.NET 開発者向け帳票作成ツール

共通マニュアル

第 22 版
2026 年 2 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

https://www.pao.ac/

目次

はじめに

Reports.net は、Microsoft .net framework / .NET Core 系 → .NET 5, .NET 6 で動作する、印刷・プレビュー、および、その処理で使用する Windows 上で動作するデザインファイルの作成ツールの総称です。

Reports.net は、各種 Linux 上で動作します。Linux 上の WEB アプリケーションより PDF 帳票出力や SVG 帳票出力が可能です。また、Linux 上に WEB API として配置すれば、WEB API を利用するクライアントで印刷・プレビュー・PDF 出力等、様々な用途にお使いいただけます。

Reports.net は、次のことを念頭において開発いたしました。

1.軽いこと

何と言っても軽さが命です。Reports.net を利用して印刷・プレビューを行なう場合、Reports.net 自体がシステムに与える負荷は微小です。他の帳票ツールと比較してほんのわずかなメモリ上で動作します。実際の印刷データはＸＭＬとしてローカルディスク上で扱いますので、ＯＳや業務アプリケーションへの影響もほとんどありません。

また、印刷・PDF 出力の速度も他の印刷ツールと比較して、かなり速いことが実証されています。

2.使いやすいこと

一般的な帳票ツールと違い、印刷データをセットするために、従来の帳票ツールのように、いくつかのイベントにロジックを書かなければならないといった煩雑なことがありません。ご自分のプログラムの１箇所に印刷を行うためのロジックを書いていただくだけで OK です。フローが帳票ツールに依存せずすっきりして、業務アプリケーションにロジック的な負荷をかけません。プログラマーに守っていただかなければならない約束ごとは、次の５点のみです。

（１）Reports.net を参照設定する
（２）プリンタ出力かプレビュー表示かを選択する
（３）ページごとの開始宣言
（４）ページごとの終了宣言
（５）実際の印刷（プレビュー）の開始宣言

プログラマーはページの開始宣言と終了宣言の間に自由にロジックを挿入することができます

3.多機能であること

一般に使われるバーコードの印字や、ＰＯＰやチラシ用の文字装飾機能など、多くの機能を搭載しております。Azure / AWS / GCP といったクラウドサービスでもご利用いただけます。もちろん、オンプレミスの WEB サーバでも OK です。

Reports.net は、海外製品の翻訳ではなく、作者がこれまで関わってきた業務系アプリケーションソフト作成において不可欠であると考えてきた要素を数多く盛り込んだ日本の帳票作成に適した製品となっております。ご利用していただく皆さんが.NET 環境における印刷（プレビュー）プログラムの作成作業に、楽しさを感じていただければ幸いです。

2023 年 7 月 Pao@Office 開発チーム

製品特徴

Reports.net には以下のような特徴があります。利用者様における帳票作成シーンにおいて、きっとお役に立てるツールであると自負しています。

- 純国産製品
  - 海外製品の翻訳版ではなく、日本の帳票に要求される機能を盛り込んだ帳票作成ツールとして独自に開発した製品です。固定行数の伝票や罫線中心の表等のデザインが容易である事は勿論の事、陰影を透かす様な機能も搭載しています。
- 印刷データの再利用が可能
  - 印刷データをそのまま XML ファイルに保存し、それを読み込む事が可能です。同じプログラムからは勿論の事、別の Reports.net を用いたプログラムからも印刷やプレビューが可能です。大量に出力する様な帳票を扱う場合、同じ帳票を出力する際に毎回データベースサーバに負荷をかける必要がなくなります。
- ページを完全掌握出来るコード設計が可能
  - ユーザプログラムからは、従来の多くのツールが採用していた各イベントにコードを書く形式ではなく、1 箇所に印刷コードを書いて制御する形式です。帳票作成のためのロジックが分散する事なく、プログラムの流れが Reports.net に依存してしまうという様な事がありません。
- 異なるデザインの帳票の混在が可能
  - 複数種類の帳票を 1 度に印刷(プレビュー)する事が可能です。勿論、そのデータを 1 つの XML ファイルに書き出す事も可能です。
- Azure / AWS / GCP / その他のクラウドサービス、他オンプレミス含む WEB サーバの Linux / Windows Server 対応
  - WEB アプリケーション

    Reports.net で作成した WEB アプリケーションを各種クラウド等にデプロイすることにより、各種クラウドやオンプレの WEB サーバ上の WEB アプリケーションより Reports.net を利用してブラウザに PDF 出力や SVG 出力等を行うことができます。SVG 出力では外部アプリケーションを必要とせず、ブラウザのみで帳票の表示・印刷が可能です。
  - WEB API (REST API 他)

    Reports.net で作成した WEB API を各種クラウド等にデプロイすることにより、Windows クライアント、その他様々なデバイス・プラットフォーム・アプリ WEB アプリケーションらのリクエストに対し、WEB API で印刷データを生成し、呼び出し元がリクエストの戻り値として受けっとった印刷データを利用できます。(印刷・プレビュー、PDF 出力・SVG 出力。etc..)

- 公開されたデータ仕様
  - 帳票のデザインを定義する「デザインファイル」は仕様公開された XML ファイルです。デザイナーを用いない設計や、独自プログラムから帳票定義を動的生成する事等が可能です。
- ドローソフト顔負けの文字装飾機能
  - ＰＯＰやチラシ等にも使える柔軟な文字装飾機能を提供します。
- バーコード描画ロジックを内蔵
  - 一般に使われるバーコードの印字に対応しております。別途バーコード作成ロジックを用意する必要がありません。
- デザイナー
  - 多機能デザイナーでフレシキブルな帳票設計が可能です。また、デザイナーは単独で使用出来るため、帳票デザイン開発とコーディング作業を完全に分離する事が可能です。例えば、エンドユーザ様自身による帳票デザイン変更等のシーンでご活用頂けます。

機能概要

【標準機能】

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デザインを元に、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。「デザインファイル」は仕様公開された XML 形式のファイルで、以下の様な情報を定義する事が可能です。(他にもあります。)

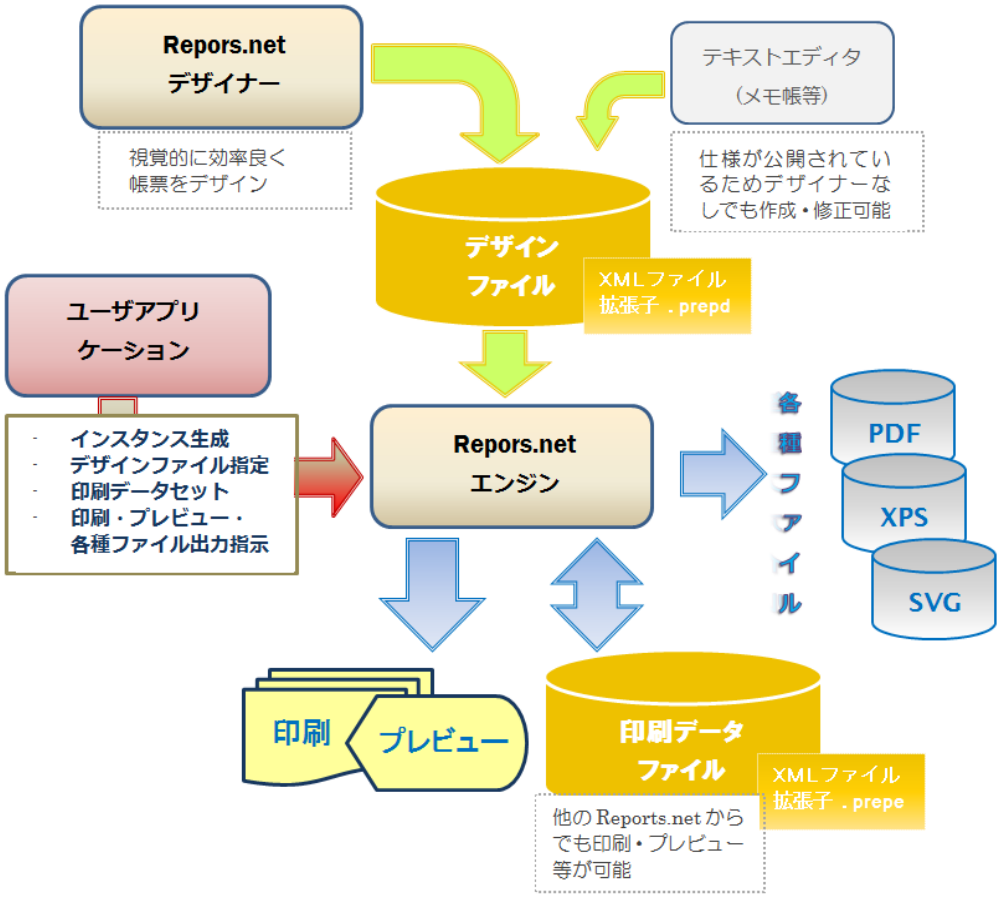
用紙の設定	テキスト情報	罫線情報	四角・角丸情報
丸・楕円情報	画像情報	バーコード情報	装飾文字情報

また、Reports.net デザイナーは、デザインファイルを視覚的に効率良く作成するためのツールです。GUI での直感的な操作により、デザインファイルを簡単に作成する事が出来ます。

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。

2026 年 2 月のバージョンより、SVG 出力機能を大幅に強化しました。GetSvg メソッドにより、プログラムからインライン SVG 埋め込み HTML 文字列を直接取得できます。取得した文字列をブラウザに出力するだけで、帳票のプレビューと印刷が行えます。PDF と異なり、Acrobat 等の外部アプリケーションを起動する必要がなく、ブラウザだけで帳票の表示から印刷までをシームレスに完結できます。特に WEB アプリケーションにおいて、高速かつスマートな帳票プレビュー・印刷手段として、今後は PDF に代わり SVG 出力が活用される場面が増えていくと考えられます。詳しくはプログラマーズマニュアルの「GetSvg メソッド」をご参照ください。また、GetSvgTag メソッドにより、特定ページの SVG タグ文字列のみを取得することも可能です。

※Reports.jar は、PDF 形式ファイルへの出力のみが可能です。



【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

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 帳票や SVG 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。

(6) 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・SVG 出力する等して利用する。

Azure / AWS / GCP /他 Linux / Windows サーバ用 WEB 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・SVG 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
   https://youtu.be/xHNLlPuMFEs
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。

【SVG 出力 ― WEB の時代の帳票プレビュー＆印刷】

WEB での帳票出力といえば PDF――そんな常識が変わります。Reports.net は、帳票を SVG（Scalable Vector Graphics）形式で出力する機能を搭載しています。SVG はブラウザがネイティブに描画できるベクター画像形式であり、PDF 出力とは一線を画す数々のメリットがあります。

SVG が帳票出力に最適な理由

別途アプリケーション不要

PDF の場合、ブラウザの PDF ビューアーや外部プラグインに依存することがありますが、SVG はブラウザが標準で描画します。Acrobat も追加のビューアーも不要。HTML の一部としてシームレスに表示されるため、ユーザーはブラウザを開くだけで帳票のプレビューと印刷が完結します。

スケーリングに圧倒的に強い

SVG はベクター形式なので、どれだけ拡大しても画質が劣化しません。4K モニター、Retina ディスプレイ、スマートフォンの小さな画面――あらゆるデバイスで文字もバーコードも線も美しく鮮明に表示されます。PDF のラスタライズによるぼやけとは無縁です。

スマートフォン・タブレットに強い

モバイルブラウザで SVG を開けば、ピンチ操作で自在にズームでき、そのまま印刷ダイアログへ進めます。PDF のように別アプリへ切り替える手間がありません。帳票の確認から印刷まで、ブラウザひとつで完結します。外出先での帳票確認、タブレットでの現場確認、スマートフォンからの緊急印刷――あらゆるモバイルシーンで威力を発揮します。

印刷品質

SVG のベクターデータはプリンターの最大解像度で出力されます。300dpi、600dpi の高精細印刷でも、文字もバーコードも罫線もすべてクリアに印刷されます。ラスター画像のような解像度不足の心配は一切ありません。

2 つの SVG 出力メソッド

Reports.net では、用途に応じて 2 つの SVG 出力方法を提供しています。

GetSvg() メソッド — 全ページ HTML 出力

インライン SVG を埋め込んだ HTML 文字列を返します。複数ページの帳票を一括でブラウザに表示でき、印刷スタイルシートにも対応しているため、そのまま「Ctrl+P」で印刷できます。WEB アプリケーションのレスポンスとして返すだけで、帳票のプレビューと印刷が即座に実現します。

GetSvgTag() メソッド — 1 ページ SVG タグ出力

指定ページの<svg>...</svg>タグ文字列のみを返します。HTML ラッパーなしの純粋な SVG タグなので、WEB ページの任意の場所に自由に埋め込めます。請求書や納品書など、1 ページの帳票を WEB ページの一部として表示する場合に最適です。JavaScript との連携

も容易で、動的な帳票表示システムの構築にも活用できます。

PDF の時代から SVG の時代へ

WEB アプリケーションにおける帳票出力は、長らく PDF が標準とされてきました。しかし、SVG には以下の優位性があります。

・ブラウザが直接描画するため、表示が高速
・HTML に埋め込めるため、既存の WEB ページとの統合が容易
・レスポンシブデザインに対応 ― 画面サイズに応じた柔軟な表示が可能
・CSS によるスタイリングの適用が可能
・ファイルサイズが PDF より小さくなることが多い
・テキストが選択・検索可能（アクセシビリティに優れる）

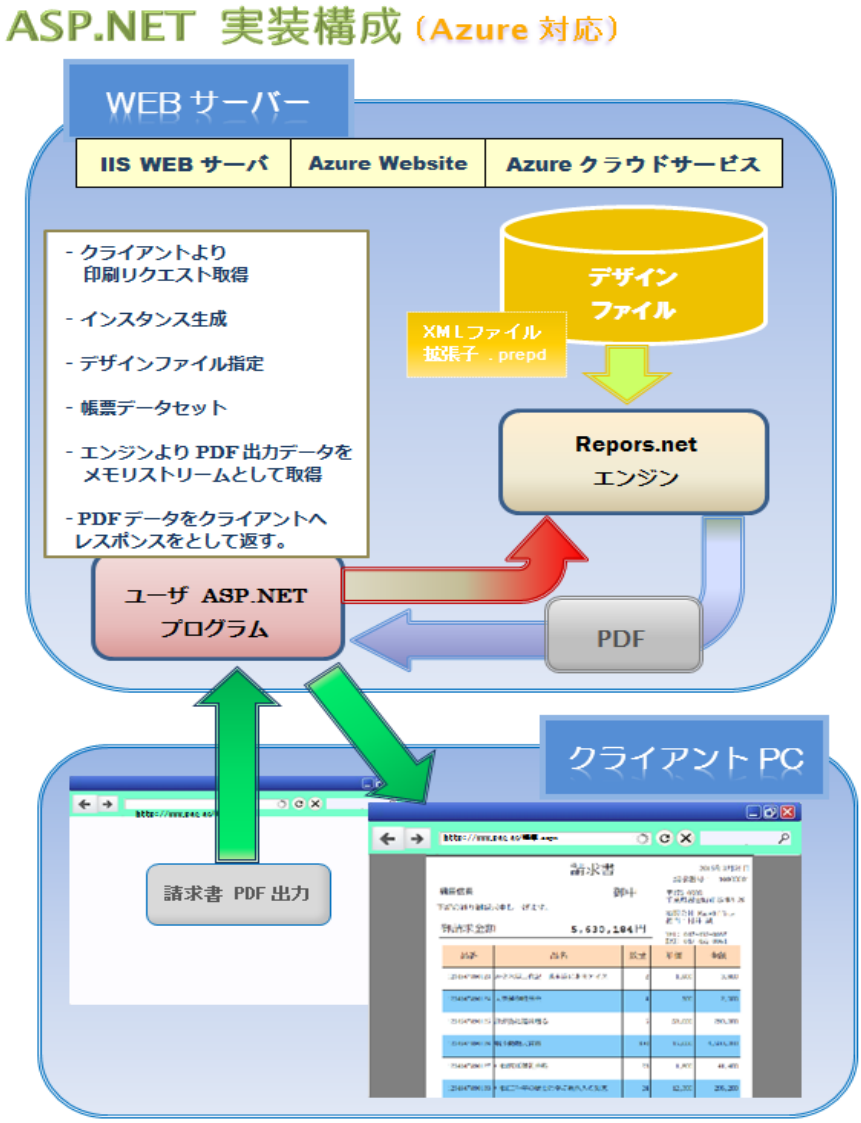
もちろん、PDF が必要な場面（メール添付、長期アーカイブ等）では、引き続き SavePDF メソッドをお使いいただけます。しかし、WEB での帳票プレビューと印刷において、SVG は PDF を超える選択肢です。

ぜひ、Reports.net の SVG 出力をお試しください。WEB での帳票出力の新しい可能性を、きっと実感していただけるはずです。

【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

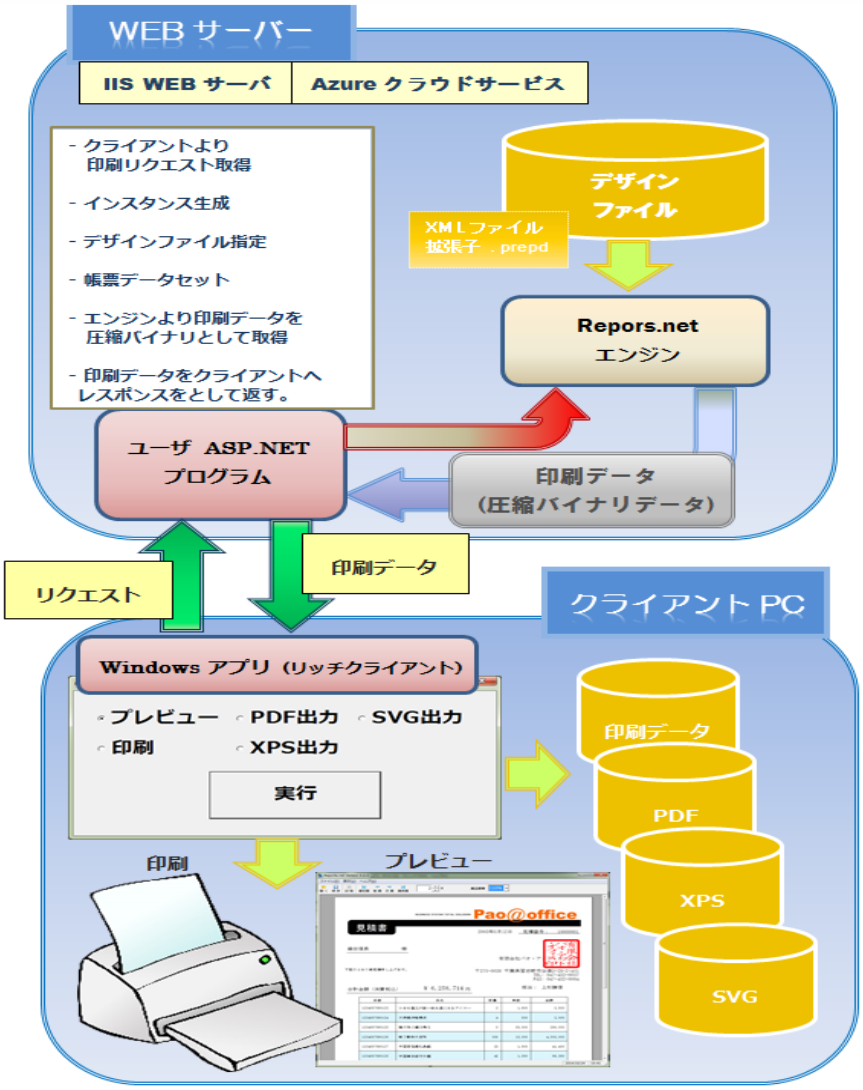
としたアーキテクチャを参照してください。ただし、アーキテクチャが異なるだけで、概要としては、.NET5/.NET6 版と変わるところはございません。
.net framework 版の説明を続けさせていただきます。
ブラウザからの PDF 出力要求に対して、PDF データを生成しブラウザに PDF 出力を行うことができます。内部的にはリクエストに対するレスポンスに PDF データをセットします。
ブラウザへの出力だけでなく PDF ファイルをダウンロードさせることも可能です。
用意しているサンプルプログラムでは、PDF のブラウザ出力／PDF ファイルダウンロードのどちらも可能です。



【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。



マニュアルの構成について

Reports.net のマニュアルは全部で 6 種類あります。それぞれのマニュアルの名称と概要は以下のとおりです。

1. 共通マニュアル（本書）
   Reports.net の全体像を説明しています。基本的にはこのマニュアルをご覧頂けば、Reports.net の Sample ファイルを動かし、その特徴をご理解頂ける内容になっています。

2. デザイナー操作説明書
   デザイナーの操作方法を説明しています。デザイナーの各部の名称や機能、簡単にデザインを行うためのテクニック等についてはこのマニュアルをご覧ください。

3. エンジン(プレビューア)操作説明書
   単体で起動可能なプレビュー画面の操作方法を説明しています。

4. Reports.net プログラマーズマニュアル
   エンジンを.NET アプリケーションプログラムから操作する方法を説明しています。Reports.net をプログラムから呼び出して用いる際のメソッド等についてはこのマニュアルをご覧ください。

5. Reports.jar プログラマーズマニュアル
   エンジンを java アプリケーションプログラムから操作する方法を説明しています。Reports.jar をプログラムから呼び出して用いる際のメソッド等についてはこのマニュアルをご覧ください。

6. デザインファイル仕様書
   デザインファイルの詳細な仕様を説明しています。Reports.net のデザインファイル仕様は完全に公開されており、この仕様書の内容に従って XML ファイルを作成すれば、デザイナーなしで Reports.net をご利用頂く事も可能です。

用語説明

1. エンジン
   Reports.net の本体にあたる部分です。「デザインファイル」により定義された帳票に対して、印刷・プレビュー・印刷データ作成等の機能を提供します。
2. デザイナー
   Reports.net 用のデザインファイルを作成するためのツールです。GUI での直感的な操作により、デザインファイルを簡単に作成する事が出来ます。
3. デザインファイル
   Reports.net で扱う帳票を定義する XML 形式のファイルです。仕様公開しているため、デザイナーからだけではなく、メモ帳等のテキストエディタや、利用者様自身が独自に作成したアプリケーションから作成する事も可能です。詳細な仕様については、「デザインファイル仕様書」をご参照下さい。
4. 印刷データファイル
   Reports.net で作成した帳票のデータを格納した XML ファイルです。このファイルには、印刷イメージの全データが含まれているため、同じプログラムからは勿論の事、他の Reports.net からでも印刷やプレビューが可能です。
5. オブジェクト
   Reports.net のレポートで取り扱う部品の事です。オブジェクトは「フォーム」「文字」「線」「四角」「丸」「画像」「バーコード」「装飾文字」の 8 種類です。

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 / .NET 7 / .NET 8 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 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 / .NET 8 .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。

Reports.net WPF 版対応

バージョン 10 の新機能

Reports.net ver.9 以前の WPF 対応は簡易版であり、一度テンポラリの XPS ファイルを経由して印刷・印刷プレビューを実現していました。ver.10 からは、XAML を用いたフルカスタマイズが可能なプレビュー画面を実装しています。プログラムからは、Windows Form または WPF のいずれかの印刷プレビュー画面で出力できます。インスタンス生成方法だけが異なります。

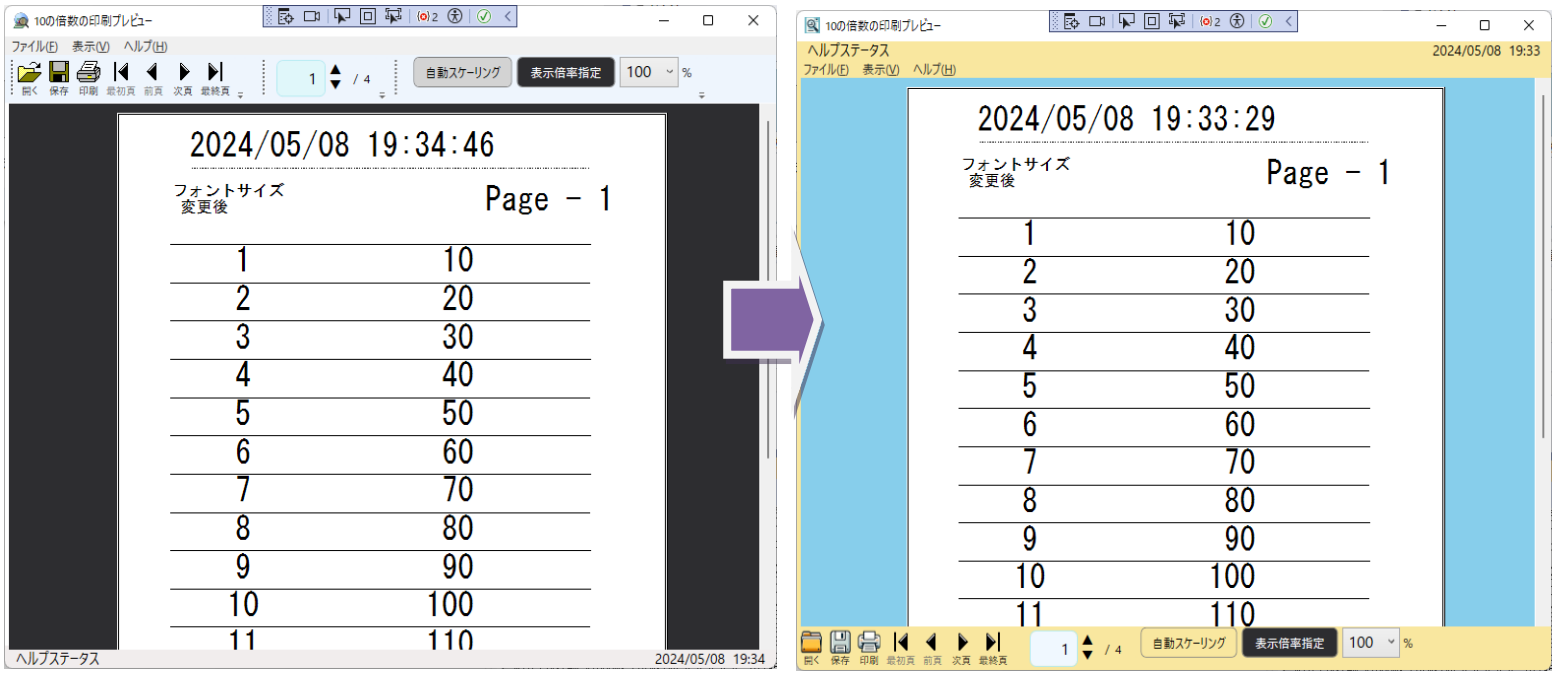
```csharp

// Windows Forms プレビュー例
IReport previewWinForms = ReportCreator.GetPreview();
:
previewWinForms.Output();

// WPF プレビュー例
IReport previewWpf = ReportCreator.GetPreviewWpf();
:
previewWpf.Output();
```

プレビューアの exe を直接起動する場合や、.prepe 拡張子のファイルを開く際のデフォルトプログラムを、Windows Form 版または WPF 版プレビューアのどちらかに設定可能です。

WPF 版印刷プレビュー - デザインカスタマイズ

WPF 版では、プレビュー画面のデザインを自在に変更できるようになりました。

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。

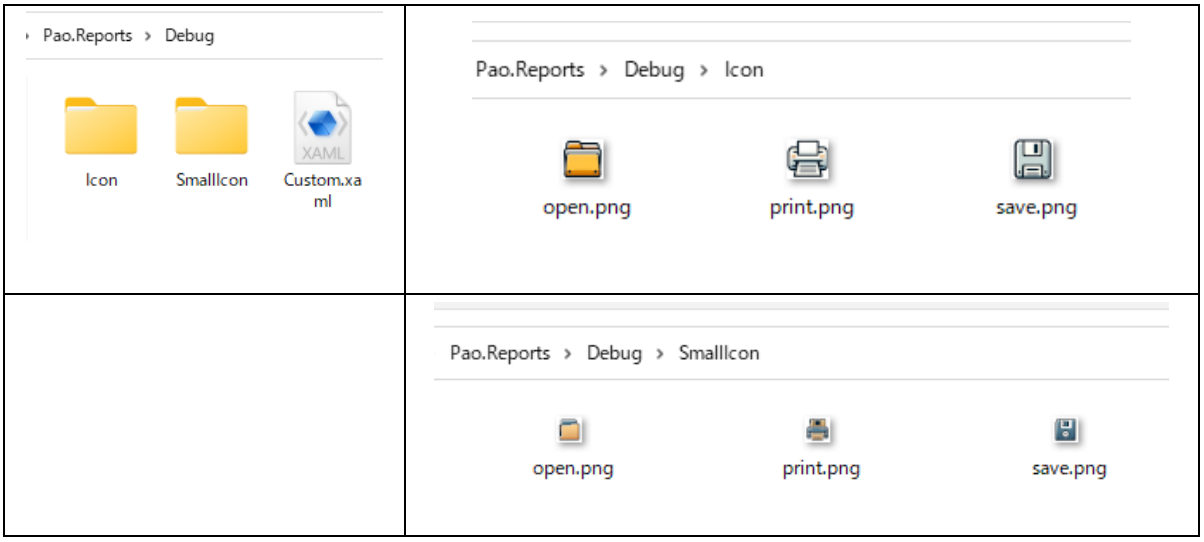
   変更前デフォルト xaml　Custom.xaml のサンプル

   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

表示倍率変更の操作性向上

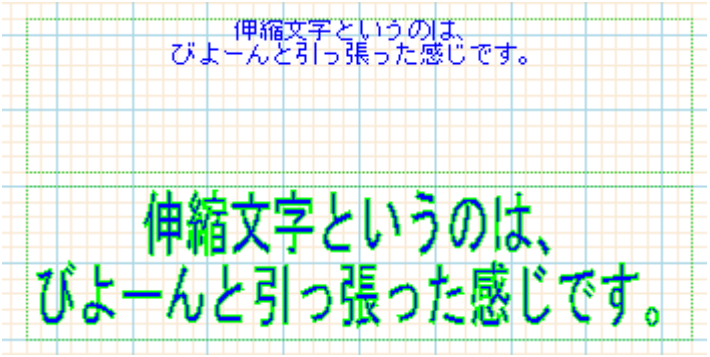
これまでの操作性と重複する個所もありますが、表示倍率変更操作を次の通り拡充しました。

(1) 「自動スケーリング」と「表示倍率指定」
バージョン 9 までは「表示倍率の指定」しか行えませんでしたが、バージョン 10 では「自動スケーリング」が追加されました。「自動スケーリング」トグルボタンを選択することで、ウィンドウのサイズに合わせて自動的にページ全体が収まるように印刷プレビューを行える機能が追加されました。

(2) 表示倍率の変更操作
ドロップダウンリストでの表示倍率の変更に加え、次の操作で印刷プレビューの拡大・縮小を行うことができます。
・Ctrl キーを押しながら、+キーを押すと拡大、-キーを押すと縮小。
・Ctrl キーを押しながら、マウスホイールを上下にスクロール。上で拡大、下で縮小。

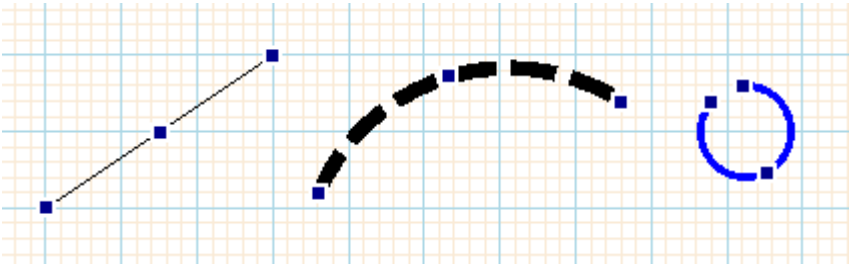
Reports.net における各オブジェクトの特徴

□ 文字



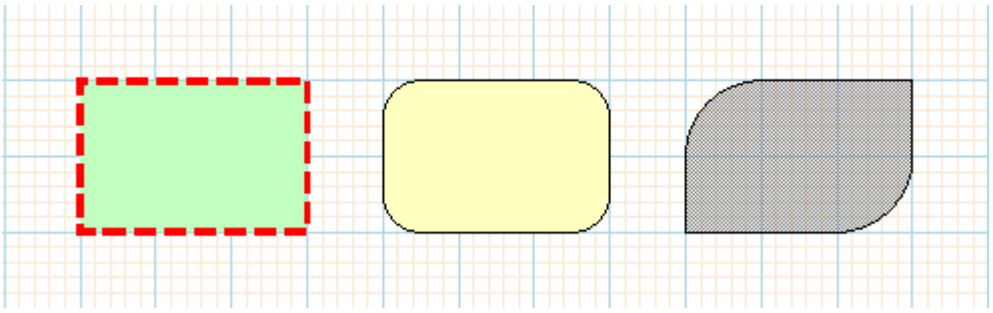
通常のテキストです。フォントや文字の位置、色等を指定します。また、伸縮文字という機能を使い、オブジェクト一杯に文字を伸縮させて描画する事も可能です。

□ 線



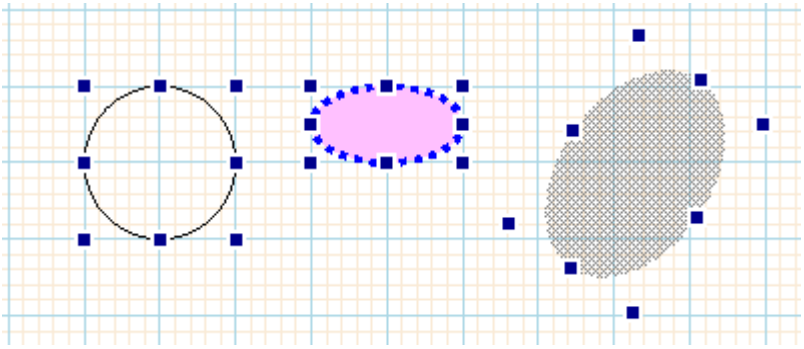
直線及び円弧です。円弧は両端と厚みを指定して描画します。

□ 四角



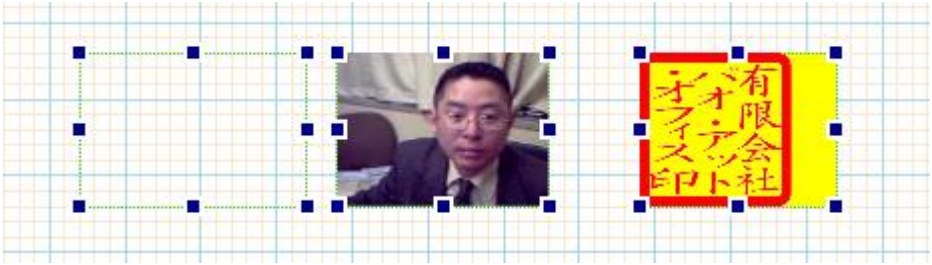
四角形です。角に丸みをつけたり網掛けを行ったりする事が出来ます。

□ 円



円及び楕円です。大きさは外接する四角形で表現されます。

□ 画像



画像を挿入します。画像データは外部にあるファイル及びデザインファイル内にエンコードされたデータ(デザイナーで作成します)に対応しています。画像を枠一杯に、もしくは、縦横の比率を維持して伸縮させる機能や、左右、上下に反転させる機能があります。透過 PNG/GIF を用いると、背景が透けるため、印影等の利用も可能です。

□ バーコード

バーコードを描画します。以下の種類に対応しています。

- ✧ JAN13(EAN13)
- ✧ JAN8(EAN8)
- ✧ UPC-A
- ✧ UPC-E
- ✧ ITF(インターリーブド 2 of 5)
- ✧ Matrix 2 of 5
- ✧ NEC 2 of 5 (Coop 2 of 5)
- ✧ NW7(Codebar)
- ✧ Code39
- ✧ Code128
- ✧ GS1-128 (UCC/EAN128)
  - コンビニ向け標準料金代理収納用バーコード
  - 医療用 医薬品等のバーコード
  - 医療用 医療材料等のバーコード
  - 食肉標準物流バーコード「基本バーコード」
- ✧ 郵便カスタマバーコード
- ✧ QR コード

☆ バーコードの描画方法は、

- **指定幅以内に一番大きな幅長で、直接描画する方法**
- **指定幅ピッタリに縮小描画する方法**

の２種類に対応しております。描画精度・速度は、直接描画する方が良いですが、幅は、プリンタの解像度(dpi)に依存します。どうしてもデザイン上、幅をピッタリに合わせたい場合は、縮小描画を使用してください。

☆ バーコードの黒バー・白バーのドット単位の幅調整が可能です。
プリンタにより、微調整が必要な場合にこの機能を使用してください。
例)　コンビニバーコード、EPSON PX-502A(360DPI)のプリンタの場合、
白バーを＋１すると丁度良い。

☆ GS1-128(UCC/EAN128)の特記事項
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
⇒(01)の前には FNC1 を挿入し(21)の前には挿入しない。
コード指定方法 → ”{FNC1}0104512345670016{AI}211”

○ 一次元バーコードの種類

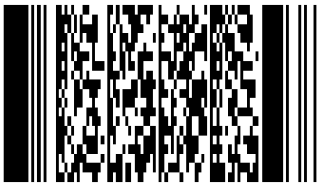
次の種類のバーコードを出力できます。

1．JAN-13(EAN-13)	2．JAN-8(EAN-8)
数字 13 桁。12 桁指定時、13 桁目のチェックディジットを自動計算付与。 	数字 8 桁。7 桁指定時、8 桁目のチェックディジットを自動計算付与。 
3．UPC-A	**4．UPC-E**
数字 12 桁。11 桁指定時、12 桁目のチェックディジットを自動計算付与。 	数字 7 桁。6 桁指定時、7 桁目のチェックディジットを自動計算付与。 
5．ITF(インターリーブド 2 of 5)	**6．Matrix 2 of 5**
使用可能文字：数字 	使用可能文字：数字 
7．NEC 2 of 5 (Coop 2 of 5)	**8．NW-7(CODA-BAR)**
使用可能文字：数字 	使用可能文字： ABCD.+:/$-0123456789  1 文字目 ABC いずれか入力した文字がスタート・ストップキャラクタとなる。入力がない場合 C が既定値。 **パラメータ：** スタート・ストップキャラクタ表示／非表示

９．CODE39	１０．CODE93
使用可能文字： 1234567890ABCDEFGHIJKLMNOPQRSTUVWXYZ-. *$/+%  **パラメータ**： スタート・ストップキャラクタ表示／非表示	使用可能文字： 1234567890ABCDEFGHIJKLMNOPQRSTUVWXYZ-. *$/+% 
１１．CODE128	**１２．GS1-128 (UCC/EAN-128)**
多くの半角英数文字記号・エスケープシーケンスが使用可能。 CODE A/B/C により出力文字・パターンを切り替えることが可能。  **パラメータ**： どのコードパターンを使用するか。 AUTO / CODE_A / CODE_B / CODE_C 既定値は、AUTO。AUTO は、数時 4 文字続けば CODE A にコードチェンジするなど､バーコードが一番小さくなるように自動調整する。	- コンビニ向け標準料金代理収納用バーコード - 医療用 医薬品等のバーコード - 医療用 医療材料等のバーコード - 食肉標準物流バーコード「基本バーコード」  コンビニバーコード 入力： {FNC1}9191234500000000000004520875004013100295006 
１３．郵便カスタマバーコード	
郵便番号＋住所の英数字部分のみ入力 例：27500263-29-2-401  **パラメータ**：8～11.5 ポイント(大きさ)	

○ 二次元バーコードの種類

次の種類のバーコードを出力できます。

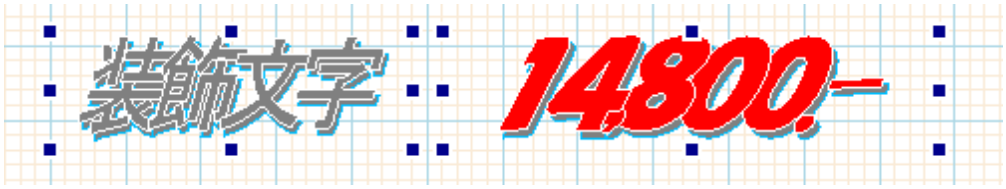
1．QR コード
パラメータ：(既定値はアンダーバー) (1) エラー訂正レベル：L / M / Q / H (2) 文字種：数字 / 大文字英数字 / 8 ビットバイトデータ(漢字含む) (3) バージョン： 1～40 5 (4) 全角文字エンコード： “utf8” / “shift_jis” /etc.. 
2．DataMatrix (GS1 DataMatrix)
パラメータ：(既定値はアンダーバー) (1) シンボルコードサイズ：自動 / 10x10～144x144 / 8x18～16x48 等 (2) 全角文字エンコード： “utf8” / “shift_jis” / etc.. 
3．PDF417
パラメータ：(既定値はアンダーバー) (1) エラー訂正レベル自動決定：する / しない (2) エラー訂正レベル：1～8 2 (3) アスペクト比：0.5 / 1.0 / 2.0 等 (4) 列・行数決定方法： アスペクト比より自動決定 / 列数指定 / 行数指定 / 列数行数指定 (5) 指定列数：1～30 5 (6) 指定行数：3～90 5 (7) 全角文字エンコード： “utf8” / “shift_jis” /etc.. 

○ GS1 Databar (RSS) の種類

次の種類のバーコードを出力できます。

<table>
<tr><th colspan="2">1．GS1 データバーOmni-directional / 表示桁数：数字 14 桁(GTIN)</th></tr>
<tr><td>Omni-directional (標準型)
 
高さが狭い Truncated(カット型)という種類もございますが、この標準型の高さを調整して出力してください。</td><td>Stacked (2 層型)
  </td></tr>
<tr><td colspan="2">Stacked Omni-directional (標準 2 層型)
  </td></tr>
<tr><th colspan="2">2．GS1 データバーLimited / 表示桁数：数字 14 桁(GTIN)</th></tr>
<tr><td colspan="2">Limited(限定型)
  </td></tr>
<tr><th colspan="2">3．GS1 データバーExpanded / 表示桁数：最大数字 74 桁または英字 41 文字</th></tr>
<tr><td>Expanded (拡張一層型)
  </td><td>Stacked Expanded (拡張多層型)
  </td></tr>
</table>

□ 装飾文字



POP やチラシ等で使える装飾文字です。様々なパラメータを指定して、柔軟な装飾を施した文字を描画できます。

□ 各オブジェクトのプロパティの変更

デザイン時のみでなくプログラム実行時に、オブジェクトの表示・非表示、位置や色など、プロパティの値の取得・変更が可能です。(z_Objecs)

Reports.net を使用して帳票を作成する方法

ここでは、Reports.net の試用版に付属するサンプルプログラムを使用して、帳票を作成する方法を簡単に説明します。

1．サンプルファイルについて

- 郵便番号一覧

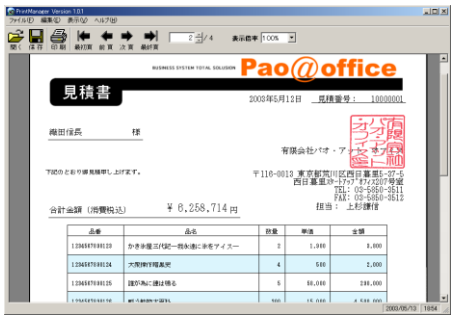
  連票形式での例です。件数が非常に多いデータとして、郵便番号のデータを 6 頁目からレイアウトが変わる帳票に一覧印刷します。ページ数の振り方や罫線の扱い方等、Reports.net における帳票設計の考え方が非常にシンプルである事が確認頂けます。また、A4 縦で 82 頁あるため、全体的な処理スピードもご確認頂けると思います。なお、枚数が多いため、いきなり印刷してしまわないように注意して下さい。

- 広告

  単票形式での例です。外部の画像ファイルの挿入も行っています。装飾文字やバーコード(10 種類のバーコードの描画機能を登載しています)等を活用すると、広告や価格 POP の様な帳票も簡単に作成することができます。

- 見積書

  伝票形式の例です。海外製レポートツールが苦手とする様な、あらかじめ固定で行数が決められている日本的な帳票であっても、Reports.net を用いれば簡単に作成することが可能です。また、透過 PNG/GIF に対応しているため、印影を最前面に配置して後の文字等を透けさせるというサインでなくハンコの文化にも対応しています。



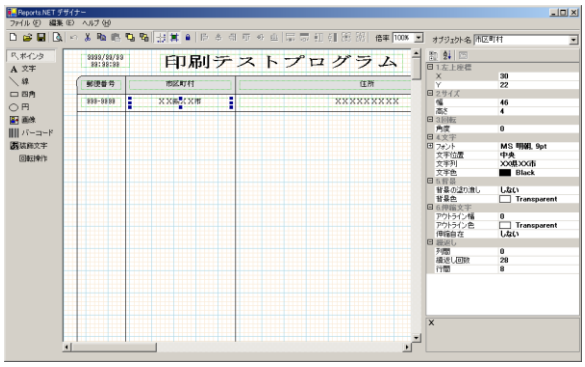
２．デザインファイルの作成

Reports.net で帳票を作成する場合には、必ず「デザインファイル(デザインファイル)」が必要です。「デザインファイル(デザインファイル)」は、「レポート定義 XML ファイル仕様書」に従ってテキストエディタで作成する方法と、デザイナーで作成する方法があります。サンプル内の各デザインファイルをデザイナーやテキストエディタ等で開いて、内容を確認してみてください。

- 郵便番号一覧
  - PaoRep1.xml
  - PaoRep2.xml
- 広告
  - 広告.xml
- 見積書
  - 表紙.xml
  - 見積書.xml

「PaoRep1.xml」をデザイナーで開いた例

印刷時には繰り返されている行が１行しか定義されていない事が確認出来ます。



「PaoRep1.xml」をデザイナーからプレビューした例

「PaoRep1.xml」をテキストエディタで開いた例

```
<?xml version="1.0" encoding="UTF-8" ?>
- <PrintDefine>
  - <PrintSetting Name="レポート定義ファイル" Ver="1.00">
    - <PrintAttrs>
        <FontAttr AttrName="Font1" Name="ＭＳ 明朝" Size="9" />
        <FontAttr AttrName="Font2" Name="ＭＳ 明朝" Size="12" Bold="True" />
        <LineAttr AttrName="Line1" Width="0.3" />
        <LineAttr AttrName="Line2" Color="0" Width="0" />
      </PrintAttrs>
      <Paper Size="A4" Direction="False" />
      <Design FontAttr="Font1" LineAttr="Line1" GridWidth="2" GridFit="True" DesignScale="100" />
    </PrintSetting>
  - <PrintObjects>
      <Square Name="Square1" X="6" Y="12" Width="190" Height="8" Angle="0" LineAttr="Line1"
        PaintColor="ffe0e0e0" R="3" BR="Angle" BL="Angle" />
      <Text Name="Field1" X="44" Y="2" Width="110" Height="8" Angle="0" Text="印刷テストプログラム
        " Elastic="True" Align="Center" FontAttr="Font2" />
      <Text Name="郵便番号" X="8" Y="22" Width="18" Height="4" Angle="0" Text="999-9999"
        Elastic="False" Align="Center" FontAttr="Font1" IntervalY="8" Repeat="28" />
      <Text Name="Field3" X="8" Y="14" Width="18" Height="4" Angle="0" Text="郵便番号"
        Elastic="False" Align="Center" FontAttr="Font1" />
      <Line Name="横罫線" StartX="6" StartY="28" EndX="196" EndY="28" LineAttr="Line1" IntervalY="8"
        Repeat="28" />
      <Text Name="Field4" X="30" Y="14" Width="46" Height="4" Angle="0" Text="市区町村"
        Elastic="False" Align="Center" FontAttr="Font1" />
      <Text Name="Field5" X="80" Y="14" Width="116" Height="4" Angle="0" Text="住所" Elastic="False"
        Align="Center" FontAttr="Font1" />
      <Text Name="市区町村" X="30" Y="22" Width="46" Height="4" Angle="0" Text="ＸＸ県ＸＸ市"
        Elastic="False" Align="Center" FontAttr="Font1" IntervalY="8" Repeat="28" />
      <Text Name="住所" X="80" Y="22" Width="116" Height="4" Angle="0" Text="ＸＸＸＸＸＸＸＸ"
        Elastic="False" Align="Center" FontAttr="Font1" IntervalY="8" Repeat="28" />
      <Text Name="日時" X="8" Y="2" Width="22" Height="6" Angle="0" Text="9999/99/99 99:99:99"
        Elastic="False" Align="Center" FontAttr="Font1" />
      <Text Name="ページ" X="180" Y="4" Width="16" Height="4" Angle="0" Text="Page-999"
        Elastic="False" Align="Left" FontAttr="Font1" />
      <Line Name="Line2" StartX="28" StartY="12" EndX="28" EndY="244" LineAttr="Line1" />
      <Line Name="Line3" StartX="78" StartY="12" EndX="78" EndY="244" LineAttr="Line1" />
      <Line Name="Line4" StartX="196" StartY="20" EndX="196" EndY="244" LineAttr="Line1" />
      <Line Name="Line5" StartX="6" StartY="20" EndX="6" EndY="244" LineAttr="Line1" />
      <Barcode Name="バーコード" X="6" Y="248" Width="60" Height="14" Angle="0" StartStop="False"
        Soeji="True" Kintou="True" FontAttr="Font1" Kind="Code128" Code="123456789012" />
      <ArtText Name="装飾文字" X="14" Y="260" Width="122" Height="28" Angle="0" FontName="Arial
        Black" FontBold="True" Color="Red" OutLineWidth="1" OutLineColor="White" PileRate="50"
        DelimiterProcess="True" DelimiterString=",." DelimiterPileRater="30" ShearX="-0.35"
        ShearStretch="True" ShadowX="3" ShadowY="3" ShadowColor="Gray"
        ShadowLineColor="DeepSkyBlue" ShadowLineWidth="0.3" Text="Reports.NET" />
    </PrintObjects>
```

※デザイナーで作成したデザインファイルの場合、画像ファイルがデザインファイル内にエンコードして格納されています。「見積書.xml」がエンコードされた画像が格納された例となっています。

３．ソースコードの記述

アプリケーションから Reports.net を用いる場合の最小セットは以下の様になります。

```
1.インスタンス生成
2.レポート定義ファイル指定
3.頁開始宣言
4.データセットロジックを記述
                :
                :
5.頁終了宣言
6.プレビューもしくは印刷実行
```

プログラムからの帳票作成においては、印刷する各頁に対して１箇所でコードを記述する事が出来る形式のため、帳票作成のためのロジックが分散する事なく、プログラムの流れが Reports.net に依存してしまうという様な事がありません。

つまり、Reports.net を使えば、多くの帳票作成ツールでみられるセクション(「ヘッダ」「詳細」「フッタ」等)の各イベントに対してコードを記述する事による業務ロジックの散在がありません。そのため、簡単に、かつ、メンテナンスが行いやすいソースコードを作成することが可能となるのです。(各イベントにコードを記述する方法では、イベントの制御の特性に皆様が書くコードが左右されてしまう事が難点だと我々は考えています。)

- C#.NET の場合の記述例
  - ✧ インスタンスの生成

```
IReport paoRep = null;
paoRep = ReportCreator.GetPreview();..... プレビューの場合
paoRep = ReportCreator.GetReport();..... 印刷の場合
```

  - ✧ デザインファイル指定

```
paoRep.LoadDefFile("ほげほげ.xml");
```

  - ✧ 頁開始宣言

```
paoRep.PageStart();
```

  - ✧ オブジェクトへの値の設定

```
paoRep.Write("オブジェクト名", "設定する値");
        :
        :
```

  - ✧ 頁終了宣言

```
paoRep.PageEnd();
```

  - ✧ プレビューもしくは印刷実行

```
paoRep.Output();
```

- VB.NET の場合の記述例
  - ✧ インスタンスの生成

```
Dim paoRep As Pao.Reports.IReport = Nothing
paoRep = ReportCreator.GetPreview()..... プレビューの場合
paoRep = ReportCreator.GetReport()..... 印刷の場合
```

  - ✧ デザインファイル指定

```
paoRep.LoadDefFile("ほげほげ.xml")
```

  - ✧ 頁開始宣言

```
paoRep.PageStart()
```

  - ✧ オブジェクトへの値の設定

```
paoRep.Write("オブジェクト名", "設定する値")
        :
        :
```

  - ✧ 頁終了宣言

```
paoRep.PageEnd()
```

  - ✧ プレビューもしくは印刷実行

```
paoRep.Output()
```

➢ オブジェクトへの値の設定例

✧ 「文字 A」という文字オブジェクトに「ほげほげ」という文字列を設定。

- C#.NET

```
paoRep.Write("文字 A", "ほげほげ");
```

- VB.NET

```
paoRep.Write("文字 A", "ほげほげ")
```

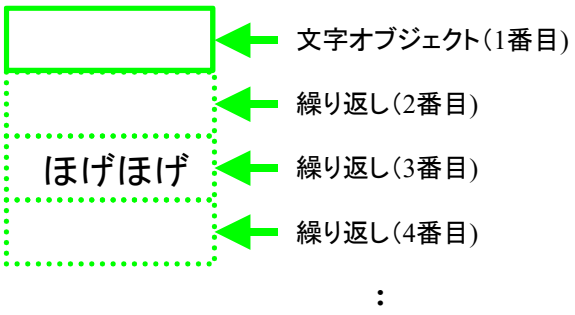
✧ 繰り返し設定がされている「文字 B」という文字オブジェクトの 3 番目に「ほげほげ」という文字列を設定。

- C#.NET

```
paoRep.Write("文字 B", "ほげほげ", 3);
```

- VB.NET

```
paoRep.Write("文字 B", "ほげほげ", 3)
```



✧ 繰り返し設定された「四角 A」という四角オブジェクトを 7 回繰り返して作成する設定。(繰り返されたオブジェクトは Write しなければプレビュー及び印刷時には描画されません。)

- C#.NET

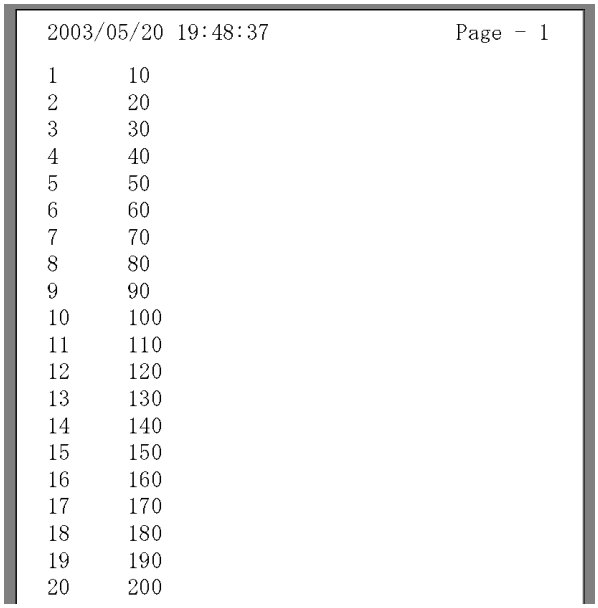
```
for (int i = 1 ; i <= 7 ; i++ ){
paoRep.Write("四角 A", i);}
```

- VB.NET

```
Dim i As Int16
For i = 1 To 7
paoRep.Write("四角 A ", i)
Next I
```

４．コーディング例

コマンドボタンをクリックすると、日付とページ数をヘッダに、行番号とその番号を 10 倍した数を 1 頁 20 行の明細にプレビューするプログラムの例です。



2003/05/20 19:48:37 Page - 1

1	10
2	20
3	30
4	40
5	50
6	60
7	70
8	80
9	90
10	100
11	110
12	120
13	130
14	140
15	150
16	160
17	170
18	180
19	190
20	200

デザインファイルには「日付」「頁数」「行番号」「10 倍数」という 4 つの文字オブジェクトがあります。

(PaoRep.xml)

```
<?xml version="1.0" encoding="UTF-8" ?>
- <PrintDefine>
  - <PrintSetting Name="レポート定義ファイル" Ver="1.00">
    - <PrintAttrs>
        <FontAttr AttrName="Font1" Name="ＭＳ 明朝" Size="24" />
      </PrintAttrs>
      <Paper Size="A4" Direction="False" />
    </PrintSetting>
  - <PrintObjects>
      <Text Name="行番号" X="10" Y="20" Width="20" Height="10" Text="行番号"
        Align="Left" FontAttr="Font1" IntervalY="10" Repeat="20" />
      <Text Name="10 倍数" X="40" Y="20" Width="30" Height="10" Text="10 倍数"
        Align="Left" FontAttr="Font1" IntervalY="10" Repeat="20" />
      <Text Name="日付" X="10" Y="4" Width="110" Height="10" Text="日付"
        Align="Left" FontAttr="Font1" />
      <Text Name="頁数" X="140" Y="4" Width="60" Height="10" Text="頁数"
        Align="Right" FontAttr="Font1" />
    </PrintObjects>
  </PrintDefine>
```

➢ C#のコーディング例

```
using Pao.Reports;


private void button1_Click(object sender, System.EventArgs e)
{
    //インスタンスを生成
    IReport paoRep = null;

    //プレビューを宣言
    paoRep = ReportCreator.GetPreview();

    //レポート定義ファイルを指定
    paoRep.LoadDefFile("PaoRep.xml");

    int page = 0; //頁数を定義
    int line = 0; //行数を定義

    for (int i = 0; i < 80; i++)
    {
        if (i % 20 == 0) //1頁20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

         //日付をセット
         paoRep.Write("日付", System.DateTime.Now.ToString());
         //頁数をセット
         paoRep.Write("頁数", "Page - " + page.ToString());
        }
        line++; //行数をインクリメント

        paoRep.Write("行番号", (i+1).ToString() , line);
        paoRep.Write("10倍数", ((i+1)*10).ToString() , line);

        if (((i+1) % 20) == 0) paoRep.PageEnd(); //1頁20行で終了
    }
    // プレビューを実行
    paoRep.Output();
}
```

➢ VB.NET のコーディング例

```
Imports Pao.Reports

Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click

        'インスタンスを生成
        Dim paoRep As IReport = Nothing

        'プレビューを宣言
        paoRep = ReportCreator.GetPreview()

        'レポート定義ファイルを指定
        paoRep.LoadDefFile("PaoRep.xml")

        Dim page As Int16 = 0 '頁数を定義
        Dim line As Int16 = 0 '行数を定義

        Dim i As Int16

        For i = 0 To 79
            If (i Mod 20 = 0) Then '1頁20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
            End If

            '日付をセット
            paoRep.Write("日付", System.DateTime.Now.ToString())
            '頁数をセット
            paoRep.Write("頁数", "Page - " + page.ToString())
            line = line + 1 '行数をインクリメント

            paoRep.Write("行番号", (i + 1).ToString(), line)
            paoRep.Write("10倍数", ((i + 1) * 10).ToString(), line)

            If (((i + 1) Mod 20) = 0) Then paoRep.PageEnd() '1頁20行で終了
        Next i
        ' プレビューを実行
        paoRep.Output()
    End Sub
```

➢ オブジェクトのプロパティ変更

実行時に、以下のコードのように z_Objects を使用して各オブジェクトのプロパティ値を取得・設定できます。

＜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
```

※ここでは、行の繰り返し機能について記述させていただきました。
Ver 7.0 以降、縦横両方向への繰り返しが可能となっております。
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。

製品版について

Reports.net は、

https://www.pao.ac/reports.net/

に試用版として最新バージョンを常にご用意させていただいております。

□ 試用版の制限事項(下記制限以外は製品版と同じです。)

- デザイナー
  - 100 回までしか保存が出来ません。
    保存時に以下のようなダイアログボックスが表示されます。

- エンジン
  - プレビュー及び印刷時に「SAMPLE」という文字列が挿入されます。(デザインファイルのプレビュー及び印刷時には挿入されません。)

- Reports.jar(Java 版　エンジン)
  - 印刷データに「JAVA SAMPLE」という文字列が挿入されます。

□ 試用版から製品版への制限解除方法

ライセンス登録時に弊社よりライセンスコードを送付させて頂きます。
ご利用になるパソコンの Reports.net デザイナーにライセンスコードを入力して下さい。
試用版制限が解除され、製品版として、Reports.net をご利用いただけます。

使用許諾

Reports.net(エンジン及びデザイナー)の使用について、Reports.net の使用者(以下「利用者様」と称します)と有限会社パオ・アット・オフィス(以下「弊社」と称します)は、以下の各項目についての内容に同意するものとします。

1.Reports.net の使用に関する使用許諾書

この使用許諾書は、利用者様がお使いのパソコンにおいて、Reports.net を使用する場合に同意しなければならない契約書です。

2.使用許諾書の同意

利用者様が Reports.net を使用する時点で、本使用許諾書に同意されたものとします。同意されない場合は、Reports.net を使用する事はできません。

3.試用制限

Reports.net は、いつでもどこでも誰でもその機能を試していただくことが可能です。ただし、デザイナーでは一定回数以上の保存ができない、エンジンでは帳票に「SAMPLE」という文字が挿入されるという制限があります。

4.ライセンス(使用権)の購入

利用者様が Reports.net を正式に使用し続ける場合には、1 台のコンピュータで Reports.net を使用するにあたり、1 ライセンスを購入する必要があります。

5.著作権

Reports.net 及の著作権は、いかなる場合においても弊社に帰属いたします。

6.免責

Reports.net の使用によって、直接的、あるいは、間接的に生じた、いかなる損害に対しても、弊社は補償賠償の責任を負わないものとします。

7.禁止事項

Reports.net 及びその複製物を第三者に譲渡・貸与する事は出来ません。Reports.net を開発ツールとして再販/再配布することを禁止します。なお、モジュールとしてアプリケーションソフトに組み込みを行い再販/再配布する場合は、開発ツールとしての再販/再配布には含まれません。

8.保証の範囲

弊社は Reports.net の仕様を予告無しに変更することがあります。その場合の利用者様に対する情報提供は、弊社 WebSite にて行う事とします。

9.適用期間

本使用許諾条件は利用者様が Reports.net を使用した日より有効です。利用者様が本使用許諾条件のいずれかの条項に違反した場合、又は、本許諾条件に同意出来ない場合は、利用者様は Reports.net を一切使用出来ないものとします。

代金支払い方法(ユーザ登録の方法)

Reports.net を正式にご利用頂く場合は、ライセンスを購入して頂く必要があります。ライセンス形態及び代金支払方法は以下のとおりです。

- 必要なライセンス数の数え方
  - Reports.net をインストールするパソコンの台数。
    ※Linux 上 / クラウド Windows 上で利用するライセンス(後述)を除く
- 1 ライセンス当たりの価格
  - エンジン + デザイナー.....80,000 円（消費税込 88,000 円）
    - ✧ 本価格には消費税を含みません。
    - ✧ バグフィックス等のバージョンアップは原則として無償とさせて頂きます。
    - ✧ 大幅な機能追加等によるバージョンアップの場合には別ライセンスとさせていただく場合がございます。

    本価格は Reports.net の使用権に対するものです。カスタマイズや保守等の費用は一切含まれておりません。

- お支払方法
  - (88,000 円×ライセンス数)を下記口座へお振り込み下さい。

銀行名	支店名 (コード)	口座番号	名義
三菱ＵＦＪ銀行	新宿支店 (341)	普通 3831891	ユ）パオアツトオフイス
PayPay 銀行	すずめ支店 (002)	普通 6461359	ユ）パオアツトオフイス

郵便口座番号	名義
００１５０－０－ ５７６８４５	有限会社 パオ・アット・オフィス

    - ✧ 手数料は利用者様負担でお願い致します。

- お支払いの通知と製品の送付
  - 振り込みが完了した時点で、必ず弊社 WEB サイトの「入金連絡フォーム」から入金のご連絡をお願いいたします。
    https://www.pao.ac/reports.net/buy.html#form
  - 弊社では上記連絡を受けて入金確認を行い、ライセンスキーを利用者様へ電子メールに添付して送付させて頂きます。
    - ✧ 利用者様へは電子メール以外でのライセンスキー送付は原則として行いません。
    - ✧ ライセンスキーは原則として再送付致しません。受信したライセンスキーは必ずバックアップを取る等して、内容を大切に保管して頂くようお願いします。

- Linux ライセンス
  ランタイムライセンスは、Linux/ Azure を除き無償です。
  Linux 上で利用する場合に限り、有償となります。
  Linux の 1 URL(サイト・クラウド)につき 1 Linux ライセンスが必要です。
  1 Linux ライセンスの価格は、税抜 7,000 円(税込 7,700 円)です。
  Linux ライセンスは、次に該当するお客様のみご購入いただけます。
  - 通常ライセンス(開発ライセンス)とセットでご購入いただくお客様
  - 既に通常ライセンス(開発ライセンス)をご購入いただいているお客様

- Azure ライセンス
  名前が Azure ライセンスとなっておりますが、実際には、Azure / AWS / GCP 等のクラウド **Windows サーバ**上で利用する場合のライセンスとなります。
  クラウド **Windows サーバ**上 Linux ライセンスでも動作はします。
  詳細の説明につきましては、こちら(クリック)をご覧ください。

  ランタイムライセンスは、Linux/ Azure を除き無償です。
  Azure / AWS / GCP 等のクラウド **Windows サーバ**上で利用する場合に限り、有償となります。
  クラウド **Windows サーバ**の 1 URL(サイト・クラウド)につき 1 Azure ライセンスが必要です。
  1 Azure ライセンスの価格は、税抜 7,000 円(税込 7,700 円)です。
  Azure ライセンスは、次に該当するお客様のみご購入いただけます。
  - 通常ライセンス(開発ライセンス)とセットでご購入いただくお客様
  - 既に通常ライセンス(開発ライセンス)をご購入いただいているお客様

- 見積書/納品書/請求書/領収証の発行、納品後のお支払いについて
  見積書/納品書/請求書/領収証の発行は可能でございます。本製品納品後のお支払いも可能でございます。
  https://www.pao.ac/reports.net/buy.html
  上記サイトでの手続きにより、弊社からの見積書/納品書/請求書/領収証の発行、及び、納品後のお客様からのお支払いを行えるようになっております。

変更履歴

版	作成日	変更点
1	2003.05.27	新規作成
2	2003.06.04	サンプル内容修正 Reports.net における各オブジェクトの特徴の追加
3	2005.05.10	QR コード対応による QR コードの記述追加
4	2006.08.01	ZIP・SVG・SVGZ 対応 プロパティ追加（プリントダイアログ etc） Reports.jar 追加
5	2007.02.22	コンビニバーコード追加 バーコードの描画方法(直接描画)追加
6	2008.10.10	Reports.jar で PDF 出力が可能になったことに伴うマニュアル変更
7	2009.02.15	バーコードの黒バーの幅ドット単位微調整機能追加
8	2010.11.04	ライセンス形態変更
9	2011.02.28	・バーコードの白バーの太さをドット単位で調整できる機能追加による説明を追加 ・デザイン時のオブジェクトのプロパティ値を実行時に取得・変更できる機能追加による説明を追加
10	2012.11.17	製品価格改定
11	2012.04.01	消費税アップによる税込価格変更
12	2014.11.04	UPC-A/UPC-E バーコード実装による説明追加。 GS1-128 バーコードの AI 識別子挿入方法説明追加。 XPS 出力機能追加による説明追加。
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
15	2016.04.01	GS1 Databar (RSS) 追加に伴う変更 バーコード全般の種類に関する記述追加 目次の階層を詳細に
16	2018.05.10	動作条件に、.net framework 4.6～4.7.2 / Visual Studio 2015 / 2017 を追加
17	2019.10.01	消費税 10%変更に伴う価格変更

版	作成日	変更点
18	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
19	2022.10.24	QR コードのデフォルトエンコード文字が shift_jis から utf8 に変更になったことに伴う文書変更 .NET 7 / .net frame work 4.8.1 対応に伴う追記
20	2023.07.17	Reports.net WPF 版と.NET MAUI (WinUI 3)版の説明追加
21	2024.05.09	WPF 版正式対応。説明追加。
22	2026.02.19	SVG 出力機能を大幅に強化。 GetSvg メソッド追加（インライン SVG 埋め込み HTML 出力）。 ブラウザのみで帳票プレビュー・印刷が可能に。 GetSvgTag メソッド追加（指定ページの SVG タグ取得）