています。
※現在、このサンプルプログラムは PDF 出力など、いくつかの機能が加えられており
説明の内容と異なる部分がございます。合致する部分を参考にされてください。

それでは、ここに示す例題サンプルプログラムにそって都度説明をしていきます。まず、大体のプログラムの流れを頭に入れておいて下さい。

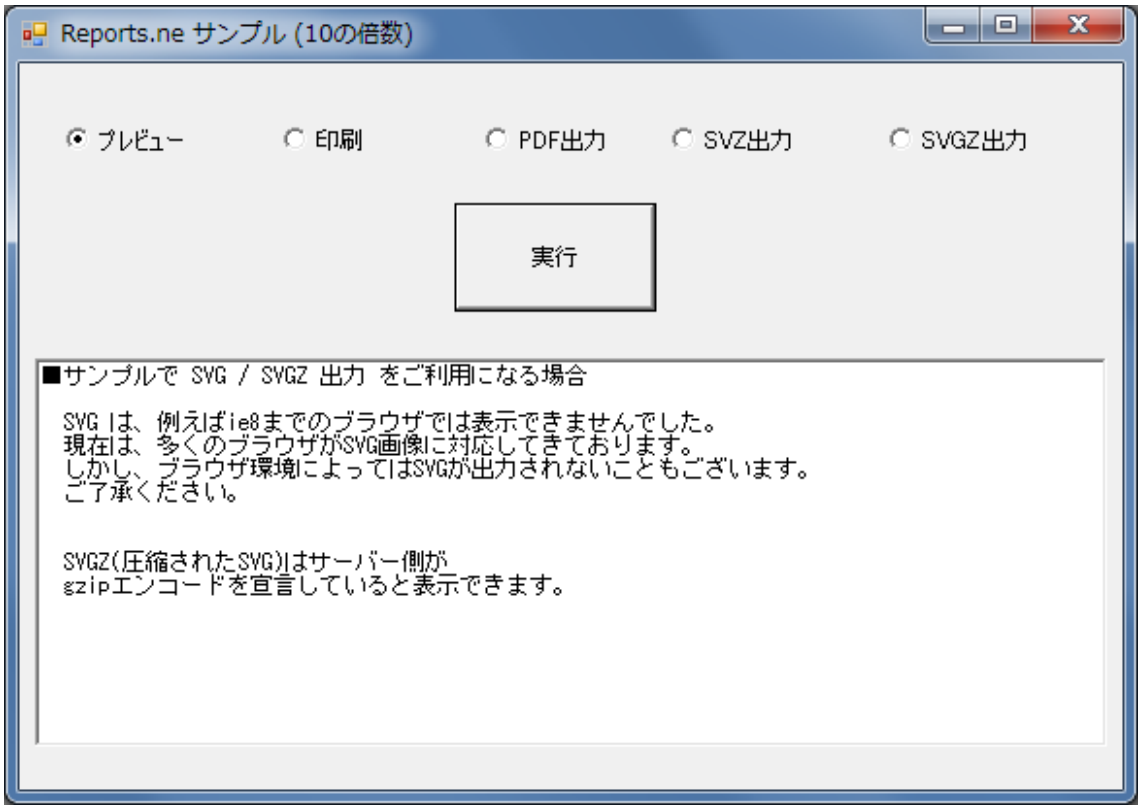
＜プログラムの説明＞

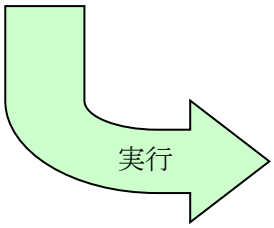
- 画面のラジオボタン(オプションボタン)にて、印刷又は、プレビューを選択されて、実行ボタンをクリックされた後、動作します。
- 帳票の各ページのヘッダに日時と頁数を書き込みます。
- 明細部は、６０回ループしてその行番号と、回数を１０倍した値を表に書き込みます。
- 明細部の各行は、横罫線で区切られます。
- 改ページの条件は１５行なので、全部で４ページになります。
- 以上の描画が済むと、画面の指示に従い、印刷、又はプレビューを行います。
- 最後に、一旦、印刷又はプレビューした印刷データを、印刷データファイルに保存し、もう一度その印刷データファイルを読み込んで、その印刷データをプレビュー表示します。

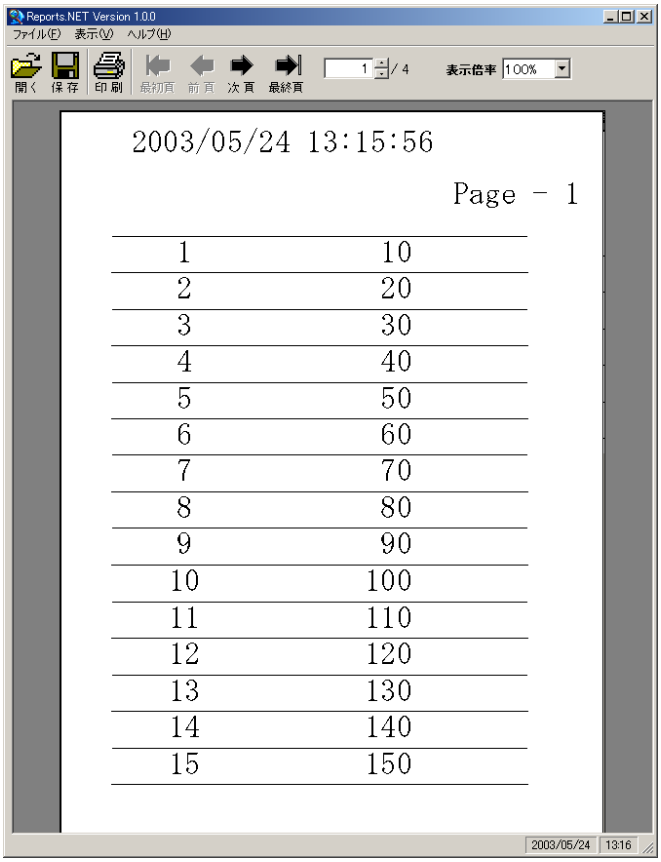
以上の処理を実現しているサンプルプログラムを C#.NET/VB.NET 共に作成しましたので、参考までに少し追ってみてください。コメントが入っておりますので、そこを読むだけでも構いません。

ここでは、サンプルプログラムの処理の流れを頭に入れておいてください。

＜サンプルプログラム実行時＞







C#の例

VB.NET の例

```
'IReport インターフェースで宣言(印刷・レポートどちらでも使える入れ物の用意)
 Dim paoRep As IReport = Nothing

 If radioButtonPreview.Checked = True Then 'ラジオボタンでプレビューが選択されている場合
     'プレビューオブジェクトのインスタンスを獲得
     paoRep = ReportCreator.GetPreview()
 Else
     '印刷オブジェクトのインスタンスを獲得
     paoRep = ReportCreator.GetReport()
 End If

 'レポート定義ファイルの読み込み
 paoRep.LoadDefFile("レポート定義ファイル.xml")

 Dim page As Integer = 0 '頁数を定義
 Dim line As Integer = 0 '行数を定義
 Dim i As Integer
 For i = 1 To 60
     If ((i - 1) Mod 15 = 0) Then '1頁15行で開始
         '頁開始を宣言
         paoRep.PageStart()
         page = page + 1  '頁数をインクリメント
         line = 0 '行数を初期化

         '＊＊＊ヘッダのセット＊＊＊
         '文字列のセット
         paoRep.Write("日付", System.DateTime.Now.ToString())
         paoRep.Write("頁数", "Page - " + page.ToString())

     End If
     line = line + 1 '行数をインクリメント

     '＊＊＊明細のセット＊＊＊
     '繰返し文字列のセット
     paoRep.Write("行番号", i.ToString(), line)
     paoRep.Write("10倍数", (i * 10).ToString(), line)
     '繰返し図形(横線)のセット
     paoRep.Write("横線", line)

     If ((i Mod 15) = 0) Then paoRep.PageEnd() '1頁15行で終了
 Next i

 '印刷/プレビューを実行
 paoRep.Output()

 paoRep.SaveXMLFile("印刷データファイル.xml") '印刷データの保存

 'プレビューオブジェクトのインスタンスを獲得しなおし(一旦初期化)
 paoRep = ReportCreator.GetPreview()

 paoRep.LoadXMLFile("印刷データファイル.xml") '印刷データの読み込み

 paoRep.Output() '  プレビューを実行
```

印刷・プレビューオブジェクトのインスタンス生成方法

印刷のクラスもプレビューのクラスも同じメソッドを共有しているため、
IReport インターフェースを使ってオブジェクトを宣言した後、
ReportCreator クラスのスタティックメソッドとなっている、

- IReport GetPreview()　・・・プレビューオブジェクトのインスタンス生成
- IReport GetReport()　・・・印刷オブジェクトのインスタンス生成
- IReport GetPDF()　・・・PDF 出力オブジェクトのインスタンス生成
- IReport GetImagePDF()　・・・イメージ PDF 出力オブジェクトのインスタンス生成

のいずれかを呼び出すことで、印刷・プレビュー・PDF 出力・イメージ PDF 出力オブジェクトのインスタンスを生成することができます。

＜C#.NET の例＞

```
//IReport インターフェースで宣言(印刷・プレビューどちらでも使える入れ物の用意)
IReport paoRep = null;

if(radioButtonPreview.Checked)//ラジオボタンでプレビューが選択されている場合
{
        //プレビューオブジェクトのインスタンスを獲得
        paoRep = ReportCreator.GetPreview();
}
else
{
        //印刷オブジェクトのインスタンスを獲得
        paoRep = ReportCreator.GetReport();
}
```

＜VB.NET の例＞

```
'IReport インターフェースで宣言(印刷・レポートどちらでも使える入れ物の用意)
Dim paoRep As IReport = Nothing

If radioButtonPreview.Checked = True Then'ラジオボタンでプレビューが選択されている場合
    'プレビューオブジェクトのインスタンスを獲得
    paoRep = ReportCreator.GetPreview()
Else
    '印刷オブジェクトのインスタンスを獲得
    paoRep = ReportCreator.GetReport()
End If
```

勿論、プレビューだけ行いたい場合等は・・・

```
IReport paoRep = ReportCreator.GetPreview();

Dim paoRep As IReport = ReportCreator.GetPreview()
```

のようにすることも可能です。

デザインファイル読み込み方法

プログラムから帳票にデータをセットする場合は、まず、デザイナーで作成されたデザインファイルを読み込みます。

※デザインファイルには、印刷時出力する各オブジェクトの位置や色など属性情報(プロパティ)が、XMLファイル形式で書き込まれております。

プログラムからデザインファイルを読み込むには、IReport インターフェースに実装されている LoadDefFile メソッドを使用します。LoadDefFile メソッドの引数に読み込むデザインファイルのパスを指定してください。

サンプルでは、相対パスになっておりますが、プログラムがどこで動作するかわからないため、絶対パスを指定することもできます。(例："C:¥Test¥デザインファイル.prepd")

＜C#.NET の例＞

```
//デザインファイルの読み込み
paoRep.LoadDefFile("デザインファイル.prepd");
```

＜VB.NET の例＞

```
'デザインファイルの読み込み
paoRep.LoadDefFile("デザインファイル.prepd")
```

デザインファイル変更方法

データをセットし直さずに、帳票のデザインだけ変更することができます。

プログラムからデザインファイルを変更するには、IReport インターフェースに実装されている ChangeDefFile メソッドを使用します。ChangeDefFile メソッドの引数に変更するデザインファイルのパスを指定してください。

サンプルでは、相対パスになっておりますが、プログラムがどこで動作するかわからないため、絶対パスを指定することもできます。(例："C:¥Test¥デザインファイル 2.prepd")

＜C#.NET の例＞

```
//デザインファイルの読み込み
paoRep.ChanegDefFile("デザインファイル2.prepd");
```

＜VB.NET の例＞

```
'デザインファイルの読み込み
paoRep.ChanegDefFile("デザインファイル2.prepd")
```

ページの開始・終了宣言の方法

プログラムから帳票にデータをセットする場合は、デザインファイルを読み込んだ後、ページ毎に、ページの開始宣言及びページの終了宣言をしなければなりません。
ページの開始宣言とページの終了宣言の間で帳票データをセットしますが、デザイナー等で作成されたデザインファイルの内容通り帳票を出力するのであれば、データのセットは不要です。
つまりプログラムからレポート定義体を読み込んで帳票を出力する最小構成は、

① 印刷・プレビューインスタンスの生成
② デザインファイルの読込
③ ページの開始宣言
④ ページの終了宣言
⑤ 印刷・プレビューの指示

ということになります。
通常の利用では、「③ページ開始宣言」と「④ページ終了宣言」の間に帳票データをセットするロジックが入る事がほとんどだと思います。

ページ開始宣言・ページ終了宣言を行うには、
IReport インターフェースに実装されている PageStart / PageEnd メソッドを使用します。
引数はありません。

＜C#.NET の例＞

```
//頁開始を宣言
paoRep.PageStart();
        :
//頁終了を宣言
paoRep.PageEnd();
```

Write() ･･･印刷データセット処理

＜VB.NET の例＞

```
'頁開始を宣言
paoRep.PageStart()
        :
'頁終了を宣言
paoRep.PageEnd()
```

Write() ･･･印刷データセット処理

オブジェクトへのデータセット方法(C#.NET 表記)

ここでは、どのようにしてデザインファイルで指定された各オブジェクトに対して値を入れたり、表の横罫線を繰返し描画するのかについて C#表記で述べていきます。
なお、オブジェクトのデータセットは、必ずページの開始宣言(PageStart)とページの終了宣言(PageEnd)の間で行ってください。

プログラムから帳票にデータをセットする場合は、IReport インターフェースに実装されている Write メソッドを使用します。Write メソッドは、３つのパターンにオーバーロードされています。

(1) void Write(string name, string value)

オブジェクトに対して文字列をセットします。
ヘッダやフッタなど繰り返さない固定オブジェクトの値のセットに使用してください。

string name

デザインファイル内のオブジェクト名を指定します。
この場合、文字列をセットするため対象となるオブジェクトのタイプは、基本的に、Text(文字列) と、ArtText(装飾文字列) と、Barcode(バーコード) のみとなります。
オブジェクトを削除したいときに、それ以外のオブジェクトを指定します。

string value

Text(文字列) と、ArtText(装飾文字列)オブジェクトの場合、セットする文字列を指定します。
それ以外のオブジェクトに空文字(“”)を指定した場合、そのオブジェクトを削除します。

(2) void Write(string name, string value, long index)

オブジェクトに対して描画位置を指定して文字列をセットします。

表の行など繰り返し値をセットするオブジェクトに使用してください。

このパターンのメソッドを使用する場合、デザインファイル内の IntervalX 又は IntervalY に１以上の値が入っている必要があります。

IntervalX とは、横方向に繰り返す間隔(mm)です。

IntervalY とは、縦方向に繰り返す間隔(mm)です。主に表の行などに使用されます。

string name

デザインファイル内のオブジェクト名を指定します。

この場合、文字列をセットするため対象となるオブジェクトのタイプは、基本的に、Text(文字列) と、ArtText(装飾文字列)と、Barcode(バーコード) のみとなります。

オブジェクトを削除したいときに、それ以外のオブジェクトを指定します。

string value

Text(文字列) と、ArtText(装飾文字列)オブジェクトの場合、セットする文字列を指定します。

それ以外のオブジェクトに空文字(“”)を指定した場合、そのオブジェクトを削除します。

long index

IntervalX／IntervalY で指定された縦方向・横方向の間隔で描画を行うページ内の描画位置です。左上から右下方向に値が大きくなります。

例えば表の場合、IntervalY に行間隔をあらかじめ設定しておき、

この index に行数をセットしていきます。

プログラム的に書くと、

出力(印字)位置＝オブジェクト(の最初)の位置＋InterbalY×(index –1)

となります。

(3) void Write(string name, long index)

オブジェクトに対して描画位置を指定します。

表の行の横罫線など繰り返し描画を行うオブジェクトに使用してください。

このパターンのメソッドを使用する場合、デザインファイル内の IntervalX 又は IntervalY に１以上の値が入っている必要があります。

IntervalX とは、横方向に繰り返す間隔(mm)です。

IntervalY とは、縦方向に繰り返す間隔(mm)です。主に表の行などに使用されます。

string name

デザインファイル内のオブジェクト名を指定します。

この場合、どのオブジェクトでも繰返し描画ができるため対象となるオブジェクトのタイプは、全てのオブジェクトです。

long index

IntervalX／IntervalY で指定された縦方向・横方向の間隔で描画を行うページ内の描画位置です。左上から右下方向に値が大きくなります。

例えば表の場合、IntervalY に行間隔をあらかじめ設定しておき、

この index に行数をセットしていきます。

プログラム的に書くと、

出力(印字)位置＝オブジェクト(の最初)の位置＋InterbalY×(index –1)

となります。

＜C#.NET の例＞

```
int page = 0; //頁数を定義
int line = 0; //行数を定義
for (int i = 0; i < 60; i++)
{
        if (i % 15 == 0) //1頁15行で開始
        {
                //頁開始を宣言
                paoRep.PageStart();
                page++;          //頁数をインクリメント
                line = 0;        //行数を初期化
                //＊＊＊ヘッダのセット＊＊＊
                //文字列のセット
                paoRep.Write("日付", System.DateTime.Now.ToString());
                paoRep.Write("頁数", "Page - " + page.ToString());
        }
        line++; //行数をインクリメント

        //＊＊＊明細のセット＊＊＊
        //繰返し文字列のセット
        paoRep.Write("行番号", (i+1).ToString() , line);
        paoRep.Write("10倍数", ((i+1)*10).ToString() , line);
        //繰返し図形(横線)のセット
        paoRep.Write("横線", line);

        if (((i+1) % 15) == 0) paoRep.PageEnd(); //1頁15行で終了宣言
}
```

Ver 7.0.0 以降で主に行の一方向への繰り返しに加え、同時に横の両方向へオブジェクトを繰り返すことが可能になりました。POP チラシや名刺など、1 ページ内の縦横に同一フォーマットを出力する場合等にご利用ください。
Write の各メソッドの index　を、indexX, indexY へ引数が追加になります。
long indexX: 列数インデックス(1~)
long indexY: 行数インデックス(1~)

追加となったメソッドは次の通りです。
(4) void Write(string name, string value, long indexX, long indexY)
(5) void Write(string name, long indexX, long indexY)

オブジェクトへのデータセット方法(VB.NET 表記)

ここでは、どのようにしてデザインファイルで指定された各オブジェクトに対して値を入れたり、表の横罫線を繰返し描画するのかについて VB.NET 表記で述べていきます。
なお、オブジェクトのデータセットは、必ずページの開始宣言(PageStart)とページの終了宣言(PageEnd)の間で行ってください。

プログラムから帳票にデータをセットする場合は、IReport インターフェースに実装されている Write() メソッドを使用します。Write()メソッドは、3 つのパターンにオーバーロードされています。

(1) Sub Write(name As String, value As String)

オブジェクトに対して文字列をセットします。
ヘッダやフッタなど繰り返さない固定オブジェクトの値のセットに使用してください。

name As String

デザインファイル内のオブジェクト名を指定します。
この場合、文字列をセットするため対象となるオブジェクトのタイプは、基本的に、Text(文字列) と、ArtText(装飾文字列) と、Barcode(バーコード) のみとなります。
オブジェクトを削除したいときに、それ以外のオブジェクトを指定します。

value As String

Text(文字列) と、ArtText(装飾文字列)オブジェクトの場合、セットする文字列を指定します。
それ以外のオブジェクトに空文字(“”)を指定した場合、そのオブジェクトを削除します。

Sub Write(name As String, value As String, index As Long)

オブジェクトに対して描画位置を指定して文字列をセットします。

表の行など繰り返し値をセットするオブジェクトに使用してください。

このパターンのメソッドを使用する場合、デザインファイル内の IntervalX 又は IntervalY に１以上の値が入っている必要があります。

IntervalX とは、横方向に繰り返す間隔(mm)です。

IntervalY とは、縦方向に繰り返す間隔(mm)です。主に表の行などに使用されます。

name As String

デザインファイル内のオブジェクト名を指定します。

この場合、文字列をセットするため対象となるオブジェクトのタイプは、基本的に、Text(文字列) と、ArtText(装飾文字列) と、Barcode(バーコード) のみとなります。

オブジェクトを削除したいときに、それ以外のオブジェクトを指定します。

value As String

Text(文字列) と、ArtText(装飾文字列)オブジェクトの場合、セットする文字列を指定します。

それ以外のオブジェクトに空文字(“”)を指定した場合、そのオブジェクトを削除します。

index As Long

IntervalX／IntervalY で指定された縦方向・横方向の間隔で描画を行うページ内の描画位置です。左上から右下方向に値が大きくなります。

例えば表の場合、IntervalY に行間隔をあらかじめ設定しておき、

この index に行数をセットしていきます。

プログラム的に書くと、

出力(印字)位置＝オブジェクト(の最初)の位置＋InterbalY×(index –1)

となります。

(2) Sub Write(name As String, index As Long)

オブジェクトに対して描画位置を指定します。

表の行の横罫線など繰り返し描画を行うオブジェクトに使用してください。

このパターンのメソッドを使用する場合、デザインファイル内の IntervalX 又は IntervalY に１以上の値が入っている必要があります。

IntervalX とは、横方向に繰り返す間隔(mm)です。

IntervalY とは、縦方向に繰り返す間隔(mm)です。主に表の行などに使用されます。

name As String

デザインファイル内のオブジェクト名を指定します。

この場合、どのオブジェクトでも繰返し描画ができるため対象となるオブジェクトのタイプは、全てのオブジェクトです。

index As Long

IntervalX／IntervalY で指定された縦方向・横方向の間隔で描画を行うページ内の描画位置です。左上から右下方向に値が大きくなります。

例えば表の場合、IntervalY に行間隔をあらかじめ設定しておき、

この index に行数をセットしていきます。

プログラム的に書くと、

出力(印字)位置＝オブジェクト(の最初)の位置＋InterbalY×(index –1)

となります。

＜VB.NET の例＞

```
Dim page As Integer = 0 '頁数を定義
Dim line As Integer = 0 '行数を定義
For i = 1 To 60
    If ((i - 1) Mod 15 = 0) Then '1頁15行で開始
        '頁開始を宣言
        paoRep.PageStart()
        page = page + 1  '頁数をインクリメント
        line = 0 '行数を初期化

        '＊＊＊ヘッダのセット＊＊＊
        '文字列のセット
        paoRep.Write("日付", System.DateTime.Now.ToString())
        paoRep.Write("頁数", "Page - " + page.ToString())
    End If
    line = line + 1 '行数をインクリメント

    '＊＊＊明細のセット＊＊＊
    '繰返し文字列のセット
    paoRep.Write("行番号", i.ToString(), line)
    paoRep.Write("10倍数", (i * 10).ToString(), line)
    '繰返し図形(横線)のセット
    paoRep.Write("横線", line)

    If ((i Mod 15) = 0) Then paoRep.PageEnd() '1頁15行で終了
Next i
```

Ver 7.0.0 以降で主に行の一方向への繰り返しに加え、同時に横の両方向へオブジェクトを繰り返すことが可能になりました。POP チラシや名刺など、1 ページ内の縦横に同一フォーマットを出力する場合等にご利用ください。
Write の各メソッドの index　を、indexX, indexY へ引数が追加になります。
indexX: 列数インデックス(1~)
indexY: 行数インデックス(1~)

追加となったメソッドは次の通りです。

(3) Sub Write(name As String, value As String, indexX As Long, indexY As Long)
(4) Sub Write(name As String, indexX As Long, indexY As Long)

印刷・プレビューの指示方法

各帳票のオブジェクトへデータのセットが終わり、最後のページ終了宣言(PageEnd)が終わると、印刷・プレビューを行う事ができます。
プログラムから印刷/プレビューを行うには、IReport インターフェースに実装されている Output メソッドを使用します。引数はありません。

＜C#.NET の例＞

```
IReport paoRep = ReportCreator.GetReport();  // or GetPreview()
paoRep.LoadDefFile("./デザイン.prepd");
paoRep.PageStart();
paoRep.Write("項目１","ああああ");
paoRep.Write("項目２","いいい");
paoRep.Write("項目３","ううう");
paoRep.PageEnd();
paoRep.Output();// 印刷/プレビューを実行
```

＜VB.NET の例＞

```
Dim paoRep As IReport = ReportCreator.GetReport() // or GetPreview()
paoRep.LoadDefFile("./デザイン.prepd ")
paoRep.PageStart()
paoRep.Write("項目１","ああああ")
paoRep.Write("項目２","いいい")
paoRep.Write("項目３","ううう")
paoRep.PageEnd()
paoRep.Output() // 印刷/プレビューを実行
```

WPF 版 印刷プレビューの指示方法

各帳票のオブジェクトへデータのセットが終わり、最後のページ終了宣言(PageEnd)が終わると、WPF 版プレビューを行う事ができます。
プログラムから印刷/プレビューを行うには、IReport インターフェースに実装されている Output メソッドを使用します。引数はありません。

＜C#.NET の例＞

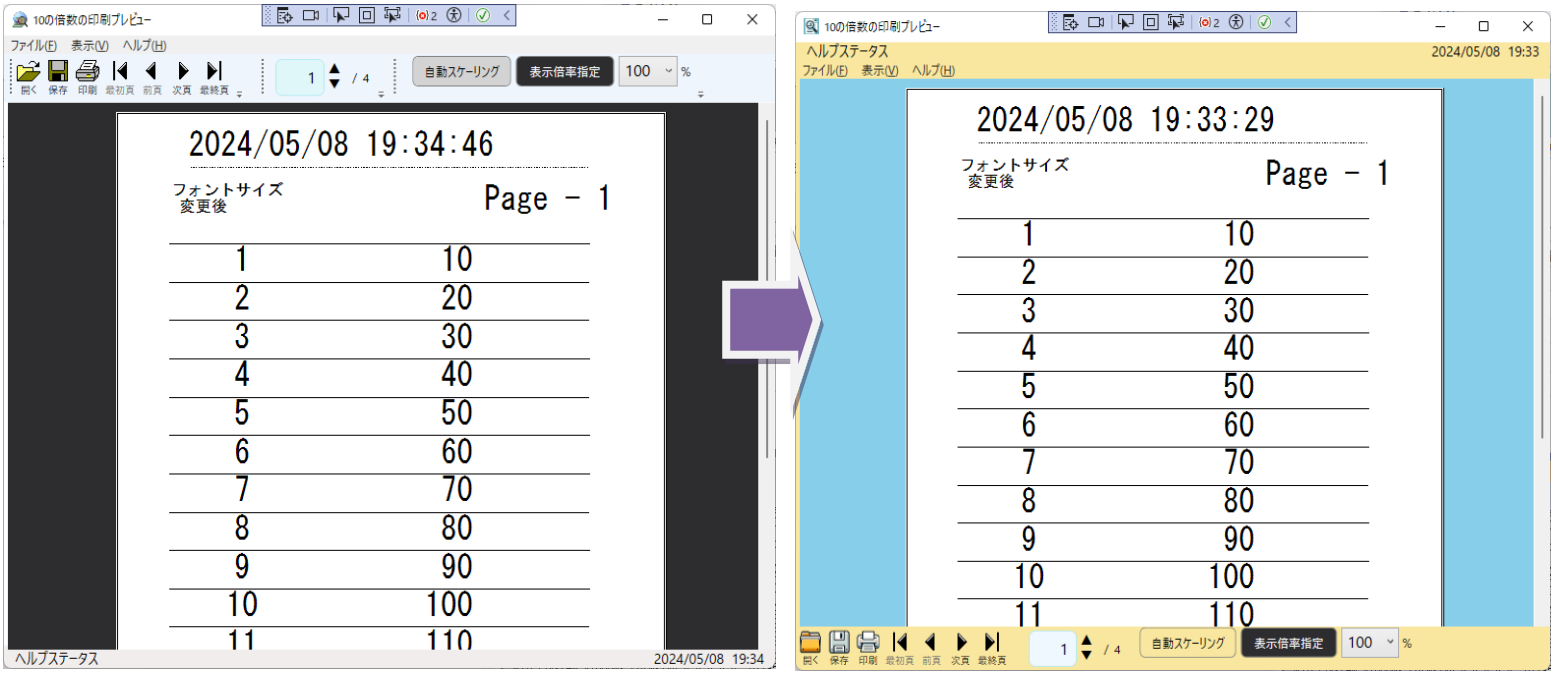
```
IReport paoRep = ReportCreator.GetPreviewWpf();
paoRep.LoadDefFile("./デザイン.prepd");
paoRep.PageStart();
paoRep.Write("項目１","ああああ");
paoRep.Write("項目２","いいい");
paoRep.Write("項目３","うううう");
paoRep.PageEnd();
paoRep.Output();// 印刷/プレビューを実行
```

＜VB.NET の例＞

```
Dim paoRep As IReport = ReportCreator.GetPreviewWpf()
paoRep.LoadDefFile("./デザイン.prepd ")
paoRep.PageStart()
paoRep.Write("項目１","ああああ")
paoRep.Write("項目２","いいい")
paoRep.Write("項目３","うううう")
paoRep.PageEnd()
paoRep.Output() // 印刷/プレビューを実行
```

WPF版では、プレビュー画面のデザインを自在に変更できます。

画面の変更例



カスタマイズは次の３つの方法で行えます。

1. XAML でのデザイン変更

   Custom.xaml ファイルをプログラムフォルダ(.exe のあるフォルダ)に配置して、プレビュー画面を自由にカスタマイズ。

   Custom.xaml の一例を、Reports.net インストーラで一緒にインストールされるサンプルプログラム- SAMPLE フォルダの「11.XAML (WPF 版プレビュー・XPS 出力)」フォルダ配下に配置しております。XAML の変更例の詳細をご覧になる場合は、XAML のテキストファイルをご用意しております。

   変更前デフォルト xaml　**Custom.xaml のサンプル**

   また、プレビュー画面表示中に、Custom.xaml を変更後、F5 キーを押すことで、変更したデザインが、表示中のプレビュー画面に反映されます。

   この操作により、ある程度視覚的にデザインを行うことができます。

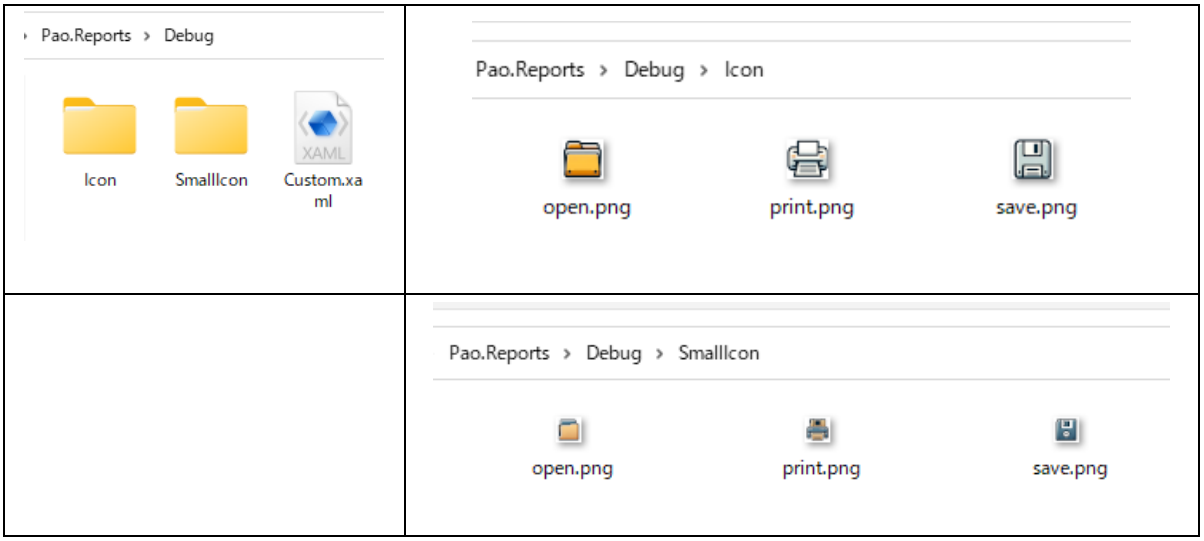
2. ウィンドウズ内アイコンの変更

   Icon および SmallIcon フォルダをプログラムフォルダに作成し、使用しているアイコンに対応する画像ファイル

   open.png, print.png, save.png, first.png, revious.png, next.png, last.png

   のうち[変更する画像のみ]配置。

変更アイコン配置例



3. ウィンドウズのアイコンの変更
   次のロジックで Windows Form 版プレビュー画面同様ウィンドウズ自体のアイコンを変更できます。ウィンドウズタイトルの変更も可能です。

```
// プレビューウィンドウのアイコン・タイトルの変更
paoRep.z_PreviewWindowWpf.z_Icon = new System.Drawing.Icon("./PreviewCustom.ico");
paoRep.z_PreviewWindowWpf.z_TitleText = "カスタムプレビュー画面";
```

参考動画

これらのカスタマイズ手順を実行する具体的な方法については、下記の参考動画をご覧ください。動画では、XAML ファイルの編集からアイコンの置き換えまで、プレビュー画面をカスタマイズする実際のプロセスをステップバイステップで説明しており、お客様の開発するアプリケーションやシステムに合わせたプレビュー画面の作成に役立つヒントやテクニックを提供しています。

動画を見る

独自印刷・プレビューの指示方法

各帳票のオブジェクトへデータのセットが終わり、最後のページ終了宣言(PageEnd)が終わり、Output()メソッドで印刷・プレビューを行うのと同じタイミングで、
印刷データがセットされている PrintDocument を取得することができます。
この PrintDocument を利用して、お客様独自のプレビューや印刷を行うことができます。
PrintDocument を取得するには、IReport インターフェースに実装されている GetPrintDocunment メソッドを使用します。引数はありません。

＜C#.NET の例＞

```
IReport paoRep = ReportCreator.GetReport();  // or GetPreview()
paoRep.LoadDefFile("./デザイン.prepd");
paoRep.PageStart();
paoRep.Write("項目１","あああ");
paoRep.Write("項目２","いいい");
paoRep.Write("項目３","ううう");
paoRep.PageEnd();

// PrintDocument取得
printDocument1 = paoRep.GetPrintDocument();

// このフォームのプレビューコントロールへ プレビュー実行
printPreviewControl1.Document = printDocument1;
printPreviewControl1.InvalidatePreview();
```

＜VB.NET の例＞

```
Dim paoRep As IReport = ReportCreator.GetReport() // or GetPreview()
paoRep.LoadDefFile("./デザイン.prepd ")
paoRep.PageStart()
paoRep.Write("項目１","あああ")
paoRep.Write("項目２","いいい")
paoRep.Write("項目３","ううう")
paoRep.PageEnd()

// PrintDocument取得
printDocument1 = paoRep.GetPrintDocument()

// このフォームのプレビューコントロールへ プレビュー実行
printPreviewControl1.Document = printDocument1
printPreviewControl1.InvalidatePreview()
```

PDF 出力方法

PDF 出力を行うには、まず、PDF 出力用のインスタンスを生成する必要があります。
IReport インターフェースに実装されている GetPDF メソッド、または、GetImagePDF メソッドを使用してインスタンスを取得してください。
インスタンス取得後、これまでの説明と同様に、
① デザインファイルの読込 (LoadDefFile メソッド)
② ページの開始宣言 (PageStart メソッド)
③ データのセット (Write メソッド)
④ ページの終了宣言 (PageEnd メソッド)
を行ってください。

各帳票オブジェクトへデータのセットが終わり、最後のページ終了宣言(PageEnd)が終わると、PDF ファイルへ出力を行うことができます。印刷・プレビューを行うタイミングと同じタイミングです。
プログラムから PDF 出力を行うには、IReport インターフェースに実装されている SavePDF メソッドを使用します。
引数は、PDF ファイル名、又は、ストリーム(System.IO.Stream) です。

＜C#.NET の例＞
```
'PDFオブジェクトのインスタンスを獲得
paoRep = ReportCreator.GetPDF();  または、paoRep = ReportCreator.GetImagePDF();
        :
        :
        :
// PDF出力
paoRep.SavePDF();
```

＜VB.NET の例＞
```
'PDFオブジェクトのインスタンスを獲得
paoRep = ReportCreator.GetPDF()  または、paoRep = ReportCreator.GetImagePDF()
        :
        :
        :
' PDF出力
paoRep.SavePDF()
```

印刷データの保存・読み込み方法

Reports.net では、印刷データをそのまま XML ファイルに保存し、それを読み込むことができます。例えば、WEB アプリケーションとクライアントとの通信などで、サーバ側でデータベース検索して帳票を作成し、そのデータをクライアントが受け取るようなことが可能なのです。
印刷データの保存を行えるタイミングは、印刷・プレビュー指示をするときと同じです。また、印刷・プレビューも同時に行う場合、印刷データの保存は、印刷・プレビューの前後どちらで行っても構いません。

印刷データの読込は、印刷・プレビューのインスタンスが作成されていれば、いつでも可能です。例えば以下のような手順で読み込んだ印刷データを印刷・プレビューすることができます。
① 印刷または、プレビューインスタンスの生成
② 印刷データファイルの読込
③ 印刷または、プレビューの指示

プログラムから印刷データの保存を行うには、IReport インターフェースに実装されている SaveXMLFile メソッドを使用します。引数はありません。
プログラムから印刷データの読込を行うには、IReport インターフェースに実装されている LoadXMLFile メソッドを使用します。引数はありません。

＜C#.NET の例＞

```
paoRep.SaveXMLFile("印刷データ.XML"); //印刷データの保存

//プレビューオブジェクトのインスタンスを獲得しなおし(一旦初期化)
paoRep = ReportCreator.GetPreview();

paoRep.LoadXMLFile("印刷データ.XML"); //印刷データの読み込み

paoRep.Output(); // プレビューを実行
```

＜VB.NET の例＞

```
  paoRep.SaveXMLFile("印刷データファイル.xml") '印刷データの保存

  'プレビューオブジェクトのインスタンスを獲得しなおし(一旦初期化)
  paoRep = ReportCreator.GetPreview()

  paoRep.LoadXMLFile("印刷データファイル.xml") '印刷データの読み込み

  paoRep.Output() ' プレビューを実行
```

圧縮した印刷バイナリデータ取得 (Web サービス用)

プログラムから圧縮した印刷バイナリデータの取得を行うには、IReport インターフェースに実装されている SaveData メソッドを使用します。引数はありません。
プログラムから圧縮した印刷バイナリデータの読込みを行うには、IReport インターフェースに実装されている LoadData メソッドを使用します。引数は印刷データ（ZIP 形式）のファイル名です。

SVG、SVGZ 出力方法

プログラムから SVG 形式の印刷データの書き出しを行うには、IReport インターフェースに実装されている SaveSVGFile メソッドを使用します。引数は SVG 形式のデータを保存するファイル名(拡張子は html)です。
プログラムから SVGZ 形式の印刷データの書き出しを行うには、IReport インターフェースに実装されている SaveSVGZFile メソッドを使用します。引数は SVGZ 形式のデータを保存するファイル名(拡張子は html)です。

プログラマーズリファレンス

IReport インターフェース

Reports.net を制御する全てのメソッドを保持しているインターフェースです。
ReportCreator クラスの持つ GetPreview メソッド 及び、GetReport メソッドによりインスタンスを生成することが可能です。プレビュー時には、GetPreview にてインスタンスを生成し、印刷時には、GetReport にてインスタンスを生成してください。

コンストラクタ

引数なし

パブリックメソッド

LoadDefFile	デザインファイルを読み込む
ChangeDefFile	デザインファイル(デザイン)を変更。データはセットし直さない。
PageStart	ページの開始を宣言する
PageEnd	ページの終了を宣言する
Write	印刷データを書き込む
Output	印刷/プレビューを指示する
GetPrintDocument	独自プレビュー・印刷用 PrintDocument を取得する
SaveXMLFile	印刷データファイルを書き出す
LoadXMLFile	印刷データファイルを読み込む
SaveData	圧縮した印刷バイナリデータを返す
LoadData	圧縮した印刷バイナリデータを書き出す
SaveSVGFile	SVG 形式の印刷データを書き出す
SaveSVGZFile	SVGZ 形式の印刷データを書き出す
SavePDF	PDF 形式の印刷データを書き出す

パブリックプロパティ

AccessFile	プレビュー画面からファイルアクセスを許可する
AllPage	印刷・プレビューオブジェクトの全ページ数
CutByPage	1 ページずつ用紙をカットするか指定(シール・ラベルプリンタ)
DisplayDialog	印刷(Output)時[印刷]ダイアログを表示するかを指定
DisplayPrinting	印刷中(ページ数)を表示する
MarginTop	上部余白(印刷・プレビュー時のみ有効) mm 単位で指定
MarginLeft	左側余白(印刷・プレビュー時のみ有効) mm 単位で指定
SwapPdfImage	PDF 出力中画像データスワップするかを指定
PreviewDialog	プレビュー画面をダイアログ表示するかを指定
z_Objects	デザイン時オブジェクトの属性取得・設定用静的クラス

ReportCreator クラス

印刷、又は、プレビューを行うオブジェクトを返すメソッドを実装したクラスです。
IReport 型の GetPreview メソッド 及び、GetReport メソッドを内蔵しています。
プレビューを行うときは、GetPreview メソッドを呼び出してください。
印刷を行うときは、GetReport メソッドを呼び出してください。

パブリックメソッド

GetPreview	プレビューオブジェクトを返す
GetReport	印刷オブジェクトを返す
GetPdf	PDF オブジェクトを返す
GetImagePdf	イメージ PDF オブジェクトを返す

GetPreview メソッド

プレビューを制御するオブジェクトを返すメソッドです。
直接印刷するときは、GetReport メソッドを使用してください。

＜C#.NET＞

IReport GetPreview()

＜VB.NET＞

Function GetPreview() As IReport

＜C#.NET の例＞

```
//プレビューオブジェクトのインスタンスを獲得
paoRep = ReportCreator.GetPreview();
```

＜VB.NET の例＞

```
'プレビューオブジェクトのインスタンスを獲得
paoRep = ReportCreator.GetPreview()
```

参照

ReportCreator クラス

GetPreviewWpf メソッド

WPF 版プレビューを制御するオブジェクトを返すメソッドです。
直接印刷するときは、GetReport メソッドを使用してください。

＜C#.NET＞

IReport GetPreviewWpf()

＜VB.NET＞

Function GetPreviewWpf() As IReport

＜C#.NET の例＞

```
//WPF版プレビューオブジェクトのインスタンスを獲得
paoRep = ReportCreator.GetPreviewWpf();
```

＜VB.NET の例＞

```
'WPF版プレビューオブジェクトのインスタンスを獲得
paoRep = ReportCreator.GetPreviewWpf()
```

参照

ReportCreator クラス

GetReport メソッド

印刷を制御するオブジェクトを返すメソッドです。
プレビューを行うときは、GetPreview メソッドを使用してください。

＜C#.NET＞

IReport GetReport()

＜VB.NET＞

Function GetReport() As IReport

＜C#.NET の例＞

```
//印刷オブジェクトのインスタンスを獲得
paoRep = ReportCreator.GetReport();
```

＜VB.NET の例＞

```
'印刷オブジェクトのインスタンスを獲得
paoRep = ReportCreator.GetReport()
```

参照

ReportCreator クラス

GetPdf メソッド

PDF オブジェクトを返すメソッドです。

＜C#.NET＞

IReport GetPdf()

＜VB.NET＞

Function GetPdf() As IReport

＜C#.NET の例＞

```
// PDFオブジェクトのインスタンスを獲得
paoRep = ReportCreator.GetPdf();
```

＜VB.NET の例＞

```
' PDFオブジェクトのインスタンスを獲得
paoRep = ReportCreator.GetPdf()
```

参照

ReportCreator クラス

GetImagePdf メソッド

イメージ PDF オブジェクトを返すメソッドです。

＜C#.NET＞

IReport GetImagePdf()

＜VB.NET＞

Function GetImagePdf () As IReport

＜C#.NET の例＞

```
//イメージPDFオブジェクトのインスタンスを獲得
paoRep = ReportCreator.GetImagePdf ();
```

＜VB.NET の例＞

```
'イメージPDFオブジェクトのインスタンスを獲得
paoRep = ReportCreator.GetImagePdf ()
```

参照

ReportCreator クラス

LoadDefFile メソッド

デザインファイルを読み込みます。
プログラムがどこで動作するかわからないため、絶対パスを指定することをお勧めします。

＜C#.NET＞
void LoadDefFile(string name)
string name
　　デザインファイル名

＜VB.NET＞
Sub LoadDefFile(name As String)
name As String
　　デザインファイル名

＜C#.NET の例＞

```
//デザインファイルの読み込み
paoRep.LoadDefFile("C:¥¥デザインファイル.prepd");
```

＜VB.NET の例＞

```
'デザインファイルの読み込み
paoRep.LoadDefFile("C:¥デザインファイル.prepd")
```

参照

IReport インターフェース

ChangeDefFile メソッド

デザインファイルを変更します。
印刷データをセットし直すことなくデザインの変更を行うことができます。

＜C#.NET＞
void ChangeDefFile(string name)
string name
　　デザインファイル名

＜VB.NET＞
Sub ChangeDefFile(name As String)
name As String
　　デザインファイル名

＜C#.NET の例＞

```
//デザインファイルの読み込み
paoRep. ChangeDefFile("C:¥¥デザインファイル2.prepd");
```

＜VB.NET の例＞

```
'デザインファイルの読み込み
paoRep. ChangeDefFile("C:¥デザインファイル2.prepd")
```

参照

IReport インターフェース

PageStart メソッド

ページの開始宣言をします。
ページの開始を宣言後、ページの終了宣言(PageEnd)までの間に、印刷データをセットするコードを入れてください。

＜C#.NET＞
void PageStart()

＜VB.NET＞
Sub PageStart()

＜C#.NET の例＞

```
//頁開始を宣言
paoRep.PageStart();
        ⋮
//頁終了を宣言
paoRep.PageEnd();
```

Write() ･･･印刷データセット処理

＜VB.NET の例＞

```
'頁開始を宣言
paoRep.PageStart()
        ⋮
'頁終了を宣言
paoRep.PageEnd()
```

Write() ･･･印刷データセット処理

参照

IReport インターフェース

PageEnd メソッド

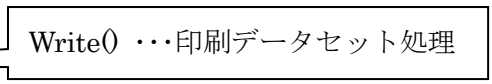
ページの終了宣言をします。
ページの開始宣言(PageStart)から、このメソッドの宣言までの間に、印刷データをセットするコードを入れてください。

＜C#.NET＞
void PageEnd()

＜VB.NET＞
Sub PageEnd()

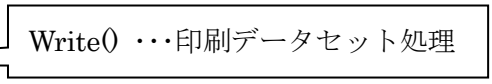
＜C#.NET の例＞

```
//頁開始を宣言
paoRep.PageStart();
        ⋮
//頁終了を宣言
paoRep.PageEnd();
```



＜VB.NET の例＞

```
'頁開始を宣言
paoRep.PageStart()
        ⋮
'頁終了を宣言
paoRep.PageEnd()
```



参照

IReport インターフェース

Write メソッド

デザインファイルで指定されたオブジェクトの操作を行います。
デザインファイルで指定されているオブジェクトに対して文字を書き込んだり、表の横罫線を繰返し描画したりします。

オーバーロードの一覧

＜C#.NET＞

void Write(string name, string value)
　オブジェクトに対して文字列をセットします。
　ヘッダやフッタなど繰り返さない固定オブジェクトの値のセットに使用してください。

void Write(string name, string value, int index)
　オブジェクトに対して描画位置を指定して文字列をセットします。
　表の行など繰り返し値をセットするオブジェクトに使用してください。

void Write(string name, int index)
　オブジェクトに対して描画位置を指定します。
　表の行の横罫線など繰り返し描画を行うオブジェクトに使用してください。

＜VB.NET＞

Sub Write(name As String, value As String)
　オブジェクトに対して文字列をセットします。
　ヘッダやフッタなど繰り返さない固定オブジェクトの値のセットに使用してください。

Sub Write(name As String, value As String, index As Long)
　オブジェクトに対して描画位置を指定して文字列をセットします。
　表の行など繰り返し値をセットするオブジェクトに使用してください。

Sub Write(name As String, index As Long)
　オブジェクトに対して描画位置を指定します。
　表の行の横罫線など繰り返し描画を行うオブジェクトに使用してください。

参照

IReport インターフェース

void Write(string name, string value) メソッド

オブジェクトに対して文字列をセットします。
ヘッダやフッタなど繰り返さない固定オブジェクトの値のセットに使用してください。

string name

デザインファイル内のオブジェクト名を指定します。
この場合、文字列をセットするため対象となるオブジェクトのタイプは、基本的にText(文字列）と、ArtText(装飾文字列）と、Barcode(バーコード) のみとなります。
オブジェクトを削除したいときに、Text(文字列)・ArtText(装飾文字列)以外のオブジェクトを指定します。

string value

セットする文字列を指定します。
Text(文字列）と、ArtText(装飾文字列)以外のオブジェクトに空文字(“”)を指定した場合、そのオブジェクトを削除します。

※ バーコードオブジェクトにバイナリデータを入力する方法

Code128 等、改行や TAB といったバイナリコードを入力する場合…

```
value += "¥n";
```

これに加えて、

```
value += ((char)10).ToString();
value += ((char)0x0a).ToString();
```

のような指定が可能です。

※ バーコード : GS1-128(UCC/EAN128)の特記事項

Barcode(バーコード)オブジェクトへの文字列の設定方法で、少し特殊なGS1-128(UCC/EAN128)において、AI(アプリケーション識別子)挿入方法は 2 通りございます。

(1) 可変長項目(データブロック)の後の AI には、FNC1 を挿入
⇒これまで通り"{FNC1}"を付ける。例 : "{FNC1}21"のようにコードを指定

(2) 固定長項目(データブロック)の後の AI には、固定長のため目印の FNC1 は不要
⇒新しく追加した"{AI}"を付ける。例 : "{AI}21" のようにコードを指定

"{AI}"を指定して FNC1 を挿入しない場合も、カッコ()付コード文字は出力されます。例えば入力コードに"{AI}21"を指定した場合、添え字には(21)と出力されます。

例) (01)04512345670016(21)1
コード指定方法 → ”{FNC1}0104512345670016{AI}211”

＜例＞

```
//文字列のセット
paoRep.Write("日付", System.DateTime.Now.ToString());
```

参照

IReport インターフェース

void Write(string name, string value, int index) メソッド

オブジェクトに対して描画位置を指定して文字列をセットします。
表の行など繰り返し値をセットするオブジェクトに使用してください。
このパターンのメソッドを使用する場合、デザインファイル内の IntervalX 又は IntervalY に１以上の値が入っている必要があります。
IntervalX とは、横方向に繰り返す間隔(mm)です。
IntervalY とは、縦方向に繰り返す間隔(mm)です。主に表の行などに使用されます。

string name

デザインファイル内のオブジェクト名を指定します。
この場合、文字列をセットするため対象となるオブジェクトのタイプは、基本的に Text(文字列) と、ArtText(装飾文字列)と、Barcode(バーコード) のみとなります。
オブジェクトを削除したいときに、Text(文字列)・ArtText(装飾文字列)以外のオブジェクトを指定します。

string value

セットする文字列を指定します。
Text(文字列) と、ArtText(装飾文字列)以外のオブジェクトに空文字(“”)を指定した場合、そのオブジェクトを削除します。

int index

IntervalX／IntervalY で指定された縦方向・横方向の間隔で描画を行うページ内の描画位置です。左上から右下方向に値が大きくなります。
例えば、表で IntervalY に値がある場合、描画位置は・・・
オブジェクトの最初の位置＋InterbalY×(index –1) のようになります。
表の場合、１行目が１、２行目が２、３行目が３となります。

＜例＞

```
//繰返し文字列のセット
paoRep.Write("No.", “1” , 1);
```

参照

IReport インターフェース

void Write(string name, int index) メソッド

オブジェクトに対して描画位置を指定します。
表の行の横罫線など繰り返し描画を行うオブジェクトに使用してください。
このパターンのメソッドを使用する場合、デザインファイル内の IntervalX 又は IntervalY に１以上の値が入っている必要があります。
IntervalX とは、横方向に繰り返す間隔(mm)です。
IntervalY とは、縦方向に繰り返す間隔(mm)です。主に表の行などに使用されます。

string name

デザインファイル内のオブジェクト名を指定します。
この場合、どのオブジェクトでも繰返し描画ができるため対象となるオブジェクトのタイプは、全てのオブジェクトです。

int index

IntervalX／IntervalY で指定された縦方向・横方向の間隔で描画を行うページ内の印字位置です。左上から右下方向に値が大きくなります。
例えば、表で IntervalY に値がある場合、印字位置は・・・
オブジェクトの最初の位置＋InterbalY×(index –1) のようになります。
表の場合、１行目が１、２行目が２、３行目が３となります。

＜例＞

```
//繰返し文字列のセット
paoRep.Write("横線", 1);
```

参照

IReport インターフェース

Sub Write(name As String, value As String) メソッド

オブジェクトに対して文字列をセットします。
ヘッダやフッタなど繰り返さない固定オブジェクトの値のセットに使用してください。

name As String

デザインファイル内のオブジェクト名を指定します。
この場合、文字列をセットするため対象となるオブジェクトのタイプは、基本的にText(文字列) と、ArtText(装飾文字列) と、Barcode(バーコード) のみとなります。
オブジェクトを削除したいときに、Text(文字列)・ArtText(装飾文字列)以外のオブジェクトを指定します。

value As String

セットする文字列を指定します。
Text(文字列) と、ArtText(装飾文字列)以外のオブジェクトに空文字(“”)を指定した場合、そのオブジェクトを削除します。

＜例＞

```
'文字列のセット
paoRep.Write("日付", System.DateTime.Now.ToString())
```

参照

IReport インターフェース

Sub Write(name As String, value As String, index As Long) メソッド

オブジェクトに対して描画位置を指定して文字列をセットします。
表の行など繰り返し値をセットするオブジェクトに使用してください。
このパターンのメソッドを使用する場合、デザインファイル内の IntervalX 又は IntervalY に１以上の値が入っている必要があります。
IntervalX とは、横方向に繰り返す間隔(mm)です。
IntervalY とは、縦方向に繰り返す間隔(mm)です。主に表の行などに使用されます。

name As String

デザインファイル内のオブジェクト名を指定します。
この場合、文字列をセットするため対象となるオブジェクトのタイプは、基本的に Text(文字列) と、ArtText(装飾文字列)と、Barcode(バーコード) のみとなります。
オブジェクトを削除したいときに、Text(文字列)・ArtText(装飾文字列)以外のオブジェクトを指定します。

value As String

セットする文字列を指定します。
Text(文字列) と、ArtText(装飾文字列)以外のオブジェクトに空文字(“”)を指定した場合、そのオブジェクトを削除します。

index As Long

IntervalX／IntervalY で指定された縦方向・横方向の間隔で描画を行うページ内の描画位置です。左上から右下方向に値が大きくなります。
例えば、表で IntervalY に値がある場合、描画位置は・・・
オブジェクトの最初の位置＋InterbalY×(index –1) のようになります。
表の場合、１行目が１、２行目が２、３行目が３となります。

＜例＞

```
paoRep.Write("No.", “1”, 1)
```

参照

IReport インターフェース

Sub Write(name As String, index As Long) メソッド

オブジェクトに対して描画位置を指定します。
表の行の横罫線など繰り返し描画を行うオブジェクトに使用してください。
このパターンのメソッドを使用する場合、デザインファイル内の IntervalX 又は IntervalY に１以上の値が入っている必要があります。
IntervalX とは、横方向に繰り返す間隔(mm)です。
IntervalY とは、縦方向に繰り返す間隔(mm)です。主に表の行などに使用されます。

name As String

デザインファイル内のオブジェクト名を指定します。
この場合、どのオブジェクトでも繰返し描画ができるため対象となるオブジェクトのタイプは、全てのオブジェクトです。

index As Long

IntervalX／IntervalY で指定された縦方向・横方向の間隔で描画を行うページ内の印字位置です。左上から右下方向に値が大きくなります。
例えば、表で IntervalY に値がある場合、印字位置は・・・
オブジェクトの最初の位置＋InterbalY×(index –1) のようになります。
表の場合、１行目が１、２行目が２、３行目が３となります。

＜例＞

```
paoRep.Write("横線", 1)
```

参照

IReport インターフェース

Output メソッド

レポート(帳票)のプリンターへの印刷、又は、プレビュー画面の表示を行います。

オーバーロードの一覧

＜C#.NET＞

bool Output()

デフォルトのプリンタへデフォルトの設定で印刷／プレビュー指示を行います。

bool Output(System.Drawing.Printing.PrinterSettings setting)

引数で指定したプリンタの設定で印刷／プレビュー指示を行います。

＜VB.NET＞

Function Output() As Boolean

デフォルトのプリンタへデフォルトの設定で印刷／プレビュー指示を行います。

Function Output(setting As System.Drawing.Printing.PrinterSettings) As Boolean

引数で指定したプリンタの設定で印刷／プレビュー指示を行います。

参照

IReport インターフェース

Output() メソッド

デフォルトのプリンタへデフォルトの設定で印刷／プレビュー指示を行います。

＜C#.NET＞
bool Output()

＜VB.NET＞
Function Output() As Boolean

＜C#.NET の例＞

```
paoRep.Output(); // 印刷/プレビューを実行
```

＜VB.NET の例＞

```
paoRep.Output() '印刷/プレビューを実行
```

参照

IReport インターフェース
Output メソッド

Output(System.Drawing.Printing.PrinterSettings setting) メソッド

引数で指定したプリンタの設定で印刷／プレビュー指示を行います。

＜C#.NET＞

bool Output(System.Drawing.Printing.PrinterSettings setting)

＜VB.NET＞

Function Output(setting As System.Drawing.Printing.PrinterSettings) As Boolean

＜C#.NET の例＞

```
System.Drawing.Printing.PrinterSettings setting
    = new System.Drawing.Printing.PrinterSettings();
setting.PrinterName = “プリンター名”;
paoRep.Output(setting); // 印刷/プレビューを実行
```

＜VB.NET の例＞

```
Dim setting As System.Drawing.Printing. PrinterSettings
    = New System.Drawing.Printing. PrinterSettings ()
setting.PrinterName = “プリンター名”
paoRep.Output(setting) '印刷/プレビューを実行
```

参照

IReport インターフェース

Output メソッド

GetPrintDocument メソッド

印刷データがセットされた PrintDocument を取得します。

＜C#.NET＞

bool Output()

＜VB.NET＞

Function Output() As Boolean

＜C#.NET の例＞

```
IReport paoRep = ReportCreator.GetReport();  // or GetPreview()
paoRep.LoadDefFile("./デザイン.prepd");
paoRep.PageStart();
paoRep.Write("項目１","あああ");
paoRep.Write("項目２","いいい");
paoRep.Write("項目３","ううう");
paoRep.PageEnd();

// PrintDocument取得
printDocument1 = paoRep.GetPrintDocument();

// このフォームのプレビューコントロールへ プレビュー実行
printPreviewControl1.Document = printDocument1;
printPreviewControl1.InvalidatePreview();
```

＜VB.NET の例＞

```
Dim paoRep As IReport = ReportCreator.GetReport() // or GetPreview()
paoRep.LoadDefFile("./デザイン.prepd ")
paoRep.PageStart()
paoRep.Write("項目１","あああ")
paoRep.Write("項目２","いいい")
paoRep.Write("項目３","ううう")
paoRep.PageEnd()

// PrintDocument取得
printDocument1 = paoRep.GetPrintDocument()

// このフォームのプレビューコントロールへ プレビュー実行
printPreviewControl1.Document = printDocument1
printPreviewControl1.InvalidatePreview()
```

参照

IReport インターフェース

SaveXMLFile メソッド

印刷データを XML ファイルに保存します。
保存した印刷データは、プログラム(LoadXMLFile)、又は、プレビュー画面から読み込むことが可能です。

＜C#.NET＞
bool SaveXMLFIle(string name)
string name
　　保存する印刷データ XML ファイルパス名

＜VB.NET＞
SaveXMLFIle(name As String) As Boolean
name As String
　　保存する印刷データ XML ファイルパス名

＜C#.NET の例＞

```
paoRep.SaveXMLFile("印刷データ.XML"); //印刷データの保存
```

＜VB.NET の例＞

```
paoRep.SaveXMLFile("印刷データファイル.xml") '印刷データの保存
```

参照

IReport インターフェース

LoadXMLFile メソッド

SaveXMLFile で保存された、印刷データ XML ファイルを読み込みます。
読み込んだ印刷データは、印刷又はプレビュー(Output)することが可能です。

＜C#.NET＞
bool LoadXMLFIle(string name)
string name
　　読み込む印刷データ XML ファイルパス名

＜VB.NET＞
LoadXMLFIle(name As String) As Boolean
name As String
　　読み込む印刷データ XML ファイルパス名

＜C#.NET の例＞

```
paoRep.LoadXMLFile("印刷データ.XML"); //印刷データの読込
```

＜VB.NET の例＞

```
paoRep.LoadXMLFile("印刷データファイル.xml") '印刷データの読込
```

参照
IReport インターフェース

SaveData メソッド

圧縮した印刷バイナリデータを返します。
WEB サービス側で、リッチクライアントに返す印刷データを作成する時に使用します。

＜C#.NET＞
byte[] SaveData()

＜VB.NET＞
SaveData() As Byte()

＜C#.NET の例＞

```
byte[] b = paoRep. SaveData(); // 圧縮した印刷バイナリデータを返す
```

＜VB.NET の例＞

```
Dim b As Byte() = paoRep. SaveData ' 圧縮した印刷バイナリデータを返す
```

参照

IReport インターフェース

LoadData メソッド

SaveData で作成された圧縮した印刷バイナリデータを読み込みます。
リッチクライアントで、WEB サービス側で作成された印刷データを読み込む時に使用します。

＜C#.NET＞
bool LoadData(string name)
string name
　　読み込む印刷データ XML ファイルパス名

＜VB.NET＞
LoadData(name As String) As Boolean
name As String
　　読み込む印刷データ XML ファイルパス名

＜C#.NET の例＞

```
byte[] data = webService.getPrintData();
IReport paoRep = ReportCreator.GetPreview()    // プレビューオブジェクトを作成
paoRep.LoadData(data); // 圧縮した印刷バイナリデータの読込
paoRep.Output(); //プレビュー
```

＜VB.NET の例＞

```
Dim data As Byte() = webTest.get帳票データ() '印刷データを取得
Dim paoRep As IReport = ReportCreator.GetPreview() ' プレビューオブジェクトを作成
paoRep.LoadData(data) '印刷データを読み込む
paoRep.Output() 'プレビューを実行
```

参照

IReport インターフェース

SaveSVGFile メソッド

SVG 形式の印刷データを書き出します。

＜C#.NET＞
bool SaveSVGFile(string name)
string name
　　書き出す印刷データ html ファイルパス名
　　SVG ファイルは、ページ数分作成されるため、
　　それを読み込む html ファイルの名前を指定します。

＜VB.NET＞
SaveSVGFile(name As String) As Boolean
name As String
　　書き出す印刷データ SVG ファイルパス名
　　SVG ファイルは、ページ数分作成されるため、
　　それを読み込む html ファイルの名前を指定します。

＜C#.NET の例＞

```
paoRep.SaveSVGFile("印刷データ.html"); //SVGデータの書出
```

＜VB.NET の例＞

```
paoRep. SaveSVGFile("印刷データファイル.html") 'SVGデータの書出
```

参照

IReport インターフェース

SaveSVGZFile メソッド

SVGZ 形式の印刷データを書き出します。

＜C#.NET＞
bool SaveSVGZFile(string name)
string name
　　書き出す印刷データ html ファイルパス名
　　SVGZ ファイルは、ページ数分作成されるため、
　　それを読み込む html ファイルの名前を指定します。

＜VB.NET＞
SaveSVGZFile(name As String) As Boolean
name As String
　　書き出す印刷データ SVGZ ファイルパス名
　　SVGZ ファイルは、ページ数分作成されるため、
　　それを読み込む html ファイルの名前を指定します。

＜C#.NET の例＞

```
paoRep.SaveSVGZFile("印刷データ.html"); //SVGZデータの書出
```

＜VB.NET の例＞

```
paoRep.SaveSVGZFile("印刷データファイル.html") 'SVGZデータの書出
```

参照

IReport インターフェース

SavePDF メソッド (Stream)

PDF 形式の印刷データを書き出します。(Stream)

＜C#.NET＞
bool SavePDF (System.IO.Stream stream)
System.IO.Stream stream
　　書き出す印刷データ PDF　の Stream

＜VB.NET＞
SavePDF (name As System.IO.Stream) As Boolean
name As System.IO.Stream
　　書き出す印刷データ PDF　の Stream

＜C#.NET の例＞

```
paoRep.SavePDF(anyStream); //PDFデータの書出
```

＜VB.NET の例＞

```
paoRep.SavePDF(anyStream) 'PDFデータの書出
```

参照

IReport インターフェース

SavePDF メソッド (ファイル)

PDF 形式の印刷データを書き出します。(ファイル)

＜C#.NET＞
bool SavePDF (string name)
string name
　　書き出す印刷データ PDF ファイルパス名

＜VB.NET＞
SavePDF (name As String) As Boolean
name As String
　　書き出す印刷データ PDF ファイルパス名

＜C#.NET の例＞

```
paoRep.SavePDF("印刷データ.PDF"); //PDFデータの書出
```

＜VB.NET の例＞

```
paoRep.SavePDF("印刷データファイル.pdf") 'PDFデータの書出
```

参照

IReport インターフェース

SaveXPS メソッド

XPS 形式の印刷データを書き出します。(ファイル)
XPS とは、Microsoft 版 PDF のようなものです。
環境によって異なりますが、大抵の Windows には、デフォルトで 1 つだけ「Document Writer」というプリンタが入っていると思います。
XPS は「Document Writer」の出力結果でもあります。
XPS ファイルは WPF アプリケーションで簡単に印刷プレビューを行うことができます。

SaveXPS メソッドを使用して、XPS ファイル出力にすることができます。
※ただしプリンタの一覧に、「Document Writer」が存在することが前提です。

＜C#.NET＞
bool SaveXPS (string name)
string name
　　書き出す印刷データ XPS ファイルパス名

＜VB.NET＞
SaveXPS (name As String) As Boolean
name As String
　　書き出す印刷データ XPS ファイルパス名

＜C#.NET の例＞
```
IReports paoRep = ReportCreator.GetPreview(); // ReportCreator.GetReport()でもＯＫ
 :
 :
paoRep.SaveXPS("印刷データ.xps"); //XPSデータの書出
```

＜VB.NET の例＞
```
Dim paoRep As IReport = ReportCreator.GetPreview() 'ReportCreator.GetReport()でもＯＫ
 :
 :
paoRep.SaveXPS("印刷データファイル.xps") 'XPSデータの書出
```

参照

IReport インターフェース

AllPage プロパティ

印刷・プレビューを行うドキュメントの全ページ数を取得できます。

＜C#.NET＞

int AllPage

　全ページ数

＜VB.NET＞

AllPage As Integer

　全ページ数

＜C#.NET の例＞

```
MessageBox.Show(paoRep.AllPage.ToString());  // メッセージボックスで全ページ数を表示
```

＜VB.NET の例＞

```
MessageBox.Show(paoRep.AllPage.ToString()  ‘メッセージボックスで全ページ数を表示
```

参照

IReport インターフェース

AccessFile プロパティ

Output メソッドで印刷を行うときに、プレビュー画面からファイルアクセス（ファイル保存等）を許可するかどうかを指定します。デフォルトは、true: 表示します。

＜C#.NET＞

bool AccessFile

true：印刷時にプレビュー画面からファイルアクセスを許可する(既定値)

false：印刷時にプレビュー画面からファイルアクセスを許可しない

＜VB.NET＞

AccessFile As Boolean

True：印刷時にプレビュー画面からファイルアクセスを許可する(既定値)

False：印刷時にプレビュー画面からファイルアクセスを許可しない

＜C#.NET の例＞

```
paoRep.AccessFile = false;  //プレビュー画面からファイルアクセスを許可しない
```

＜VB.NET の例＞

```
paoRep. AccessFile = False  'プレビュー画面からファイルアクセスを許可しない
```

参照

IReport インターフェース

AcceptDragDrop プロパティ

Output メソッドで印刷を行うときに、プレビュー画面にファイルのドラッグ＆ドロップを許可するかどうかを指定します。
ドラッグ＆ドロップの対象ファイルは、印刷データファイルや帳票定義ファイル(デザインファイル)です。
印刷データファイルをドラッグ＆ドロップすることで、ドロップしたファイルの印刷プレビューを行う機能がございます。ただし、ドラッグ＆ドロップを許可すると、プレビュー画面をマルチスレッドで立ち上げることができません。
AcceptDragDrop の既定値(デフォルト)は、以下の通りです。

- プログラムからプレビューする場合、このプロパティは (デフォルト)false なので、プログラムから指定された印刷ドキュメント以外のファイルをドラッグ＆ドロップしてプレビューすることができません。
  ただし、プレビューを行う前に、AcceptDragDrop に true をセットすれば、ドラッグ＆ドロップでのプレビューは可能となります。
- プレビューアの exe を単独起動した時、このプロパティは true にしているため、印刷データやデザインファイルのドラッグ＆ドロップを行ってプレビューすることができます。

＜C#.NET＞
bool AcceptDragDrop
　true：印刷時にプレビュー画面にファイルのドラッグ＆ドロップを許可する
　false：印刷時にプレビュー画面にファイルのドラッグ＆ドロップを許可しない(既定値)

＜VB.NET＞
AcceptDragDrop As Boolean
　True：印刷時にプレビュー画面にファイルのドラッグ＆ドロップを許可する
　False：印刷時にプレビュー画面にファイルのドラッグ＆ドロップを許可しない(既定値)

＜C#.NET の例＞

```
paoRep.AcceptDragDrop = false;  //プレビュー画面にドラッグ＆ドロップを許可しない
```

＜VB.NET の例＞

```
paoRep. AcceptDragDrop = False  ‘プレビュー画面にドラッグ＆ドロップを許可しない
```

参照
IReport インターフェース

CutByPage プロパティ

印刷時、1 ページずつ用紙をカットするか指定できます。
シール・ラベルプリンタをご利用のお客様からリクエストがあり実装したプロパティです。

＜C#.NET＞
bool CutByPage
　true：1 ページずつ用紙カットをする
　false： 全ページ出力後用紙カットをする(既定値)

＜VB.NET＞
CutByPage As Boolean
　True：1 ページずつ用紙カットをする
　False： 全ページ出力後用紙カットをする(既定値)

＜C#.NET の例＞
```
paoRep. CutByPage = true;  //1ページずつ用紙カットをする
```

＜VB.NET の例＞
```
paoRep. CutByPage = True  ‘1ページずつ用紙カットをする
```

参照

IReport インターフェース

DisplayDialog プロパティ

Output メソッドで印刷を行うときに、[印刷]ダイアログボックスを表示するかどうかを指定します。デフォルトは、true: 表示します。このプロパティは、印刷時のみ有効で、プレビュー表示時に指定されても意味を持ちません。

＜C#.NET＞

bool DisplayDialog

true：印刷時に[印刷]ダイアログボックスを表示(既定値)

false：印刷時に[印刷]ダイアログボックスを表示しない

＜VB.NET＞

DisplayDialog As Boolean

True：印刷時に[印刷]ダイアログボックスを表示(既定値)

False：印刷時に[印刷]ダイアログボックスを表示しない

＜C#.NET の例＞

```
paoRep.DisplayDialog = false;  //[印刷]ダイアログを表示しない
paoRep.Output(); //印刷
```

＜VB.NET の例＞

```
paoRep.DisplayDialog = False  '[印刷]ダイアログを表示しない
paoRep.Output() '印刷
```

参照

IReport インターフェース

DisplayPrinting プロパティ

Output メソッドで印刷を行うときに、印刷中(ページ数)を表示するかどうかを指定します。デフォルトは、true: 表示します。このプロパティは、印刷時のみ有効で、プレビュー表示時に指定されても意味を持ちません。

＜C#.NET＞

bool DisplayPrinting

true： 印刷時に印刷中(ページ数)を表示(既定値)

false： 印刷時に印刷中(ページ数)を表示しない

＜VB.NET＞

DisplayPrinting As Boolean

True： 印刷時に印刷中(ページ数)を表示(既定値)

False： 印刷時に印刷中(ページ数)を表示しない

＜C#.NET の例＞

```
paoRep.DisplayPrinting = false;  //印刷中(ページ数)を表示しない
paoRep.Output(); //印刷
```

＜VB.NET の例＞

```
paoRep.DisplayPrinting  = False  '印刷中(ページ数)を表示しない
paoRep.Output() '印刷
```

参照

IReport インターフェース

MarginTop プロパティ

Output メソッドで印刷または、プレビューを行う時の上部余白を mm 単位で指定します。
印刷または、プレビューのみに有効です。
プリンタによって出力結果が違う場合などの微調整に使うことができます。

＜C#.NET＞
float MarginTop

＜VB.NET＞
MarginTop As float

＜C#.NET の例＞

```
paoRep. MarginTop = 10;  // 上部余白を1cm に指定
```

＜VB.NET の例＞

```
paoRep. MarginTop = 10  ‘上部余白を1cm に指定
```

参照

IReport インターフェース

MarginLeft プロパティ

Output メソッドで印刷または、プレビューを行う時の左側余白を mm 単位で指定します。
印刷または、プレビューのみに有効です。
プリンタによって出力結果が違う場合などの微調整に使うことができます。

＜C#.NET＞
float MarginLeft

＜VB.NET＞
MarginLeft As float

＜C#.NET の例＞

```
paoRep. MarginLeft = 10;  // 左側余白を1cm に指定
```

＜VB.NET の例＞

```
paoRep. MarginLeft = 10  '左側余白を1cm に指定
```

参照

IReport インターフェース

PreviewDialog プロパティ

Output メソッドで印刷を行うときに、プレビュー画面をダイアログ表示するかどうかを取得・設定します。デフォルトは、true(ダイアログ表示)です。このプロパティを false にした場合、通常のフォームとしてプレビュー画面を起動するため、複数のプレビュー画面を同時に起動することができます。モーダレスフォームのことです。

＜C#.NET＞
bool PreviewDialog
　true：プレビュー時にダイアログ画面(モーダルフォーム)を起動する。
　false：プレビュー時に通常フォーム(モーダレスフォーム)を起動する。

＜VB.NET＞
PrviewDialog As Boolean
　True：プレビュー時にダイアログ画面(モーダルフォーム)を起動する。
　False：プレビュー時に通常フォーム(モーダレスフォーム)を起動する。

＜C#.NET の例＞

```
paoRep. PreviewDialog = false;  //プレビュー画面を同時に複数起動させたいい場合
```

＜VB.NET の例＞

```
paoRep. PreviewDialog = False  ‘プレビュー画面を同時に複数起動させたいい場合
```

参照
IReport インターフェース

PreviewInTaskbar プロパティ

Output メソッドで印刷を行うときに、プレビュー表示時、タスクバーに表示するかどうかを設定します。デフォルトは、false(タスクバーに入れない)です。このプロパティを true にした場合、プレビューウィンドウ(タスク)がタスクバーに表示されます。
次の制限がございます。
(1) 複数起動の場合、重なって 1 か所に表示されます。
(2) プレビューアイコンを独自で指定していた場合でも Reports.net 既定のプレビューアイコンタスクバー上に表示されます。1 タスクとして表示するため(アイコンは EXE で一つのため)、独自指定のアイコンを別々に表示できないためです。
(3) タスクバーに表示する条件としては、ダイアログ表示でなく、モーダルで表示する必要があります。

＜C#.NET＞
bool PreviewInTaskbar
　true：プレビューをタスクバーに表示する。
　false：プレビューをタスクバーに表示しない。

＜VB.NET＞
PreviewInTaskbar As Boolean
　True：プレビューをタスクバーに表示する。
　False：プレビューをタスクバーに表示しない。

＜C#.NET の例＞

```
paoRep.PreviewDialog = false;  //プレビュー画面を同時に複数起動可能
paoRep. PreviewInTaskbar = true;  //プレビューをタスクバーに入れる
```

＜VB.NET の例＞

```
paoRep. PreviewDialog = False  ‘プレビュー画面を同時に複数起動可能
paoRep. PreviewInTaskbar = True  ‘プレビューをタスクバーに入れる
```

参照

IReport インターフェース

SwapPdfImage プロパティ

PDF 出力中に画像データをディスクにスワップするかどうかを指定できます。
元々無いプロパティでしたが、大きな画像をで多いページを PDF 出力したときにメモリ不足(Out Of Memory)が発生する不具合がございました。そのために加えられたプロパティです。 既定値：false

＜C#.NET＞
bool SwapPdfImage
　true：PDF 出力中に画像データをディスクにスワップする。
　false：PDF 出力中に画像データは、プログラム内メモリで処理する。

＜VB.NET＞
SwapPdfImage　As Boolean
　True：PDF 出力中に画像データをディスクにスワップする。
　False：PDF 出力中に画像データは、プログラム内メモリで処理する。

＜C#.NET の例＞

```
paoRep. SwapPdfImage　= true;　// PDF出力中画像データをスワップする
```

＜VB.NET の例＞

```
paoRep. SwapPdfImage　= True　'PDF出力中画像データをスワップする
```

参照

IReport インターフェース

z_Objects プロパティ ／ IObjects インターフェース

デザイン時の各オブジェクトの各プロパティの値を実行時に、設定・取得する時に使います。

このプロパティを使用して、オブジェクトの色や、位置、フォントなどのプロパティを実行時に変更できます。

SetObject() に**行番号**を指定して、特定行のオブジェクト属性のみ変更することも可能です。

パブリックメソッド

SetObject SetObject (行指定)	オブジェクト名を指定してプロパティを編集するオブジェクト設定 (プロパティを変更する繰り返しオブジェクトの行番号を指定することも可能)

パブリックプロパティ

z_Text	文字列オブジェクト用プロパティ
z_Line	罫線 xxx オブジェクト用プロパティ
z_Square	四角形オブジェクト用プロパティ
z_Circle	円オブジェクト用プロパティ
z_Image	画像オブジェクト用プロパティ
z_Barcode	バーコードオブジェクト用プロパティ
z_ArtText	装飾文字オブジェクト用プロパティ

＜C#.NET の例＞

```
// 文字列オブジェクトの文字位置・フォントサイズ・太字を変更
paoRep.z_Objects.SetObject("文字列");
paoRep.z_Objects.z_Text.TextAlign = PmAlignType.Right;
paoRep.z_Objects.z_Text.z_FontAttr.Size = 8;
paoRep.z_Objects.z_Text.z_FontAttr.Bold = true;
```

＜VB.NET の例＞

```
'文字列オブジェクトの文字位置・フォントサイズ・太字を変更
paoRep.z_Objects.SetObject("文字列")
paoRep.z_Objects.z_Text.TextAlign = PmAlignType.Right
paoRep.z_Objects.z_Text.z_FontAttr.Size = 8
paoRep.z_Objects.z_Text.z_FontAttr.Bold = True
```

SetObject (string　objName) メソッド

これから、どのオブジェクトのプロパティの取得・設定を行うかを指定します。
引数には、デザイン時のオブジェクト名を指定してください。
このメソッドを呼び出した後に、引数で指定したオブジェクトのプロパティの値を取得・設定することができます。

＜C#.NET＞
bool SetObject(string objName)

＜VB.NET＞
Function SetObject(String objName) As Boolean

＜C#.NET の例＞

```
paoRep.z_Objects.SetObject("オブジェ名"); // プロパティを編集するオブジェクトの指定
```

＜VB.NET の例＞

```
paoRep.z_Objects.SetObject("オブジェ名") ' プロパティを編集するオブジェクトの指定
```

参照
IObjects インターフェース／z_Objects プロパティ

SetObject (string　objName, int　lineNo) メソッド

これから、どのオブジェクトの何行目のプロパティの取得・設定を行うかを指定します。
引数には、デザイン時のオブジェクト名と、行番号を指定してください。
このメソッドを呼び出した後に、引数で指定したオブジェクトとその行のプロパティの値を取得・設定することができます。

＜C#.NET＞

```
bool SetObject(string objName, int lineNo)
```

＜VB.NET＞

```
Function SetObject(objName As String, lineNo As Integer) As Boolean
```

＜C#.NET の例＞

```
// これから、指定オブジェクトの3行目のプロパティを設定(編集)します。という宣言。
paoRep.z_Objects.SetObject("オブジェ名", 3);
```

＜VB.NET の例＞

```
‘ これから、指定オブジェクトの3行目のプロパティを設定(編集)します。という宣言。
paoRep.z_Objects.SetObject("オブジェ名", 3)
```

参照

IObjects インターフェース／z_Objects プロパティ

z_Text プロパティ ／ ZText クラス

z_Objects の一つ下の階層のクラスオブジェクトです。文字列(テキスト)オブジェクトの各プロパティの値を取得・設定する時に、この z_Text の下のプロパティの値を取得・設定します。

Z_Text 配下の以下のプロパティの値の取得・設定が可能です。

型(C#)	型(VB.NET)	プロパティ名	説明
float	Single	Angle	回転角度
System.Drawing.Color		BackColor	背景色(文字列・画像用)
float	Single	Height	描画範囲(高さ)
float	Single	IntervalX	描画間隔(X 座標に対して)
float	Single	IntervalY	描画間隔(Y 座標に対して)
Bool	Boolean	IsElastic	伸縮自在
System.Drawing.Color		OutLineColor	アウトライン色
float	Single	OutLineWidth	アウトライン幅
Int	Integer	Repeat	繰り返し回数
String	String	Text	表示文字列
Pao.Reports.PmAlignType		TextAlign	表示位置
float	Single	Width	描画範囲(幅)
float	Single	X	始点(左上)の X 座標
float	Single	Y	始点(左上)の Y 座標
Pao.Reports.ZFontAttr		z_FontAttr	フォント属性

参照

IObjects インターフェース／z_Objects プロパティ

ZFontAttr クラス／z_FontAttr プロパティ

z_Line プロパティ／ ZLine クラス

z_Objects の一つ下の階層のクラスオブジェクトです。罫線オブジェクトの各プロパティの値を取得・設定する時に、この z_Line の下のプロパティの値を取得・設定します。

Z_Line 配下の以下のプロパティの値の取得・設定が可能です。

型(C#)	型(VB.NET)	プロパティ名	説明
float	Single	EndX	罫線の終点の X 座標
float	Single	EndY	罫線の終点の Y 座標
float	Single	IntervalX	描画間隔(X 座標に対して)
float	Single	IntervalY	描画間隔(Y 座標に対して)
int	Integer	Repeat	繰り返し回数
float	Single	Thick	罫線の円弧の厚み
float	Single	X	始点(左上)の X 座標
float	Single	Y	始点(左上)の Y 座標
Pao.Reports.ZLineAttr		z_LineAttr	罫線属性

参照

IObjects インターフェース／z_Objects プロパティ

ZLineAttr クラス／z_LineAttr プロパティ

z_Square プロパティ／ ZSquare クラス

z_Objects の一つ下の階層のクラスオブジェクトです。四角形オブジェクトの各プロパティの値を取得・設定する時に、この z_Square の下のプロパティの値を取得・設定します。

Z_Square 配下の以下のプロパティの値の取得・設定が可能です。

型(C#)	型(VB.NET)	プロパティ名	説明
float	Single	Angle	回転角度
Pao.Reports.ZCornerType		CornerType	四角形の角の状態
Int	Integer	HatchDensity	網掛け濃度(%)
float	Single	Height	描画範囲(高さ)
float	Single	IntervalX	描画間隔(X 座標に対して)
float	Single	IntervalY	描画間隔(Y 座標に対して)
System.Drawing.Color		PaintColor	塗り潰す色
float	Single	R	四角の角の丸さを表す値
Int	Integer	Repeat	繰り返し回数
float	Single	Width	描画範囲(幅)
float	Single	X	始点(左上)の X 座標
float	Single	Y	始点(左上)の Y 座標
Pao.Reports.ZLineAttr		z_LineAttr	罫線属性

参照

IObjects インターフェース／z_Objects プロパティ

ZLineAttr クラス／z_LineAttr プロパティ

z_Circle プロパティ／ ZCircle クラス

z_Objects の一つ下の階層のクラスオブジェクトです。円オブジェクトの各プロパティの値を取得・設定する時に、この z_Circle の下のプロパティの値を取得・設定します。

Z_Circle 配下の以下のプロパティの値の取得・設定が可能です。

型(C#)	型(VB.NET)	プロパティ名	説明
float	Single	Angle	回転角度
Int	Integer	HatchDensity	網掛け濃度(%)
float	Single	Height	描画範囲(高さ)
float	Single	IntervalX	描画間隔(X 座標に対して)
float	Single	IntervalY	描画間隔(Y 座標に対して)
System.Drawing.Color		PaintColor	塗り潰す色
Int	Integer	Repeat	繰り返し回数
float	Single	Width	描画範囲(幅)
float	Single	X	始点(左上)の X 座標
float	Single	Y	始点(左上)の Y 座標
Pao.Reports.ZLineAttr		z_LineAttr	罫線属性

参照

IObjects インターフェース／z_Objects プロパティ

ZLineAttr クラス／z_LineAttr プロパティ

z_Image プロパティ／ ZImage クラス

z_Objects の一つ下の階層のクラスオブジェクトです。画像オブジェクトの各プロパティの値を取得・設定する時に、この z_Image の下のプロパティの値を取得・設定します。

Z_Image 配下の以下のプロパティの値の取得・設定が可能です。

型(C#)	型(VB.NET)	プロパティ名	説明
float	Single	Angle	回転角度
System.Drawing.Color		BackColor	背景色(文字列・画像用)
float	Single	Height	描画範囲(高さ)
Pao.Reports.PmImgAlignType		ImageAlign	イメージ位置
String	String	ImageData	イメージファイルのパス又はデータ
Pao.Reports.PmImgRevType		ImageRev	イメージ反転
float	Single	IntervalX	描画間隔(X 座標に対して)
float	Single	IntervalY	描画間隔(Y 座標に対して)
Int	Integer	Repeat	繰り返し回数
float	Single	Width	描画範囲(幅)
float	Single	X	始点(左上)の X 座標
float	Single	Y	始点(左上)の Y 座標
Pao.Reports.ZLineAttr		z_LineAttr	罫線属性

参照

IObjects インターフェース／z_Objects プロパティ

ZLineAttr クラス／z_LineAttr プロパティ

z_Barcode プロパティ／ ZBarcode クラス

z_Objects の一つ下の階層のクラスオブジェクトです。画像オブジェクトの各プロパティの値を取得・設定する時に、この z_Barcode の下のプロパティの値を取得・設定します。

Z_Barcode 配下の以下のプロパティの値の取得・設定が可能です。

型(C#)	型(VB.NET)	プロパティ名	説明
float	Single	Angle	回転角度
Bool	Boolean	DispStartStop	スタート･ストップコードを表示するかどうか
float	Single	Height	描画範囲（高さ）
float	Single	IntervalX	描画間隔（X 座標に対して）
float	Single	IntervalY	描画間隔（Y 座標に対して）
Bool	Boolean	IsWriteDirect	直接描画するかどうか
Pao.Reports.PmBarcodeType		Kind	バーコードの種類
Bool	Boolean	Kintou	添え字を均等割付するかどうか
Int	Integer	KuroBar	ドット単位で黒バーの幅を調整
float	Single	Point	郵便カスタマバーコードのポイント
String	String	QrErrCorrect	QR コードのエラー訂正レベル(L/M/Q/H)
Int	Integer	QrVersion	QR コードのバージョン(1～40)
Int	Integer	Repeat	繰り返し回数
Int	Integer	ShiroBar	ドット単位で白バーの幅を調整
Bool	Boolean	Soeji	添え字を表示するかどうか
float	Single	Width	描画範囲（幅）
float	Single	X	始点（左上）の X 座標
float	Single	Y	始点（左上）の Y 座標
Pao.Reports.ZFontAttr		z_FontAttr	フォント属性

参照

IObjects インターフェース／z_Objects プロパティ

ZFontAttr クラス／z_FontAttr プロパティ

z_ArtText プロパティ／ ZArtText クラス

z_Objects の一つ下の階層のクラスオブジェクトです。装飾文字列オブジェクトの各プロパティの値を取得・設定する時に、この z_ArtText の下のプロパティの値を取得・設定します。

Z_ArtText 配下の以下のプロパティの値の取得・設定が可能です。

型(C#)	型(VB.NET)	プロパティ名	説明
float	Single	Angle	回転角度
System.Drawing.Color		BackColor	背景色(文字列・画像用)
Int	Integer	CharAngle	文字回転角度
System.Drawing.Color		Color	文字色
Int	Integer	DelimiterPileRatef	桁区切り重ね率__前
Int	Integer	DelimiterPileRater	桁区切り重ね率__後
Bool	Boolean	DelimiterProcess	桁区切り重ね処理
String	String	DelimiterString	桁区切り対象文字
Bool	Boolean	FontBold	フォント太字
String	String	FontName	フォント名
float	Single	Height	描画範囲（高さ）
float	Single	IntervalX	描画間隔（X 座標に対して）
float	Single	IntervalY	描画間隔（Y 座標に対して）
System.Drawing.Color		Color	アウトライン色
float	Single	OutLineWidth	アウトライン幅
Bool	Boolean	PileOrderLeftFront	重ね左前
Int	Integer	PileRate	重ね率
Bool	Boolean	ProjectionX	上下反転
Bool	Boolean	ProjectionY	左右反転
Int	Integer	Repeat	繰り返し回数
Bool	Boolean	RevText	逆転
System.Drawing.Color		Color	影文字色
System.Drawing.Color		Color	影枠線色
float	Single	ShadowLineWidth	影枠線幅
Bool	Boolean	ShadowStretch	影付による伸縮
float	Single	ShadowX	影 X 位置

float	Single	ShadowY	影 Y 位置
Bool	Boolean	ShearStretch	斜体による伸縮
float	Single	ShearX	斜体__横
float	Single	ShearY	斜体__縦
String	String	Text	表示文字列
float	Single	Width	描画範囲(幅)
Bool	Boolean	WriteVertically	縦書き
float	Single	X	始点(左上)の X 座標
float	Single	Y	始点(左上)の Y 座標
Pao.Reports.ZFontAttr		z_FontAttr	フォント属性

参照

IObjects インターフェース／z_Objects プロパティ

ZFontAttr クラス／z_FontAttr プロパティ

z_FontAttr プロパティ ／ ZFontAttr クラス

フォント属性を持つ各オブジェクトのフォントのプロパティ用クラスです。この z_FontAttr の下のプロパティの値を取得・設定することで、各オブジェクト内で使用する文字列のプロパティの値を取得・設定することが可能です。

Z_FontAttr 配下の以下のプロパティの値の取得・設定が可能です。

型(C#)	型(VB.NET)	プロパティ名	説明
Bool	Boolean	Bold	ボールドの場合に true
System.Drawing.Color		Color	文字色
Bool	Boolean	Italic	イタリックの場合に true
String	String	Name	フォント名
float	Single	Size	フォントサイズ
Bool	Boolean	Strikeout	取消線
Bool	Boolean	UnderLine	アンダーライン
System.Drawing.GraphicsUnit		Unit	フォントの高さの単位

参照

ZText クラス／z_Text オブジェクト

ZBarcode クラス／z_Barcode オブジェクト

ZArtText クラス／z_ArtText オブジェクト

z_LineAttr プロパティ ／ ZLineAttr クラス

フォント属性を持つ各オブジェクトのフォントのプロパティ用クラスです。この z_LineAttr の下のプロパティの値を取得・設定することで、各オブジェクト内で使用する罫線のプロパティの値を取得・設定することが可能です。

Z_LineAttr 配下の以下のプロパティの値の取得・設定が可能です。

型(C#)	型(VB.NET)	プロパティ名	説明
System.Drawing.Color		Color	罫線色
float	Single	DashLine	ダッシュの罫線の長さ
float	Single	DashPattern	ダッシュパターン
float	Single	DashSpace	ダッシュの空白の長さ
Pao.Reports.PmLineStyle		Style	罫線種
Pao.Reports.PmLineType		Type	タイプ
float	Single	Width	罫線幅

参照

ZSquare クラス／z_Square オブジェクト
ZCircle クラス／z_Circle オブジェクト
ZLine クラス／z_Line オブジェクト
ZImage クラス／z_Image オブジェクト

z_ PreviewWindow プロパティ ／ IPreviewWindow インターフェース

プレビューウィンドウの設定をお客様プログラムから動的に変更する時に使います。
このプロパティを使用して、プレビューウィンドウのタイトルやアイコンなどを実行時に変更することが可能です。

[IPreviewWindow z_PreviewWindow] 配下の以下のプロパティの値の取得・設定が可能です。

型(C#)	型(VB.NET)	プロパティ名	説明
string	String	z_TitleText	プレビュー画面タイトル
Icon	Icon	z_Icon	プレビュー画面アイコン
int	Integer	z_Top	プレビュー画面上位置(Y 座標)
int	Integer	z_Left	プレビュー画面左位置(X 座標)
int	Integer	z_Width	プレビュー画面幅
int	Integer	z_Height	プレビュー画面高さ
bool	Boolean	z_Maximum	プレビューウィンドウの最大化表示
string	String	z_SavePdfPath	PDF 保存先
string	String	z_SaveXmlPath	印刷データ保存先
bool	Boolean	z_VisibleOpenButton	プレビュー画面 開くボタン表示
bool	Boolean	z_VisibleSaveButton	プレビュー画面 保存ボタン表示
bool	Boolean	z_VisiblePrintButton	プレビュー画面 印刷ボタン表示
bool	Boolean	z_VisibleMenu	プレビュー画面 メニュー表示
IVersionWindow	IVersionWindow	z_ VersionWindow	バージョンウィンドウ の内容を編集可能。 ※次ページで説明
double	Double	z_Zoom	プレビュー表示倍率
bool	Boolean	z_SmallToolbarIcon	プレビュー画面のツールバーに小さいアイコンを表示する場合、true を指定。 既定値： false (大きいアイコン)
bool	Boolean	z_DispToolbarText	プレビュー画面のツールバーに、テキストを表示する場合、true。 アイコンのみ出力して、テキストを表示しない場合、false。 既定値： true (テキストを表示)

z_ VersionWindow プロパティ ／ IVersionWindow インターフェース

プレビューウィンドウの子ウィンドウであるバージョンウィンドウの設定をお客様プログラムから動的に変更する時に使います。
このプロパティを使用して、バージョンウィンドウの会社名や URL などを実行時に変更することが可能です。

型(C#)	型(VB.NET)	プロパティ名	説明
Form	Form	FormVersion	バージョンフォーム
Image	Image	ProductImage	製品画像
string	String	ProductName	製品名
string	String	CopyRight	製品バージョン
string	String	CompanyName	著作権
string	String	Url	会社名
string	String	ProductImage_Top	URL
int	Integer	ProductName_Top	Top 座標 － 製品画像
int	Integer	ProductVersion_Top	Top 座標 － 製品名
int	Integer	CopyRight_Top	Top 座標 － 製品バージョン
int	Integer	CompanyName_Top	Top 座標 － 著作権
int	Integer	Url_Top	Top 座標 － 会社名
int	Integer	ProductImage_Left	Top 座標 － URL
int	Integer	ProductName_Left	Left 座標 － 製品画像
int	Integer	ProductVersion_Left	Left 座標 － 製品名
int	Integer	CopyRight_Left	Left 座標 － 製品バージョン
int	Integer	CompanyName_Left	Left 座標 － 著作権
int	Integer	Url_Left	Left 座標 － 会社名
int	Integer	ProductImage_Size	Left 座標 － URL
Size	Size	ProductName_Font	製品画像サイズ
Font	Font	ProductVersion_Font	製品名 フォント
Font	Font	CopyRight_Font	製品バージョン フォント
Font	Font	Url_Font	著作権 フォント
Font	Font	ProductName_ForeColor	URL フォント
Color	Color	ProductVersion_ForeColor	製品名 文字色
Color	Color	CopyRight_ForeColor	製品バージョン 文字色
Color	Color	Url_ForeColor	著作権 文字色
Color	Color	FormVersion	URL 文字色

変更履歴

版	作成日	変更点
1	2003.05.25	新規作成
2	2003.06.10	QR コード、Web サービス、PDF 対応
3	2006.08.05	ZIP・SVG・SVGZ 対応 プロパティ追加（プリントダイアログ etc）
4	2006.03.02	Write メソッドの value に空文字を指定した場合、 オブジェクトを削除できる機能の説明を追加
5	2010.11.09	SavePDF メソッド：Stream 出力追加
6	2011.02.28	デザイン時のオブジェクトのプロパティを取得・設定する機能追加(z_Objects)
7	2011.07.27	マニュアル不具合修正
8	2012.01.16	繰り返し(行)オブジェクトの属性(フォントや背景色等)を変更できる機能追加(z_Objects.SetObject 引数追加) z_Objects.SetObject("項目名", lineNo); 以下の追加分プロパティの説明追加 AllPage / CutByPage / SwapPdfImage
9	2013.05.10	マニュアル全体見直し。不具合点修正。 プレビュー画面情報を変更するプロパティの記述追加。
10	2013.10.19	VB.NET の Output メソッドの例 PageSettings → PrinterSettings に誤りを訂正。
11	2014.05.30	GetPrintDocument メソッド追加による説明追加。 GetPrintDocument：独自プレビュー用 PrintDocumet 取得 ChangeDefFile メソッド追加による説明追加。 ChangeDefFile：デザインのみ変更。印刷データ再セット不要
12	2014.11.04	- Write メソッドに、GS1-128 の AI 識別子挿入方法の説明追加。 - XPS 出力機能追加による SaveXPS メソッド説明追加 - プレビュー画面ドラッグ＆ドロップ許可制御用 AcceptDragDrop プロパティ説明追加
13	2015.04.01	Azure 対応記述追加
14	2015.09.09	縦横両方向へのオブジェクト繰り返し機能追加による記述追加

版	作成日	変更点
15	2016.04.05	プレビューをタスクバーに入れることを可能とするプロパティ(PreviewInTaskbar)追加による記述追加。
16	2022.04.10	.NET 5 / .NET 6 / Linux Azure / AWS / GCP 等各種クラウド対応に伴う変更・追記
17	2024.05.09	WPF 版印刷プレビュー機能追加に伴う追記