Web クライアント印刷

1. 概要

Web クライアント印刷とは、インターネットやイントラネットの Web システムにおいて、Web ブラウザーで印刷処理を行うことです。
この「Web ブラウザーで印刷処理」を行うにあたっては、以下に説明する印刷方法があります。

① PDF 表示印刷
② PDF 非表示印刷
③ブラウザー任意指定印刷

構築するシステムの用途に合わせて、いずれかの方法で Web クライアント印刷処理を実装します。印刷方法の②と③については製品に付属の印刷コントロールを利用します。

Web クライアント印刷では以下の Web ブラウザーが利用できます。

・Google Chrome
・Internet Explorer 11
・Microsoft Edge

<< 参考 >>
Web ブラウザーからの指示で印刷する場合でも、サーバーから LAN 上にあるプリンター（または、IPP プロトコルでインターネット上にあるプリンター）に印刷する場合は、Web クライアント印刷ではなくサーバー印刷となりますので、PrintStage ランタイム、または Print ランタイムの印刷処理で行ってください。

以降は Google Chrome を Chrome、Internet Explorer 11 を IE、Microsoft Edge を Edge と表記します。

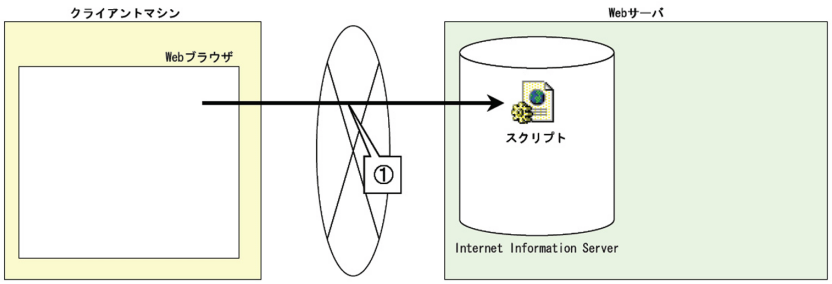
2. PDF 表示印刷とは

PDF ファイルを表示して表示画面から Adobe Acrobat や Adobe Acrobat Reader の機能で印刷する方法です。
この印刷方法は、サーバー側で作成された PDF ファイルを、Web ブラウザーのプラグイン機能で Adobe Acrobat や Adobe Acrobat Reader を起動して表示、印刷するだけですので、特別な処理を実装する必要はありません。

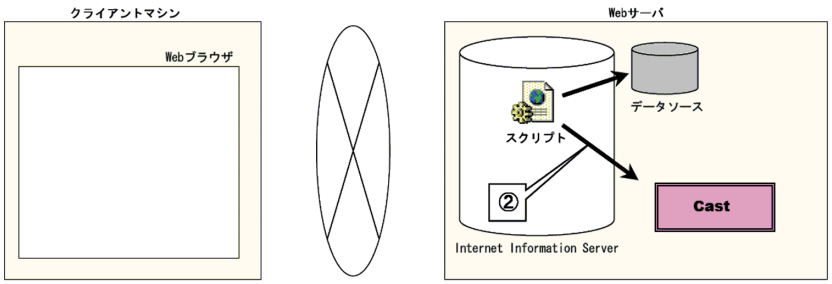
<< 注意 >>
印刷元となるデータは PDF ファイルとなりますので、サーバー側では PDF ファイルを生成する Cast ランタイムが必要です。

<< 注意 >>
Adobe Acrobat や Adobe Acrobat Reader DC 以上がクライアントマシンにインストールされている必要があります。

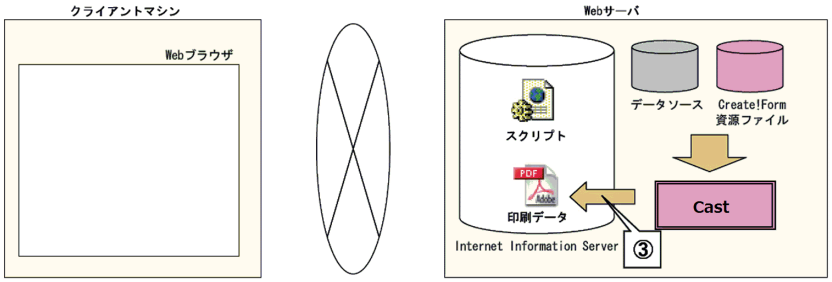
①クライアントは、Cast ランタイムを起動するスクリプトにアクセスします。



②リクエストを受けたスクリプトは、ランタイムの実行に必要なデータソースを参照、あるいは作成し、ランタイムを実行します。

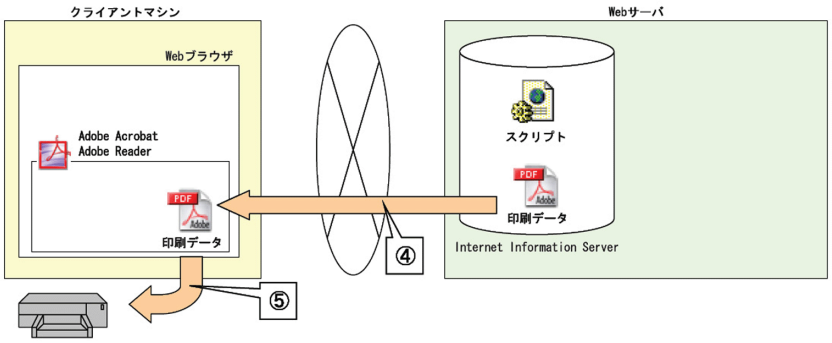


③ランタイムは、サーバー上のデータソースと資源ファイルから PDF ファイルを作成します。



④生成された PDF ファイルは、Web ブラウザーを通じてクライアント側に表示されます。

⑤ Web ブラウザーに表示された PDF ファイルを Adobe Acrobat や Adobe Acrobat Reader の印刷メニューから印刷します。



また Web ブラウザーに関しては、Adobe Acrobat や Adobe Acrobat Reader のアドインをサポートする Web ブラウザーでなければなりません。

以上が PDF 表示印刷の基本的な流れになります。

3.PDF 非表示印刷とは

Web ブラウザーでの印刷指示により、PDF ファイルを非表示（表示も可能）にして、あらかじめ指定されたプリンターへ印刷する方法です。
この非表示での印刷方法は、サーバー側で作成された PDF ファイルを表示せずに Windows 環境で「通常使うプリンター」に設定されたプリンターへ印刷する場合に利用します。
給紙選択などの印刷情報は「通常使うプリンター」で設定された内容が使用されます。
この処理を実装するには、製品に付属の印刷コントロールを使用する必要があります。

<< 注意 >>
印刷元となるデータは PDF ファイルとなりますので、サーバー側では PDF ファイルを生成する Cast ランタイムが必要です。
また、Web クライアント製品をクライアント環境にインストールする必要があります。
インストール方法の詳細はインストールマニュアルの「第 5 章 Windows 製品 [Web クライアント製品]」をご覧ください。

<< 注意 >>
Adobe Acrobat や Adobe Acrobat ReaderDC 以上がクライアントマシンにインストールされている必要があります。

<< 注意 >>
Adobe Acrobat や Adobe Acrobat Reader の初回起動時は使用許諾契約書の確認画面が表示されます。確認画面では [同意する] を選択し、あらかじめ利用可能な状態としておく必要があります。同意していない場合は正常に印刷が行われません。なお、使用許諾契約書の確認画面はユーザー単位で表示されることがあります。

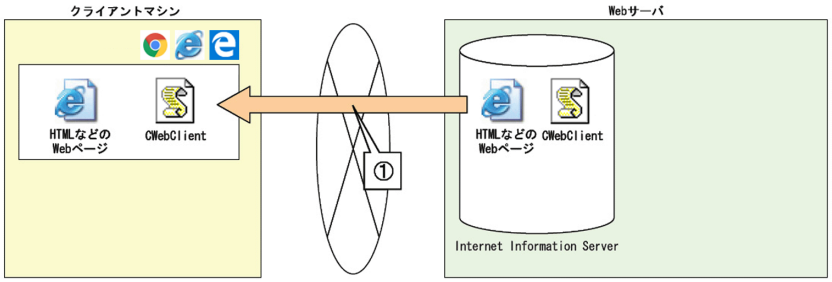
<< 注意 >>
Adobe Acrobat はライセンス認証を行い、あらかじめ利用可能な状態としておく必要があります。ライセンス認証していない場合は正常に印刷が行われません。

<< 注意 >>
環境設定にある「セキュリティ（拡張）」の「サンドボックスによる保護」の「保護されたビュー」は ″ オフ ″ を選択してください。″ オフ ″ 以外を選択した場合は正常に印刷が行われません。

<< 注意 >>
PDF 非表示印刷の処理中は Adobe Acrobat や Adobe Acrobat Reader の起動や操作は行わないでください。起動や操作を行うと正常に印刷が行われません。

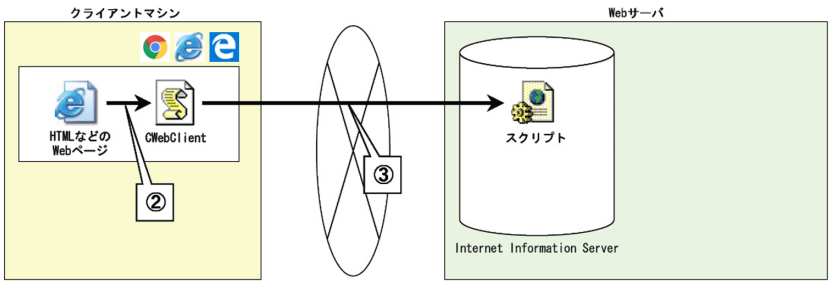
印刷コントロールは JavaScript 形式のため、Chrome、IE、Edge に対応しています。

①script タグで印刷コントロールを参照している Web ページをクライアントが Web ブラウザーで参照すると、印刷コントロールが読み込まれます。

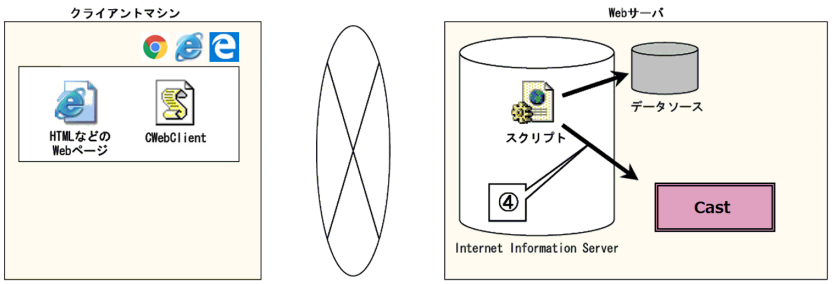


②その Web ページから印刷要求を発生させます。
例えば、Web ページ上のボタンをクリックした場合に、印刷コントロールの API を実行します。

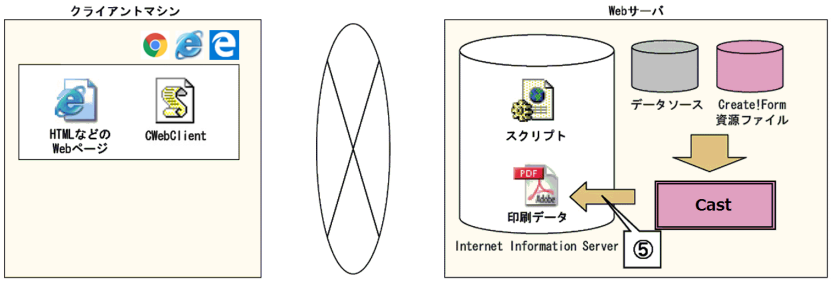
③印刷コントロールは設定されたプロパティ値を元に、Web サーバー上のスクリプトにアクセスします。



④リクエストを受けたスクリプトは、ランタイムの実行に必要なデータソースを参照、あるいは作成し、ランタイムを実行します。

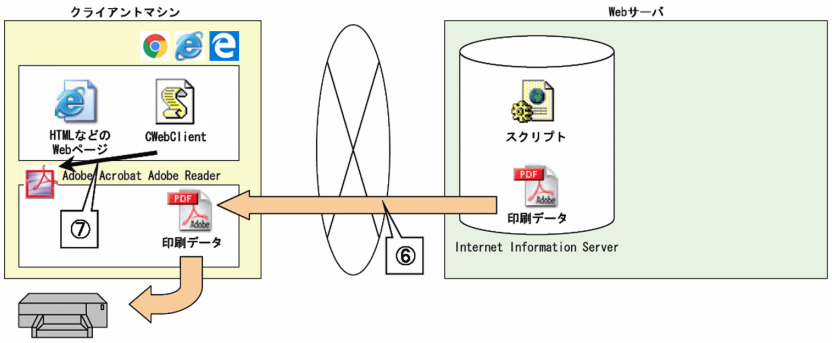


⑤ランタイムは、サーバー上のデータソースと資源ファイルから PDF ファイルを作成します。



⑥印刷コントロールは、生成された PDF ファイルをクライアントにダウンロードします。

⑦印刷コントロールは、Adobe Acrobat や Adobe Acrobat Reader を起動して、ダウンロードした PDF ファイルに対して印刷処理を実行します。



以上が PDF 非表示印刷の基本的な流れになります。

4. ブラウザー任意指定印刷とは

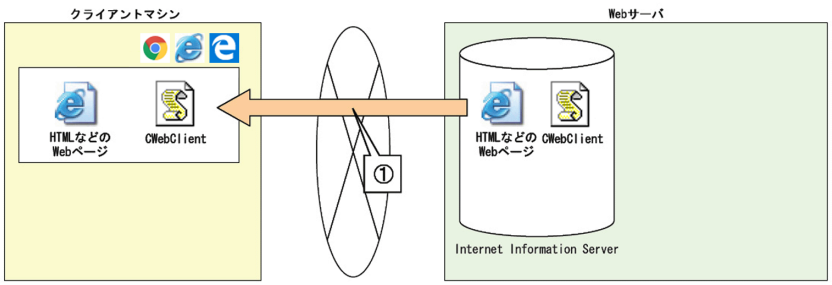
Web ブラウザーでの印刷指示により任意のプリンターへ任意の印刷情報で印刷する方法です。
この印刷方法は、サーバー側で作成された印刷データを Windows に登録されているプリンターへ印刷する場合に利用します。
プリンターが複数の場合にはプリンターの設定、給紙選択などの印刷情報を設定し印刷することができます。
この処理を実装するには、製品に付属の印刷コントロールを使用する必要があります。

ブラウザー任意指定印刷では、サーバーで生成された CCD ファイルを印刷コントロールでダウンロードして印刷処理を行います。
CCD ファイルとは、Create!Form 資源ファイルやデータソースといった帳票イメージの〝素〟を圧縮し、一つにまとめたものです。
この CCD ファイルを解凍して印刷するには、PrintStageWeb Client ランタイムが必要となります。

<< 注意 >>
ブラウザー任意指定印刷を行うにあたり、サーバー側では PrintStageWeb ランタイム、クライアント側では PrintStageWeb Client ランタイムが必要です。

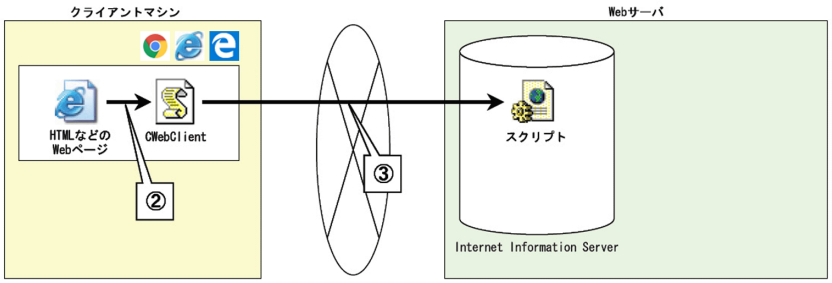
印刷コントロールは JavaScript 形式のため、Chrome、IE、Edge に対応しています。

①script タグで印刷コントロールを参照している Web ページをクライアントが Web ブラウザーで参照すると、印刷コントロールが読み込まれます。

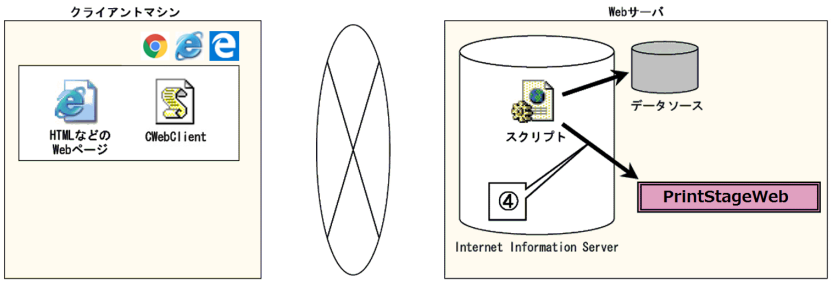


②その Web ページから印刷要求を発生させます。
例えば、Web ページ上のボタンをクリックした場合等に、印刷コントロールの API を実行させることが可能です。

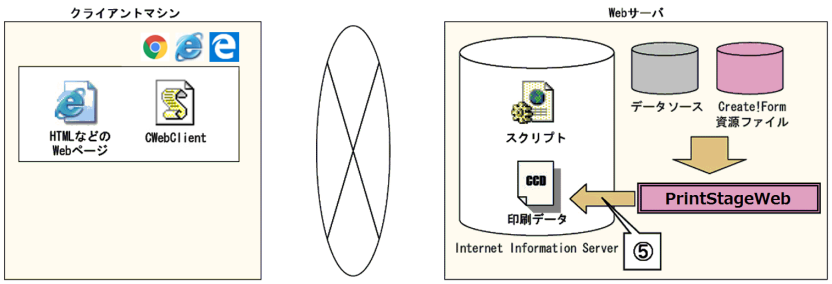
③印刷コントロールは設定されたプロパティ値を元に、Web サーバー上のスクリプトにアクセスします。



④リクエストを受けたスクリプトは、ランタイムの実行に必要なデータソースを参照、あるいは作成し、ランタイムを実行します。

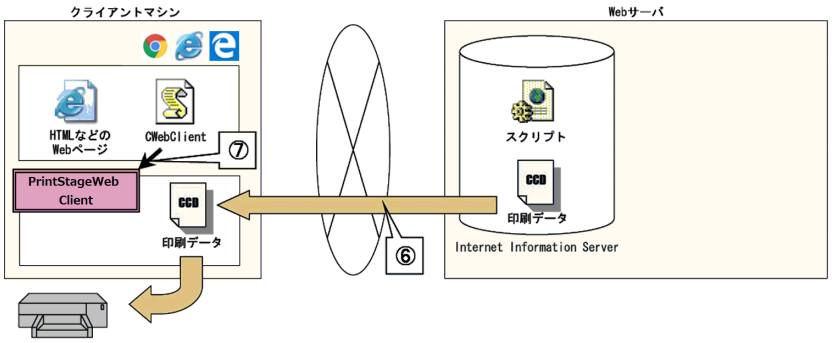


⑤ランタイムは、サーバー上のデータソースと資源ファイルから CCD ファイルを作成します。



⑥印刷コントロールは、生成された CCD ファイルをクライアントにダウンロードします。

⑦印刷コントロールは、PrintStageWeb Client ランタイムを起動し、ダウンロードした CCD ファイルを使用して、指定されたプリンターへ印刷を実行します。



以上がブラウザー任意指定印刷の基本的な流れになります。

PrintStageWeb Client ランタイムは、クライアントに保存されている印刷情報を反映して印刷します。
例えば、帳票 "A" はプリンター "B" に、帳票 "C" はプリンター "D" に印刷するという設定を保存することが可能なため、帳票によって複数のプリンターを使い分けることが可能です。
設定できる印刷情報は以下の通りです。

・出力するプリンター
・給紙トレイ
・印刷部数
・両面印刷
・拡大 / 縮小
・解像度
・出力位置の補正

<< 参考 >>
CCD ファイルはクライアントに保存することも可能です。
これにより、サーバー側は印刷要求を受けた際、クライアントには固定のデータ（CCD ファイル）が保存されているため、可変のデータのみ送信するという処理が実現でき、トラフィック量はさらに軽減されることになります。

5.Web サーバーの基本設定

サーバーマシンの構成

Web サーバーの設定を行うにあたり、CentOS 7 + Apache Tomcat 9 + Java Servlet を使用して説明します。

ここでは、印刷方法ごとに対応したランタイム製品が事前に導入され、正常に動作しているものとします。Web サーバーのホスト名としては、"testsv" を使用しています。
必要に応じて、HTML ファイルやスクリプトに記述する URL を変更してください。

コンテンツの配置

ここでは以下のようにファイルを配置します。

```
/usr/local/tomcat/webapps/
    + createform/
        + index.html                HTML ファイル（※1）
        + lib/
            + cwebclient.min.js     印刷コントロール（※3）
        + printings/                印刷データ生成ディレクトリ（※4）
        + WEB-INF/
            + web.xml
            + lib
                + CreateFormLib.jar JavaAPI（※5）
            + classes/
                + RunServlet.class  ランタイム実行クラス（※2）
/opt/resource/
    + cwork/                        ランタイム作業ディレクトリ
        + datamap/
        + form/
        + style/
    + data/                         テキストファイルディレクトリ
        + test.csv                  ランタイム実行に使用するデータソース
```

※1:HTML ファイルは、印刷要求を発行する Web ページになりますので、各印刷方法によって内容は変わります。詳細は後述の構築例をご覧ください。
※2: ランタイム実行スクリプトは、各印刷方法によって実行するランタイムが異なります。「PDF 表示印刷」、「PDF 非表示印刷」であれば Cast ランタイム、「ブラウザー任意指定印刷」であれば PrintStageWeb ランタイムをスクリプトで起動します。詳細は後述の構築例をご覧ください。
※3:cwebclient.min.js は印刷コントロールのファイル名です。ランタイム導入ディレクトリの「lib」ディレクトリに配置されていますので、「/usr/local/tomcat/webapps/createform/lib」に転送してください。ただし、印刷コントロールを使用しない「PDF 表示印刷」においては、cwebclient.min.js を配置する必要はありません。
※4: 印刷データ生成ディレクトリには、各ランタイムが生成するファイル（PDF ファイル、CCD ファイル）が格納されます。
※5:CreateFormLib.jar はランタイム呼び出し API です。ランタイム導入ディレクトリの「lib」ディレクトリに配置されています。

Java Servlet のマッピング

Web ブラウザーから Java Servlet にアクセスできるようにするため web.xml ファイルから URL とクラスのマッピングを行います。

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
    http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"
    version="3.1">

    <servlet>
        <servlet-name>RunServlet</servlet-name>
        <servlet-class>RunServlet</servlet-class>
    </servlet>

    <servlet-mapping>
        <servlet-name>RunServlet</servlet-name>
        <url-pattern>/run</url-pattern>
    </servlet-mapping>

</web-app>
```

6. クライアントの基本設定

6-1.Web クライアント製品のインストール

PDF 非表示印刷またはブラウザー任意指定印刷を利用する場合、Web クライアント製品をインストールする必要があります。
インストール方法の詳細は、インストールマニュアル - 第 5 章 Windows 製品 [Web クライアント製品] をご覧ください。

7. PDF 表示印刷の構築例

① "index.html" の作成

```
<!DOCTYPE html>
<html lang="ja">
<head>
<meta charset="UTF-8">
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
<title>PDF 表示印刷 </title>
<script type="text/javascript">
function ShowPdf() {
   window.open("http://testsv:8080/createform/run?OutputFileName=test.pdf",
   "_blank");
}
</script>
</head>
<body>
   <input type="button" value="PDF 表示印刷 " onclick="ShowPdf()"/>
</body>
</html>
```

"index.html" は、クライアントが Web ブラウザーに読み込んで、印刷要求を発行する Web ページになります。
"PDF 表示印刷 " というボタンをクリックすることで、JavaScript の "ShowPdf" という関数を実行します。"ShowPdf" 関数は、Cast ランタイムを実行する Java Servlet を呼び出すためのものです。これにより、Web ブラウザー上から "PDF 表示印刷 " ボタンをクリックすることで、サーバーに対して PDF 生成要求を発行することが可能です。
"index.html" は /usr/local/tomcat/webapps/createform ディレクトリ直下に配置します。

<< 注意 >>

open 関数の第 1 引数に指定している URL の GET 文字列 "OutputFileName=test.pdf" は、サーバーの Cast ランタイムが生成する PDF ファイルの名前として、Java Servlet に渡しています。
このままでは、どのクライアントからのアクセスでも "test.pdf" という PDF ファイルを生成してしまうため、アクセス状況によってはクライアントが印刷処理を完了する前に、PDF ファイルが上書きされてしまう可能性があります。
従って、実際には PDF ファイル名は、クライアントごとにユニークな文字列を指定する必要がありますので注意してください。

② "RunServlet.java" の作成

```
import java.io.*;
import java.nio.file.*;

import javax.servlet.*;
import javax.servlet.http.*;

import net.createform.cji.*;
import net.createform.common.*;

public class RunServlet extends HttpServlet {
   @Override
   protected void doGet(HttpServletRequest req, HttpServletResponse res)
            throws ServletException, IOException {
            // 作業ディレクトリ
            File workDir = new File("/opt/resource/cwork");
            // ジョブファイル名
            String jobFileName = "SchoolLife.sty";
            // 出力 PDF ファイル
            File outputFile = new File(
                     "/usr/local/tomcat/webapps/createform/printings",
                     req.getParameter("OutputFileName")
            );
            // 入力データファイル
            File inputFile = new File("/opt/resource/data", "test.csv");

            RuntimeParam param = new RuntimeParam();
            param.setWorkDir(workDir.getPath());
            param.setStyleFile(jobFileName);
            param.setOutFile(outputFile.getPath());
            param.setDataFile(inputFile.getPath());

            CCast cast = new CCast();
            cast.executeRuntime(param);

            OutputStream out = res.getOutputStream();
            out.write(Files.readAllBytes(outputFile.toPath()));
   }
}
```

"RunServlet.java" は、Cast ランタイムを呼び出す Java Servlet クラスです。
Cast ランタイムは JavaAPI を利用して呼び出しています。
"index.html" から GET 文字列で渡された "OutputFileName" を使用して PDF ファイル名を設定していることが確認できます。Cast ランタイム実行後、生成された PDF ファイルを HTTP レスポンスに書き出しています。これにより、クライアントは PDF ファイルを Web ブラウザーで表示できることになります。
"RunServlet.java" をコンパイル後に生成される "RunServlet.class" を /usr/local/tomcat/webapps/createform/WEB-INF/classes ディレクトリ直下に配置します。

③クライアントからの実行

それでは、実際に Web クライアント印刷を実行します。

Web ブラウザーから "http://testsv:8080/createform/index.html" にアクセスします。Web ブラウザーに表示された "PDF 表示印刷 " ボタンをクリックします。新しいウィンドウが起動し、Java Servlet にアクセスします。サーバー側で Cast ランタイムの実行が終了すると、作成された PDF ファイルが HTTP レスポンスで書き出されるので、クライアント側に PDF ファイルが表示されます。

※ Adobe Acrobat や Adobe Acrobat Reader がインストールされており、アドイン機能または PDF ファイルの描画が有効な Web ブラウザーを使用している場合のみです。

Adobe Acrobat や Adobe Acrobat Reader の [印刷] ボタンをクリックして印刷処理を実行します。

以上で、PDF 表示印刷による Web クライアント印刷のサンプル実行は終了です。

8. PDF 非表示印刷の構築例

① "index.html" の作成

```
<!DOCTYPE html>
<html lang="ja">
<head>
<meta charset="UTF-8">
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
<title>PDF 非表示印刷 </title>
<script type="text/javascript" src="http://testsv:8080/createform/lib/
cwebclient.min.js?version=12.0.0" charset="utf-8"></script>
<script type="text/javascript">
function KickPrintExecutePdf() {
   cwebclient.CreatePrintingDataScriptUrl = "http://testsv:8080/createform/run";
   cwebclient.PrintExecutePdf();
}
</script>
</head>
<body>
   <input type="button" value="PDF 非表示印刷 " onclick="KickPrintExecutePdf()"/>
</body>
</html>
```

"index.html" は、クライアントが Web ブラウザーに読み込んで、印刷要求を発行する Web ページになります。
印刷コントロールをクライアントマシンの Web ブラウザーから参照するため、script タグが記述されています。
各属性値の意味は以下の通りです。

type:
印刷コントロールは JavaScript 形式のため「text/javascript」を指定します。
src:
印刷コントロールが配置されている URL を指定します。URL に続けてバージョンを識別するためのクエリー文字列を「version=xx.x.x」の形式で指定します。
charset:
印刷コントロールは文字コードが UTF-8 のため「utf-8」を指定します。

"PDF 非表示印刷 " というボタンをクリックすることで、JavaScript の "KickPrintExecutePdf" という関数を実行します。"KickPrintExecutePdf" 関数は、印刷コントロールのプロパティやメソッドを実行する役割を果たします。

ここで使用されている印刷コントロールのプロパティ、メソッドについて簡単に説明します。
プロパティ「CreatePrintingDataScriptUrl」は、Cast ランタイムを実行するスクリプトの URL を指定します。今回は、Java Servlet から Cast ランタイムを実行するので、"http://testsv:8080/createform/run" を指定することになります。メソッド「PrintExecutePdf」は、印刷処理を実行するメソッドです。

メソッド「PrintExecutePdf」が呼び出されたタイミングで、プロパティ「CreatePrintingDataScriptUrl」で指定されたURLにアクセスし、PDFファイルをダウンロード、印刷実行までを行います。処理結果に応じたステータスコード（数値）と文字列がダイアログに表示されます。戻り値が"0"の場合は、正常終了を表します。
※印刷コントロールの詳細は、「10. 印刷コントロール API 仕様」をご覧ください。
"index.html"は/usr/local/tomcat/webapps/createformディレクトリ直下に配置します。

② "RunServlet.java" の作成

```
import java.io.*;
import java.nio.file.*;

import javax.servlet.*;
import javax.servlet.http.*;

import net.createform.cji.*;
import net.createform.common.*;

public class RunServlet extends HttpServlet {
   @Override
   protected void doPost(HttpServletRequest req, HttpServletResponse res)
            throws ServletException, IOException {
            // 作業ディレクトリ
            File workDir = new File("/opt/resource/cwork");
            // ジョブファイル名
            String jobFileName = "SchoolLife.sty";
            // 出力PDFファイル
            File outputFile = new File(
                     "/usr/local/tomcat/webapps/createform/printings",
                     req.getParameter("OutputFileName")
            );
            // 入力データファイル
            File inputFile = new File("/opt/resource/data", "test.csv");

            RuntimeParam param = new RuntimeParam();
            param.setWorkDir(workDir.getPath());
            param.setStyleFile(jobFileName);
            param.setOutFile(outputFile.getPath());
            param.setDataFile(inputFile.getPath());

            CCast cast = new CCast();
            cast.executeRuntime(param);

            OutputStream out = res.getOutputStream();
            out.write(Files.readAllBytes(outputFile.toPath()));
   }
}
```

"RunServlet.java" は、Cast ランタイムを呼び出す Java Servlet クラスです。
Cast ランタイムは JavaAPI を利用して呼び出しています。
変数 "outputFile" に注目してください。
リクエストパラメーターの "OutputFileName" から PDF ファイル名を取得しています。
"OutputFileName" は、印刷コントロールが Java Servlet に対して自動で送信したものです。
PDF ファイル名を固定なものにしてしまうと、複数のクライアントからアクセスされた場合、PDF ファイルが上書きされてしまう可能性があります。その問題を解決するためには、生成する PDF ファイルにユニークなファイル名を付けなければなりません。そこで印刷コントロールでは、ユニークな文字列を作成し、それを "OutputFileName" で送信しています。また印刷コントロールは "OutputFileName" で送信した PDF ファイル名を印刷対象の PDF ファイルと認識します。
以上の理由から、特別な状況がない限り、このファイル名を使用して PDF ファイルを生成することを推奨します。

※今回は "OutputFileName" を POST 文字列で送信（POST メソッドによる送信）しましたが、印刷コントロールのプロパティ「DataSendMethod」に "GET" を指定することで、GET 文字列として送信（GET メソッドによる送信）することも可能です。
※印刷コントロールの詳細は、「10. 印刷コントロール API 仕様」をご覧ください。

Cast ランタイム実行後、生成された PDF ファイルを HTTP レスポンスに書き出しています。これにより、印刷コントロールは PDF ファイルを受け取り、クライアント側で印刷処理を実行することになります。"RunServlet.java" をコンパイル後に生成される "RunServlet.class" を /usr/local/tomcat/webapps/createform/WEB-INF/classes ディレクトリ直下に配置します。

③クライアントからの実行
それでは、実際に Web クライアント印刷を実行します。
Web ブラウザーから "http://testsv:8080/createform/index.html" にアクセスします。Web ブラウザーに表示された "PDF 非表示印刷" ボタンをクリックします。印刷コントロールが起動し、Java Servlet にアクセスします。
サーバー側で Cast ランタイムの実行が終了すると、作成された PDF ファイルが HTTP レスポンスで書き出され、印刷コントロールは PDF ファイルをダウンロードします。
印刷コントロールは、PDF ファイルのダウンロードを終了すると、Adobe Acrobat や Adobe Acrobat Reader を起動し、PDF ファイルを使用して印刷処理を実行します。
印刷処理は「通常使うプリンター」に設定されているプリンターに対して実行されます。
印刷情報も「通常使うプリンター」で設定されているデフォルトの設定値が使用されます。
※ Adobe Acrobat や Adobe Acrobat Reader がインストールされている必要があります。

<< 注意 >>
印刷設定は、PDF 設定の印刷ダイアログプリセットで設定できます。
また、印刷ダイアログプリセットの設定を行わない場合は、すでに設定されている Adobe Acrobat や Adobe Acrobat Reader の印刷設定が適用されます。印刷する PDF ファイルのフォームサイズとプリンターで選択されている用紙サイズが異なる場合、印刷ダイアログプリセットの設定を行うか、あらかじめ以下の操作を行っておくことにより、用紙サイズを合わせて印刷することが可能です。

〈Acrobat DC の場合〉
Acrobat の［印刷］ダイアログボックスにある［ページサイズ処理］の［PDF のページサイズに合わせて用紙を選択］チェックボタンを ON にします。

以上で、PDF 非表示印刷による Web クライアント印刷のサンプル実行は終了です。
※印刷コントロールの詳細は、「10. 印刷コントロール API 仕様」をご覧ください。

9. ブラウザー任意指定印刷の構築例

① "index.html" の作成

```
<!DOCTYPE html>
<html lang="ja">
<head>
<meta charset="UTF-8">
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
<title>ブラウザー任意指定印刷</title>
<script type="text/javascript" src="http://testsv:8080/createform/lib/
cwebclient.min.js?version=12.0.0" charset="utf-8"></script>
<script type="text/javascript">
function KickPrintExecuteEx() {
   cwebclient.CreatePrintingDataScriptUrl = "http://testsv:8080/createform/run";
   // 帳票名（Job ファイル名）を引数に指定します。
   cwebclient.PrintExecuteEx("SchoolLife");
}
</script>
</head>
<body>
   <input type="button" value="ブラウザー任意指定印刷"
          onclick="KickPrintExecuteEx()" />
</body>
</html>
```

"index.html" は、クライアントが Web ブラウザーに読み込んで、印刷要求を発行する Web ページになります。
印刷コントロールをクライアントマシンの Web ブラウザーから参照するため、script タグが記述されています。各属性値の意味は以下の通りです。

type:
印刷コントロールは JavaScript 形式のため「text/javascript」を指定します。
src:
印刷コントロールが配置されている URL を指定します。URL に続けてバージョンを識別するためのクエリー文字列を「version=xx.x.x」の形式で指定します。
charset:
印刷コントロールは文字コードが UTF-8 のため「utf-8」を指定します。

"ブラウザー任意指定印刷"というボタンをクリックすることで、JavaScript の "KickPrintExecuteEx" という関数を実行します。"KickPrintExecuteEx" 関数は、印刷コントロールのプロパティやメソッドを実行する役割を果たします。

ここで使用されている印刷コントロールのプロパティ、メソッドについて簡単に説明します。
プロパティ「CreatePrintingDataScriptUrl」は、PrintStageWeb ランタイムを実行するスクリプトの URL を指定します。今回は、Java Servlet から PrintStageWeb ランタイムを実行するので、"http://testsv:8080/createform/run" を指定することになります。

メソッド「PrintExecuteEx」は、印刷処理を実行するメソッドです。
引数に指定する文字列は、サーバーで生成する帳票の帳票名（Job ファイル名）です。
メソッド「PrintExecuteEx」が呼び出されたタイミングで、プロパティ「CreatePrintingDataScriptUrl」で指定されたスクリプトにアクセスし、CCD ファイルをダウンロード、印刷実行までを行います。処理結果に応じたステータスコード（数値）と、文字列がダイアログに表示されます。戻り値が"0"の場合は、正常終了を表します。
※印刷コントロールの詳細は、「10. 印刷コントロール API 仕様」をご覧ください。
"index.html"は /usr/local/tomcat/webapps/createform ディレクトリ直下に配置します。

② "RunServlet.java" の作成

```
import java.io.*;
import java.nio.file.*;

import javax.servlet.*;
import javax.servlet.http.*;

import net.createform.common.*;
import net.createform.sji.*;

public class RunServlet extends HttpServlet {
   @Override
   protected void doPost(HttpServletRequest req, HttpServletResponse res)
            throws ServletException, IOException {
            // 作業ディレクトリ
            File workDir = new File("/opt/resource/cwork");
            // ジョブファイル名
            String jobFileName = req.getParameter("StyleName") + ".sty";
            // 出力 PDF ファイル
            File outputFile = new File(
                     "/usr/local/tomcat/webapps/createform/printings",
                     req.getParameter("OutputFileName")
            );
            // 帳票資源データ設定
            String ccdFileOption = req.getParameter("CcdFileOption");
            // 入力データファイル
            File inputFile = new File("/opt/resource/data", "test.csv");

            RuntimeParam param = new RuntimeParam();
            param.setWorkDir(workDir.getPath());
            param.setStyleFile(jobFileName);
            param.setOutFile(outputFile.getPath());
            param.setStageOption(ccdFileOption);
            param.setDataFile(inputFile.getPath());

            CPrintSTCompress compress = new CPrintSTCompress();
            compress.executeRuntime(param);

            OutputStream out = res.getOutputStream();
            out.write(Files.readAllBytes(outputFile.toPath()));
   }
}
```

"RunServlet.java" は、PrintStageWeb ランタイムを呼び出す Java Servlet クラスです。
PrintStageWeb ランタイムは JavaAPI を利用して呼び出しています。
変数 "outputFile" に注目してください。リクエストパラメーターの "OutputFileName" から

CCD ファイル名を取得しています。"OutputFileName" は、印刷コントロールが Java Servlet に対して自動で送信したものです。CCD ファイル名を固定なものにしてしまうと、複数のクライアントからアクセスされた場合、CCD ファイルが上書きされてしまう可能性があります。その問題を解決するためには、生成する CCD ファイルにユニークなファイル名を付けなければなりません。そこで印刷コントロールでは、ユニークな文字列を作成し、それを "OutputFileName" で送信しています。また印刷コントロールは "OutputFileName" で送信した CCD ファイル名を印刷対象ファイルと認識します。
以上の理由から、特別な状況がない限り、このファイル名を使用して CCD ファイルを生成することを推奨します。

※今回は "OutputFileName" を POST 文字列で送信（POST メソッドによる送信）しましたが、印刷コントロールのプロパティ「DataSendMethod」に "GET" を指定することで、GET 文字列として送信（GET メソッドによる送信）することも可能です。
以下に紹介される "StyleName"、"CcdFileOption" に関しても同様です。
※印刷コントロールの詳細は、「10. 印刷コントロール API 仕様」をご覧ください。

コマンドの作成において、Job ファイルの指定を POST 文字列から "StyleName" を取得して指定しています。"StyleName" から取得する文字列は、"index.html" において印刷コントロールのメソッド「PrintExecuteEx」の引数に指定されたものです。これは、クライアントが印刷要求している帳票の種類とサーバーで出力する帳票の種類を確実に合わせるためです。
「ブラウザー任意指定印刷」においては、クライアント側で印刷する帳票ごとに印刷情報を管理しているので、クライアントの要求する帳票をサーバーで出力しなければなりません。
従って、"StyleName" で取得した文字列を Job ファイルに指定することで、この問題を解決することが可能です。またコマンドの作成で、POST 文字列から "CcdFileOption" を取得し、"-c" オプションの引数に指定します。"-c" オプションは、PrintStageWeb ランタイムが出力する CCD ファイルの形式を設定するものです。"-c" オプションの値は、クライアントの要求に合わせて指定します。印刷コントロールは、クライアントの印刷情報を参照して、クライアントの要求に合わせた "-c" オプションの値を、"CcdFileOption" として送信します。
従って、スクリプトでは "CcdFileOption" を受信して、そのまま "-c" オプションに指定しなければなりません。
※ "CcdFileOption" と "-c" オプションの詳細は、「19.CCD ファイル保存機能」をご覧ください。
PrintStageWeb ランタイム実行後、生成された CCD ファイルを HTTP レスポンスに書き出しています。これにより、印刷コントロールは CCD ファイルを受け取り、クライアント側で印刷処理を実行することになります。"RunServlet.java" をコンパイル後に生成される "RunServlet.class" を /usr/local/tomcat/webapps/createform/WEB-INF/classes ディレクトリ直下に配置します。

③クライアントからの実行
それでは、実際に Web クライアント印刷を実行します。
Web ブラウザーから "http://testsv:8080/createform/index.html" にアクセスします。Web ブラウザーに表示された " ブラウザー任意指定印刷 " ボタンをクリックします。印刷コントロールが起動し、Java Servlet にアクセスします。サーバー側で PrintStageWeb ランタイムの実行が終了すると、作成された CCD ファイルが HTTP レスポンスで書き出され、印刷コントロールは CCD ファイルをダウンロードします。
印刷コントロールは、CCD ファイルのダウンロードを終了すると、PrintStageWeb Client ランタイムを実行し、CCD ファイルを使用して印刷処理を実行します。印刷情報は、帳票の種類ごとにクライアントに保存されている印刷情報を反映します。
今回は、"SchoolLife" という帳票が指定されているので、"SchoolLife" の印刷情報を参照します。

※今回、"SchoolLife"の印刷情報を設定していませんが、印刷コントロールのメソッド「PrintExecuteEx」実行時に自動的に作成されます。
自動的に作成された印刷情報の各設定値は、全てデフォルト値となります。
※印刷情報の設定の詳細は、「16. 印刷設定の利用方法」をご覧ください。

以上で、ブラウザー任意指定印刷による Web クライアント印刷のサンプル実行は終了です。
※印刷コントロールの詳細は、「10. 印刷コントロール API 仕様」をご覧ください。

「ブラウザー任意指定印刷」は印刷コントロールと PrintStageWeb Client を使用して実装します。印刷コントロールとクライアントランタイムを合わせて使用することで、強力な Web クライアント印刷を行うことが可能になります。
※詳細は、「15.PrintStageWeb Client ランタイムについて」をご覧ください。

10. 印刷コントロール API 仕様

10-1. 概要

印刷コントロールの実行には「cwebclient.min.js」ファイルを使用します。

このファイルはプログラムフォルダーの「lib」フォルダーに配置されています。
印刷コントロールは、あらかじめ Web ページに定義しておき、この Web ページを読み込んだ際に起動します。起動後に印刷コントロールのプロパティ、メソッドを使用して Web クライアント印刷を実行します。

10-2.API 仕様

印刷コントロールの API 仕様です。プロパティとメソッドの型はそれぞれ以下の型を表します。

文字列	String
数値	Number
真偽値	Boolean

必須プロパティ：文字列 CreatePrintingDataScriptUrl

概要：

メソッド「PrintExecuteEx」「PrintExecutePdf」「PreviewExecutePdf」を実行した際に、このプロパティ値で設定された URL にアクセスします。このプロパティで設定される URL は、データソースを作成し、ランタイムを実行して印刷データをクライアントに返すスクリプトになります。URL は HTTP、HTTPS プロトコルに対応しています。URL は相対パスではなく http または https から始まるアドレスを設定してください。このプロパティは必須です。

各メソッド実行時に、ヌル値や空文字列が設定されている場合、エラーが発生します。

印刷コントロールは、このプロパティで設定されるスクリプトに対して、GET 文字列（GET で送信される文字列）あるいは POST 文字列（POST で送信される文字列）で "OutputFileName" を送信します。"OutputFileName" から、常にユニークな文字列のファイル名が取得できるため、ランタイム出力時にファイル名として使用することで、クライアントからのリクエストが多数発生した場合でも、上書きされることなく印刷データのファイルを作成することが可能です。

※メソッド「PrintExecuteEx」を実行した場合には、さらに "StyleName" と "CcdFileOption" を取得することができます。

※詳細は、「PrintExecuteEx」や「15.PrintStageWeb Client ランタイムについて」をご覧ください。

プロパティ：文字列 DeletePrintingDataScriptUrl

概要：

メソッド「PrintExecuteEx」「PrintExecutePdf」「PreviewExecutePdf」を実行した際に、このプロパティ値で設定された URL にアクセスします。サーバー上に残った印刷データを削除したい場合に、このプロパティに印刷データ（PDF ファイル、CCD ファイル）を削除する記述をしたスクリプトの URL を指定します。URL は HTTP、HTTPS プロトコルに対応しています。URL は相対パスではなく http または https から始まるアドレスを設定してください。

ヌル値や空文字列が設定されている場合、削除スクリプトへのアクセスは行いません。

印刷コントロールは、このプロパティで設定されるスクリプトに対して、GET 文字列（GET で送信される文字列）あるいは POST 文字列（POST で送信される文字列）で "DeleteFileName" を送信します。プロパティ「CreatePrintingDataScriptUrl」で指定されたスクリプトに送信される "OutputFileName" と全く同じ文字列を取得するので、削除したい印刷データのファイル名に指定することが可能です。

プロパティ：文字列 DataSendMethod

概要：

メソッド「PrintExecuteEx」「PrintExecutePdf」「PreviewExecutePdf」を実行した際に、このプロパティ値で設定された方法で、プロパティ「CreatePrintingDataScriptUrl」「DeletePrintingDataScriptUrl」に対して文字列を送信します。

このプロパティで設定できる値は "GET" と "POST" です。デフォルト値は "POST" です。

各メソッド実行時に、ヌル値や空文字列、または "GET"、"POST" 以外の値が設定されている場合、エラーが発生します。

プロパティ：文字列 SetRequestHeader

概要：

メソッド「PrintExecuteEx」「PrintExecutePdf」「PreviewExecutePdf」を実行した際に、プロパティ「CreatePrintingDataScriptUrl」で指定されたサーバーに対してこのプロパティ値で設定した HTTP リクエストヘッダを送信します。
設定のフォーマットは以下の通りです。

"<key1>=<value1>&<key2>=<value2>&...&<keyN>=<valueN>"

例えば以下のような文字列を指定した場合は、プロパティ「CreatePrintingDataScriptUrl」で設定されたスクリプトでは、POST 文字列を取得する方法で、"name"、"product" からそれぞれ "create"、"header" を取得することができます。

SetRequestHeader = "name=create&product=header"

プロパティ：文字列 FormRequestArray

概要：

メソッド「PrintExecuteEx」「PrintExecutePdf」「PreviewExecutePdf」を実行した際に、プロパティ「CreatePrintingDataScriptUrl」で指定されたスクリプトに対してこのプロパティ値で設定した文字列を送信します。
※プロパティ「DeletePrintingDataScriptUrl」で指定するスクリプトへは送信しません。
また、このプロパティ値が送信されるのは、プロパティ「DataSendMethod」に "POST" が指定されている時のみです。
設定のフォーマットは以下の通りです。

"<key1>=<value1>&<key2>=<value2>&...&<keyN>=<valueN>"

例えば以下のような文字列を指定した場合は、プロパティ「CreatePrintingDataScriptUrl」で設定されたスクリプトでは、POST 文字列を取得する方法で、"name"、"product" からそれぞれ "create"、"form" を取得することができます。

FormRequestArray = "name=create&product=form"

プロパティ：数値 MinimizedProgressDlg

概要：

メソッド「PrintExecuteEx」「PrintExecutePdf」「PreviewExecutePdf」実行時に表示される印刷ステータス画面の表示設定を指定します。
このプロパティに "1" を指定すると、印刷ステータス画面は表示されません。指定なし、及び "0" を指定すると、印刷ステータス画面は表示されます。
※詳細は、「18. 印刷ステータス画面」をご覧ください。

プロパティ：文字列 ResultString

概要：

メソッド「SyncPrintExecuteEx」「SyncPrintExecutePdf」「SyncPreviewExecutePdf」を実行した際の処理結果を文字列で返します。

プロパティ：数値 SecurityOption

概要：

転送時のファイルを暗号化するかを指定します。
このプロパティに "1" を指定すると転送ファイルは暗号化されます。指定なし、および "0" を指定すると、転送ファイルは暗号化されません。
※詳細は、「20. セキュリティ印刷」をご覧ください。

プロパティ：数値 ServicePort

概要：

Create!Form WebClient Service への接続ポート番号を指定します。
デフォルト値は "55555" です。接続ポート番号をデフォルト値以外に変更している場合のみ指定します。

メソッド：PrintExecuteEx(文字列 StyleName, 真偽値 IsDisplayedSetting)

引数：

StyleName

指定された文字列は、プロパティ「CreatePrintingDataScriptUrl」で設定されたスクリプトに対して、GET 文字列、あるいは POST 文字列で送信される（どちらで送信されるかはプロパティ「DataSendMethod」の設定値によります）ので、スクリプト側では "StyleName" で取得することが可能です。

引数：

IsDisplayedSetting

スクリプトへ送信前に印刷設定画面を表示するかを指定します。デフォルト値は "false" です。

概要：

プロパティ「CreatePrintingDataScriptUrl」で設定されたスクリプトを実行し、印刷データをダウンロード後、印刷を行います。
このメソッドが使用できる印刷データは「CCD ファイル」です。従って、サーバーのランタイムは PrintStageWeb ランタイムでなければなりません。
また、クライアントマシンには PrintStageWeb Client ランタイムが導入されている必要があります。
※詳細は、「15.PrintStageWeb Client ランタイムについて」をご覧ください。

メソッド：PrintExecutePdf(真偽値 IsDisplayedSetting)

引数：

IsDisplayedSetting

スクリプトへ送信前に印刷設定画面を表示するかを指定します。デフォルト値は "false" です。

概要：

プロパティ「CreatePrintingDataScriptUrl」で設定されたスクリプトを実行し、印刷データをダウンロード後、印刷を行います。
このメソッドが使用できる印刷データは「PDF ファイル」です。従って、サーバーのランタイムは Cast ランタイムでなければなりません。
また、クライアントマシンには Adobe Acrobat や Adobe Acrobat Reader が必要です。

メソッド：PreviewExecutePdf(真偽値 IsShownInTab)

引数：

IsShownInTab

MinimizedProgressDlg が "1" の場合に PDF ファイルの表示を Web ブラウザーの新規タブとして表示するかを指定します。デフォルト値は "false" です。

概要：

プロパティ「CreatePrintingDataScriptUrl」で設定されたスクリプトを実行し、印刷データをダウンロード後、PDF ファイルの表示を行います。

このメソッドが使用できる印刷データは「PDF ファイル」です。従って、サーバーのランタイムは Cast ランタイムでなければなりません。

また、クライアントマシンには Adobe Acrobat や Adobe Acrobat Reader が必要です。

メソッド：Init()

概要：

印刷コントロールのすべてのプロパティをデフォルト値に戻します。

メソッド：ChangePrintInfo(文字列 StyleName)

引数：

StyleName

クライアントに保存されている印刷情報の帳票名、あるいは新規作成する帳票名を指定します。

概要：

引数「StyleName」で指定された帳票名を持つ印刷情報の設定変更、あるいは新規作成を行います。帳票名は Job ファイル名となります。

クライアントに PrintStageWeb Client ランタイムが導入されている場合、このメソッドを実行すると、印刷設定画面が表示されます。

※印刷設定画面の詳細は、「16. 印刷設定の利用方法」をご覧ください。

メソッド：ChangePrintPdf()

概要：

PDF 非表示印刷の印刷情報の設定変更を行います。

※印刷設定画面の詳細は、「16. 印刷設定の利用方法」をご覧ください。

メソッド：ResetClosedFinishing()

概要：

メソッド「PrintExecuteEx」「PrintExecutePdf」「PreviewExecutePdf」実行時に表示される印刷ステータス画面を自動的に閉じないようにします。

処理が正常終了していない場合などで、処理結果を確認する際に使用します。

メソッド：SetClosedFinishing()

概要：

メソッド「PrintExecuteEx」「PrintExecutePdf」「PreviewExecutePdf」実行時に表示される印刷ステータス画面を自動的に閉じるようにします。

メソッド：数値 retval SyncPrintExecuteEx(文字列 StyleName, 真偽値 IsDisplayedSetting)

戻り値：

retval

処理結果を返します。処理が成功した場合は "0" です。"0" 以外は不正終了です。
プロパティ「ResultString」で詳細なエラー情報を確認できます。

引数：

StyleName

指定された文字列は、プロパティ「CreatePrintingDataScriptUrl」で設定されたスクリプトに対して、GET 文字列、あるいは POST 文字列で送信される（どちらで送信されるかはプロパティ「DataSendMethod」の設定値によります）ので、スクリプト側では "StyleName" で取得することが可能です。

引数：

IsDisplayedSetting

スクリプトへ送信前に印刷設定画面を表示するかを指定します。デフォルト値は "false" です。

概要：

処理内容はメソッド「PrintExecuteEx」と同様ですが、メソッド「PrintExecuteEx」が戻り値を返さないのに対して、このメソッドは戻り値を返します。
処理中は Web ブラウザーを操作することができません。

メソッド：数値 retval SyncPrintExecutePdf(真偽値 IsDisplayedSetting)

引数：

IsDisplayedSetting

スクリプトへ送信前に印刷設定画面を表示するかを指定します。デフォルト値は "false" です。

戻り値：

retval

処理結果を返します。処理が成功した場合は "0" です。"0" 以外は不正終了です。
プロパティ「ResultString」で詳細なエラー情報を確認できます。

概要：

処理内容はメソッド「PrintExecutePdf」と同様ですが、メソッド「PrintExecutePdf」が戻り値を返さないのに対して、このメソッドは戻り値を返します。
処理中は Web ブラウザーを操作することができません。

メソッド：数値 retval SyncPreviewExecutePdf(真偽値 IsShownInTab)

引数：

IsShownInTab

MinimizedProgressDlg が "1" の場合に PDF ファイルの表示を Web ブラウザーの新規タブとして表示するかを指定します。デフォルト値は "false" です。

戻り値：

retval

処理結果を返します。処理が成功した場合は "0" です。"0" 以外は不正終了です。
プロパティ「ResultString」で詳細なエラー情報を確認できます。

概要：

処理内容はメソッド「PreviewExecutePdf」と同様ですが、メソッド「PreviewExecutePdf」が戻り値を返さないのに対して、このメソッドは戻り値を返します。
処理中は Web ブラウザーを操作することができません。

11. 印刷コントロール 結果コード一覧

印刷コントロールから Web クライアント印刷を行うと、処理結果として結果コードと説明が印刷ステータス画面に表示されます。この結果コードや説明を確認することでエラー発生時の対処を行うことができます。
なお、この結果コードや説明についてはログファイルにも出力されます。
※ログファイルの詳細は、「21-7. ログ出力」をご覧ください。

正常レベル

0：処理は正常に終了しました。

警告レベル

1100

説明：印刷データ削除スクリプトにアクセスできませんでした。
DeletePrintingDataScriptUrl に不正な値が指定されているか、スクリプトへの要求に失敗しています。

対処：プロパティ「DeletePrintingDataScriptUrl」で指定された URL が正しくないか、サーバーあるいはクライアントの設定に問題があります。

1101

説明：印刷データ削除スクリプトにアクセスできませんでした。ステータスコード「＜数値＞」が返されました。

対処：プロパティ「DeletePrintingDataScriptUrl」で指定されたスクリプトへアクセスした時に、戻り値として返されたエラーコードが＜数値＞に表示されています。エラーコードを元に、サーバーの設定、あるいはスクリプトにエラーがないか確認してください。

1102

説明：ダウンロードした印刷データの削除に失敗しました。ディレクトリのアクセス権等を確認してください。

対処：CCD ファイル保存フォルダーで指定したディレクトリの権限を確認してください。

1103

説明：暗号鍵ファイルの削除に失敗しました。

1400

説明：ダウンロードした CCD ファイルは保存できません。この帳票資源データは保存が禁止されています。

対処：この帳票資源データは保存が禁止されています。サーバー側の PrintStagWeb ランタイムを実行したときのオプションを確認してください。

1401

説明：クライアントランタイム処理中に警告「＜クライアントランタイム処理結果＞」が返されました。

対処：印刷処理時に警告が発生しました。＜クライアントランタイム処理結果＞は警告コードで表示されるので、そのコードを元に原因を確認してください。

1500

説明：Acrobat プロセスの終了に失敗しました。

対処:Adobe Acrobat や Adobe Acrobat Reader のプロセスが残っているため、タスクマネージャーから該当するプロセスを削除してください。

エラーレベル

2100
説明：メモリが不足しています。
対処：その他のアプリケーションを終了するなどして、メモリを確保してください。

2101
説明：「通常使うプリンター」を取得することができません。Windows のコントロール
パネルから「通常使うプリンター」の設定を行ってください。
対処：クライアント端末にプリンターがインストールされていない可能性があります。
プリンターの設定を確認してください。

2102：処理は中断されました。

2103
説明：一時ファイルを保存するディレクトリの取得に失敗しました。
対処：アカウントの権限が制限されている可能性があります。CCD ファイル保存フォルダー
に対するアカウントの権限等を確認してください。

2104
説明：印刷データ生成スクリプトにアクセスできません。ステータスコード「＜数値＞」が
返されました。
対処：プロパティ「CreatePrintingDataScriptUrl」で指定されたスクリプトへのアクセス
時に返されたステータスコードが＜数値＞に表示されています。ステータスコード
を元にサーバーの設定、あるいはスクリプトにエラーがないか確認してください。

2105
説明：Create!Form V12 のインストールを確認できませんでした。正常にインストールされ
ていない可能性があります。
対処：Create!Form をアンインストールして、再度インストールしてください。

2118
説明：実行タイプが正しくありません。

2120
説明：CPrtDevice.exe 実行中に割り込みが発生しました。

2121
説明：プリンター一覧ファイルの読み込みに失敗しました。
対処：テンポラリフォルダーへのアクセス権を確認してください。

2122
説明：CPrtDevice.exe 実行において例外が発生しました。

2123
説明：CPdfModuleVersion.exe 実行において例外が発生しました。
対処：拡張子「.pdf」に関連付けるアプリケーションを Adobe Acrobat または
Adobe Acrobat Reader に設定し、サービスを再起動してください。

2124
説明：プリンタースプーラ―サービスが停止しています。
対処：OS のサービス画面から「Print Spooler」サービスを起動してください。

2130
説明：CPrtDevice.exe が見つかりませんでした。
対処：Create!Form をアンインストールして、再度インストールしてください。

2131
説明：CPdfExecutable.exe が見つかりませんでした。
対処：Create!Form をアンインストールして、再度インストールしてください。

2132
説明：CPdfModuleVersion.exe が見つかりませんでした。
対処：Create!Form をアンインストールして、再度インストールしてください。

2133
説明：CPrtWatcher.exe が見つかりませんでした。
対処：Create!Form をアンインストールして、再度インストールしてください。

2134
説明：CSec.exe が見つかりませんでした。
対処：Create!Form をアンインストールして、再度インストールしてください。

2135
説明：CProcess.exe が見つかりませんでした。
対処：Create!Form をアンインストールして、再度インストールしてください。

2136
説明：CSwitchDesktop.exe が見つかりませんでした。
対処：Create!Form をアンインストールして、再度インストールしてください。

2140
説明：キャンセルファイルの生成に失敗しました。
対処：テンポラリフォルダーへの書き込みに関するアクセス権を確認してください。

2141
説明：キャンセルファイルの書き込みに失敗しました。
対処：テンポラリフォルダーへの書き込みに関するアクセス権を確認してください。

2142
説明：ステータスファイルの生成に失敗しました。
対処：テンポラリフォルダーへの書き込みに関するアクセス権を確認してください。

2143
説明：ステータスファイルの書き込みに失敗しました。
対処：テンポラリフォルダーへの書き込みに関するアクセス権を確認してください。

2144
説明：印刷設定ファイルへのアクセスに失敗しました。
対処：ユーザー設定（共通）フォルダーへのアクセス権を確認してください。

2145
説明：印刷設定の取得に失敗しました。
対処：印刷設定ファイルが正しく保存されていることを確認してください。

2146
説明：印刷設定ファイルの保存に失敗しました。
対処：ユーザー設定（共通）フォルダーへの書き込みに関するアクセス権を確認してください。

2147
説明：サービス設定ファイルへのアクセスに失敗しました。
対処：ユーザー設定（共通）フォルダーへのアクセス権を確認してください。

2148
説明：HTTP 設定ファイルへのアクセスに失敗しました。
対処：ユーザー設定（共通）フォルダーへのアクセス権を確認してください。

2201
説明:不正なプロトコルが指定されています。本製品は HTTP、あるいは HTTPS のみ有効です。
対処：使用しているプロトコルが「HTTP」もしくは「HTTPS」か確認してください。

2203
説明：サーバースクリプトへの要求に失敗しました。
対処：サーバーのスクリプトへ要求を発行することができませんでした。クライアントのネットワーク環境を確認してください。

2204
説明：印刷データのダウンロードに失敗しました。
対処：クライアントのネットワーク環境を確認してください。

2205
説明：DataSendMethod プロパティに不正なメソッドが指定されています。
本製品は POST、GET のみ有効です。
対処：使用しているメソッドが「POST」もしくは「GET」か確認してください。

2212
説明：制限以上の長さの URL が指定されています。CreatePrintingDataScriptUrl
プロパティに設定できる URL の長さは、2048 バイトまでです。
対処：プロパティ「CreatePrintingDataScriptUrl」に設定した URL の長さを確認してください。

2213
説明：URL エンコードに失敗しました。URL を確認してください。

2214
説明：CreatePrintingDataScriptUrl が設定されていません。
対処：印刷データ生成スクリプトの URL を設定してください。

2215
説明：URL アクセスにおいてタイムアウトが発生しました。
対処：クライアントのネットワーク環境を確認してください。

2216
説明：URL アクセス結果取得においてタイムアウトが発生しました。
対処：クライアントのネットワーク環境を確認してください。

2217
説明：URL アクセスエラーが発生しました。
対処：クライアントのネットワーク環境を確認してください。

2218
説明：URL アクセスにおいて例外が発生しました。
対処：クライアントのネットワーク環境を確認してください。

2219
説明：ダウンロードしたデータが空です。
対処：サーバーで実行結果を返しているか確認してください。

2300
説明：Adobe Acrobat、Adobe Acrobat Reader のモジュールパスファイルが存在しません。
対処：テンポラリフォルダーへの書き込みに関するアクセス権を確認してください。

2301
説明：Adobe Acrobat、Adobe Acrobat Reader のモジュールパスの取得に失敗しました。
対処：テンポラリフォルダーへのアクセス権を確認してください。

2302
説明：Adobe Acrobat、Adobe Acrobat Reader のモジュールパスが空です。
対処：Adobe Acrobat や Adobe Acrobat Reader を再インストールしてください。

2303
説明：PDF ファイルに関連付けられているアプリケーションが Adobe Acrobat、
Adobe Acrobat Reader ではありません。
対処：拡張子「.pdf」に関連付けるアプリケーションを Adobe Acrobat または
Adobe Acrobat Reader に設定し、サービスを再起動してください。

2304
説明：ダウンロードした PDF ファイルのヘッダーが正しくありません。サーバーに導入さ
れているランタイムを確認してください。
対処：ダウンロードしたファイルが PDF ファイルではありませんでした。サーバーで
実行されたランタイム製品が「Cast」か確認してください。

2305
説明：PDF ファイルが見つかりませんでした。
対処：テンポラリフォルダーへの書き込みに関するアクセス権を確認してください。

2306
説明：PDF ファイルの読み込みに失敗しました。
対処：テンポラリフォルダーへのアクセス権を確認してください。

2399
説明：PDF 非表示印刷において不明なエラーが発生しました。

2400
説明：印刷データ生成スクリプトへ送信する文字列の作成に失敗しました。不正な
スタイル名が指定されているか、または印刷情報が正常でない可能性があります。
対処：印刷実行時に指定した帳票名と Job ファイル名を確認してください。

2401
説明：ダウンロードした CCD ファイルにアクセスできません。ディレクトリのアクセス権
等を確認してください。
対処：アカウントの権限が制限されている可能性があります。アカウントの権限等を確認
してください。

2402
説明：ダウンロードした CCD ファイルを読み込むことができません。ディレクトリの
アクセス権等を確認してください。
対処：アカウントの権限が制限されている可能性があります。アカウントの権限等を
確認してください。

2405
説明：ダウンロードした CCD ファイルは要求した帳票のものではありません。
サーバー管理者に問い合わせてください。
対処：印刷を実行した帳票名とダウンロードした CCD ファイルの帳票名が異なります。
印刷時の設定を確認してください。

2406
説明：CCD ファイル保存フォルダーパスの取得に失敗しました。
対処：アカウントの権限が制限されている可能性があります。アカウントの権限等を確認
してください。

2407
説明：CCD ファイルの保存に失敗しました。ディレクトリのアクセス権等を確認してくださ
い。
対処：アカウントの権限が制限されている可能性があります。アカウントの権限等を確認
してください。

2408
説明：CCD ファイルのタイムスタンプを保存できません。
対処：アカウントの権限が制限されている可能性があります。アカウントの権限等を確認
してください。

2409
説明：ダウンロードした CCD ファイルのヘッダーが正しくありません。サーバーに導入されているランタイムを確認してください。
対処：ダウンロードしたファイルが CCD ファイルではありませんでした。サーバーで実行されたランタイム製品が「PrintStageWeb」か確認してください。

2411
説明：クライアントランタイム処理中にエラー値「＜クライアントランタイム処理結果＞」が返されました。
対処：PrintStageWeb Client が印刷処理中にエラーを返しました。PrintStageWeb のエラーコードから＜クライアントランタイム処理結果＞を元に原因を確認してください。

2413
説明：PrintStageWeb ランタイムの実行権限がありません。
対処：サーバーに導入した PrintStageWeb ランタイムのライセンスパスワードを確認してください。

2414
説明：CPClient.exe が見つかりません。
対処：Create!Form をアンインストールして、再度インストールしてください。

2417
説明：CCD ファイルのタイムスタンプの取得に失敗しました。

2418
説明：CCD ファイルのエンコードに失敗しました。

2499
説明：ブラウザー任意指定印刷において不明なエラーが発生しました。

2700
説明：PDF プレビューの一時ファイルが見つかりませんでした。
対処：テンポラリフォルダーへの書き込みに関するアクセス権を確認してください。

2701
説明：PDF プレビューの一時ファイルの読み込みに失敗しました。
対処：テンポラリフォルダーへのアクセス権を確認してください。

2799
説明：PDF プレビューにおいて不明なエラーが発生しました。

3000
説明：CPrtDevice.exe において不明なエラーが発生しました。

3001
説明：CPrtDevice.exe 実行メモリが不足しています。

3002
説明：CPrtDevice.exe の引数に誤りがあります。

3003
説明：ファイルの削除に失敗しました。

3004
説明：printers ファイルの生成に失敗しました。
対処：テンポラリフォルダーへの書き込みに関するアクセス権を確認してください。

3005
説明：プリンター覧の取得に失敗しました。
対処：クライアント端末にプリンターがインストールされていない可能性があります。
プリンターの設定を確認してください。

3006
説明：プリンター覧の格納に失敗しました。

3007
説明：プリンターの取得に失敗しました。
対処：プリンターの設定を確認してください。

3008
説明：printers ファイルへの書き込みに失敗しました。
対処：テンポラリフォルダーへの書き込みに関するアクセス権を確認してください。

3009
説明：printers ファイルのクローズに失敗しました。
対処：テンポラリフォルダーへの書き込みに関するアクセス権を確認してください。

3014
説明：default ファイルの生成に失敗しました。
対処：テンポラリフォルダーへの書き込みに関するアクセス権を確認してください。

3015
説明：通常使うプリンターの取得に失敗しました。
対処：プリンターの設定を確認してください。

3016
説明：default ファイルへの書き込みに失敗しました。
説明：テンポラリフォルダーへの書き込みに関するアクセス権を確認してください。

3017
説明：default ファイルのクローズに失敗しました。
説明：テンポラリフォルダーへの書き込みに関するアクセス権を確認してください。

3100
説明：CPdfExecutable.exe において不明なエラーが発生しました。

3101
説明：CPdfExecutable.exe 実行メモリが不足しています。

3102
説明：temp.pdf ファイルの生成に失敗しました。
対処：テンポラリフォルダーへの書き込みに関するアクセス権を確認してください。

3103
説明：temp.pdf ファイルのクローズに失敗しました。
対処：テンポラリフォルダーへの書き込みに関するアクセス権を確認してください。

3104
説明：temp.pdf ファイルの削除に失敗しました。

3105
説明：executable ファイルの削除に失敗しました。

3106
説明：executable ファイルの生成に失敗しました。
対処：テンポラリフォルダーへの書き込みに関するアクセス権を確認してください。

3107
説明：executable ファイルの書き込みに失敗しました。
対処：テンポラリフォルダーへの書き込みに関するアクセス権を確認してください。

3108
説明：executable ファイルのクローズに失敗しました。
対処：テンポラリフォルダーへの書き込みに関するアクセス権を確認してください。

3200
説明：CPdfModuleVersion.exe において不明なエラーが発生しました。

3201
説明：CPdfModuleVersion.exe の取得に失敗しました。
対処：Create!Form をアンインストールして、再度インストールしてください。

3202
説明：CPdfModuleVersion.exe において MFC の初期化に失敗しました。
対処：Create!Form をアンインストールして、再度インストールしてください。

3203
説明：CPdfModuleVersion.exe の引数に誤りがあります。

3204
説明：Acrobat プロセスが正しく指定されませんでした。
対処：Adobe Acrobat や Adobe Acrobat Reader を再インストールしてください。

3205
説明：Acrobat プロセスバージョン情報ファイルの書き込みに失敗しました。
対処：テンポラリフォルダーへの書き込みに関するアクセス権を確認してください。

3300
説明：CPrtWatcher.exe においてエラー値「< 数値 >」が返されました。

3301
説明：CPrtWatcher.exe の引数に誤りがあります。

3302
説明：印刷が中断されました。

3303
説明：プリンターハンドルの取得に失敗しました。
対処：プリンターの設定を確認してください。

3304
説明：プリンターハンドルの取得に失敗しました。
対処：プリンターの設定を確認してください。

3305
説明：プリンター情報の取得に失敗しました。
対処：プリンターの設定を確認してください。

3306
説明：プリンタードライバーのデータ取得メモリが不足しています。

3307
説明：プリンタードライバーのデータの取得に失敗しました。
対処：プリンターの設定を確認してください。

3308
説明：Acrobat プロセスの起動に失敗しました。
対処：Adobe Acrobat や Adobe Acrobat Reader が起動できませんでした。
正しくインストールをされているか確認してください。

3309
説明：印刷ジョブ情報取得メモリが不足しています。

3310
説明：印刷ジョブ情報取得メモリが不足しています。

3311
説明：印刷ジョブの取得に失敗しました。

3312
説明：印刷ジョブ情報取得メモリが不足しています。

3313
説明：印刷ジョブの取得に失敗しました。
対処：プリンターの設定を確認してください。

3315
説明：セッションの列挙に失敗しました。

3490
説明：戻り値の取得に失敗しました。

3491
説明：引数が正しくありません。

3492
説明：セッションの列挙に失敗しました。

3494
説明：プロセスの実行に失敗しました。

4000
説明：暗号化機能モジュールにおいてエラー値「<数値>」が返されました。

4001
説明：暗号化機能モジュールの取得に失敗しました。
対処：暗号化機能モジュールが正しくインストールされているか確認してください。

4002
説明：MFC の初期化に失敗しました。
対処：Create!Form をアンインストールして、再度インストールしてください。

4003
説明：公開鍵ファイルの生成に失敗しました。
対処：テンポラリフォルダーへの書き込みに関するアクセス権を確認してください。

4004
説明：秘密鍵ファイルの生成に失敗しました。
対処：テンポラリフォルダーへの書き込みに関するアクセス権を確認してください。

4005
説明：秘密鍵ファイルのオープンに失敗しました。
対処：テンポラリフォルダーへのアクセス権を確認してください。

4006
説明：復号化するファイルが存在しません。
対処：テンポラリフォルダーへのアクセス権を確認してください。

4007
説明：復号化したファイルが存在しません。
対処：テンポラリフォルダーへの書き込みに関するアクセス権を確認してください。

4101
説明：メモリが不足しています。

4201
説明：ファイルを作成できません。ファイルのアクセス権限を確認してください。

4202
説明：ファイルを参照できません。ファイルのアクセス権限を確認してください。

4203
説明：ファイルを操作できません。ファイルのアクセス権限を確認してください。

4204
説明：ファイル内容がありません。ファイル転送中にエラーが起きた可能性があります。

4301
説明：キー生成に失敗しました。

4302
説明：暗号化に失敗しました。

4303
説明：復号化に失敗しました。

4401
説明：キー生成に失敗しました。

4402
説明：暗号化に失敗しました。

4403
説明：復号化に失敗しました。

4501
説明：公開鍵のエンコードに失敗しました。

4502
説明：公開鍵ファイルがありません。

4503
説明：公開鍵ファイルへのアクセスに失敗しました。

4504
説明：暗号化機能モジュールにおいて割り込みが発生しました。

12. 送信クエリ文字列の予約キーワード

プロパティ「FormRequestArray」やGET文字列を使用して、プロパティ「CreatePrintingDataScriptUrl」で指定したスクリプトに対して、データ送信を行うことが可能ですが、使用するメソッドによって予約されたキーワードがあります。
このキーワードを設定してしまうと、印刷コントロールは正常に処理されません。

使用するメソッド	予約キーワード
PrintExecutePdf (SyncPrintExecutePdf)	"OutputFileName"
PreviewExecutePdf (SyncPreviewExecutePdf)	"OutputFileName"
PrintExecuteEx (SyncPrintExecuteEx)	"OutputFileName"、"StyleName"、"CcdFileOption"

また、「DeletePrintingDataScriptUrl」で指定したスクリプトでは「DeleteFileName」が予約キーワードとなります。

13. 印刷データを削除する

システムの運用ポリシーによっては、サーバーで作成された印刷データが不要である場合があります。ここでは、サーバー側に残った印刷データを削除する方法を説明します。

13-1. バイナリを扱えないスクリプト（クラシック ASP 等）を使用している場合

CreatePrintingDataScriptUrl で指定された印刷データ作成スクリプトは、HTTP レスポンスに印刷データを書き込まなければなりません。従って、クラシック ASP のようなバイナリを使用できないスクリプトでは、印刷データを Web で参照できるディレクトリに作成して、URL リダイレクトを使用して HTTP レスポンスに印刷データをセットする必要があります。
後述の「2. バイナリを扱えるスクリプト（Java Servlet 等）を使用している場合」のように、印刷データ作成スクリプトで印刷データを削除することはできません。このような場合は、印刷コントロールのプロパティ DeletePrintingDataScriptUrl に、印刷データを削除する処理を行うスクリプトを設定しなければなりません。
印刷コントロールは、CreatePrintingDataScriptUrl で指定されたスクリプトにアクセスした後、DeletePrintingDataScriptUrl で指定されたスクリプトにアクセスします。
印刷データ削除スクリプトの例は以下の通りです。

```
<%@ Language=VBScript%>
<%
Set obj = Server.CreateObject("Scripting.FileSystemObject")
obj.DeleteFile(Server.MapPath("printings/" + Request.Form("DeleteFileName")))
Set obj = Nothing
%>
```

13-2. バイナリを扱えるスクリプト（Java Servlet 等）を使用している場合

バイナリを扱える Java Servlet 等のスクリプトで印刷データを作成している場合、その印刷データを削除する方法は簡単です。
CreatePrintingDataScriptUrl で指定された印刷データ作成スクリプトは、HTTP レスポンスに印刷データを書き込まなければなりません。クラシック ASP のようなバイナリを使用できないスクリプトでは、URL リダイレクト等を使用して HTTP レスポンスに印刷データをセットしなければなりませんが、Java Servlet のようなスクリプトはバイナリデータを HTTP レスポンスに書き込めるため、印刷データを HTTP レスポンスに書き込んだ後、その印刷データを削除することができます。
印刷データを HTTP レスポンスに書き込んだ後に印刷データを削除する Java Servlet の例は以下の通りです。

```
--------------------------------------------------------------------------------
protected void doGet(HttpServletRequest req, HttpServletResponse res)
   throws ServletException, IOException {
   // 出力 PDF ファイル
   File outputFile = new File("/usr/local/tomcat/webapps/createform/printings",
            req.getParameter("OutputFileName"));

   //・・・ランタイム実行など・・・

   OutputStream out = res.getOutputStream();
   out.write(Files.readAllBytes(outputFile.toPath()));

   // 印刷データを削除
   outputFile.delete();
}
--------------------------------------------------------------------------------
```

バイナリを扱える場合はリダイレクトを使用する必要がないため、印刷データファイルはサーバーのローカルであればどこにでも作成できる、という利点もあります。

なお、JavaAPI の RuntimeParam#setInputStream や RuntimeParam#setOutputStream を利用することで印刷データファイルを作成することなくストリームで処理することもできます。

14. スクリプトへデータを送信する

通常、Web システムでは、HTML フォームや GET 文字列などを利用して、Web ページからスクリプトに対してデータを送信する処理があります。
印刷コントロールではプロパティ「SetRequestHeader」、「FormRequestArray」を使用する他、印刷コントロールを起動する Web ページから、プロパティ「CreatePrintingDataScriptUrl」で指定されたスクリプトに対してデータを送信する方法が用意されています。

14-1. POST 文字列

POST 文字列を利用したデータ送信の方法を紹介します。
通常 POST 文字列は、HTML フォームを使用してサブミットされた際にスクリプトに送信されます。印刷コントロールでは、プロパティ「SetRequestHeader」、「FormRequestArray」に送信する文字列を指定します。

```
SetRequestHeader = "request=header&date=20231006";
FormRequestArray = "name= 帳票 太郎 &id=A10256";
```

例えば、上記のようにプロパティ「SetRequestHeader」、「FormRequestArray」の値をそれぞれ設定すると（例は JavaScript です）、プロパティ「CreatePrintingDataScriptUrl」で指定されたスクリプトでは、以下のようにして、" 帳票 太郎 "、"A10256" という文字列を取得することが可能です。（クラシック ASP では、HTTP リクエストヘッダ情報はすべて取得されます）

クラシック ASP の例：

```
http_header= Request.ServerVariables("ALL_HTTP")

name = Request.Form("name")
id = Request.Form("id")
```

Java Servlet の例：

```
request = req.getHeader("request");
date = req.getHeader("date");

name = req.getParameter("name");
id = req.getParameter("id");
```

プロパティ「SetRequestHeader」、「FormRequestArray」に設定する文字列は、特定のフォーマットに合わせてデータの形を変更しなければなりません。
※詳細は、「10. 印刷コントロール API 仕様」をご覧ください。
プロパティ「FormRequestArray」に指定された文字列は、プロパティ「DataSendMethod」で "POST" を設定しておかなければなりません。"GET" が設定されている場合、プロパティ「FormRequestArray」に文字列が設定されていてもデータの送信は行われません。

<< 注意 >>
メソッド「PrintExecutePdf」「PreviewExecutePdf」「SyncPrintExecutePdf」「SyncPreviewExecutePdf」を使用している場合は"OutputFileName"、メソッド「PrintExecuteEx」「SyncPrintExecuteEx」を使用している場合は"OutputFileName"、"StyleName"、"CcdFileOption"というキーワードは予約されています。
これらのキーワードと同名のものをFormRequestArrayに使用した場合、正常な処理が行われません。

14-2.GET 文字列

通常GET文字列は、URLの最後に"?"を付けて、その後に記述します。印刷コントロールでも全く同じ指定方法になります。
プロパティ「CreatePrintingDataScriptUrl」に指定したURLの最後に通常と同じようにテキストデータを付けることで、GET文字列の送信を行います。

```
CreatePrintingDataScriptUrl = "http://testsv/createform/run.asp?name=帳票 太郎
&id=A10256";
```

例えば、上記のようにURLにGET文字列を設定すると（例はJavaScriptです）、プロパティ「CreatePrintingDataScriptUrl」で指定されたスクリプトでは、以下のようにして"帳票 太郎"、"A10256"という文字列を取得することが可能です。

クラシックASPの例：
```
name = Request.QueryString("name")
id = Request.QueryString("id")
```
Java Servletの例：
```
name = req.getParameter("name");
id = req.getParameter("id");
```

<< 注意 >>
メソッド「PrintExecutePdf」「PreviewExecutePdf」「SyncPrintExecutePdf」「SyncPreviewExecutePdf」を使用している場合は"OutputFileName"、メソッド「PrintExecuteEx」「SyncPrintExecuteEx」を使用している場合は"OutputFileName"、"StyleName"、"CcdFileOption"というキーワードは予約されています。
これらのキーワードと同名のものを使用した場合、正常な処理が行われません。
「SetRequestHeader」の指定はPOSTと同様です。詳細は「14-1.POST文字列」をご覧ください。

14-3. セッション変数

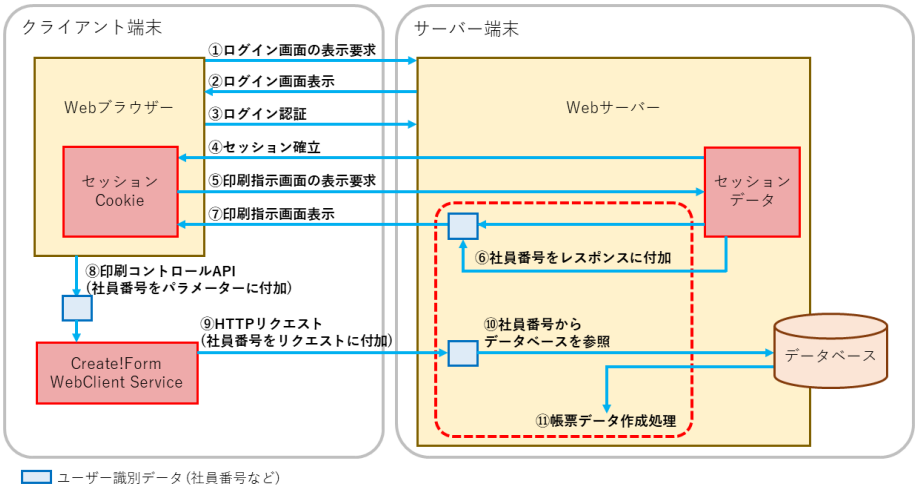
Webシステムでは、ユーザー名などをセッション変数に格納し、多くのスクリプトで使用するケースもありますが、印刷コントロールではセッション変数を使用することができません。

ただし、以下の方法でユーザー識別の仕組みを構築していただくことでサーバーへアクセスしたユーザーを識別できるようになるため、セッション変数と似たようなことは実現できます。

（a）ユーザー識別データを HTTP パラメーターで渡す
（b）暗号化したユーザー識別データを HTTP パラメーターで渡す
（c）入力データとワンタイム ID を生成してワンタイム ID を HTTP パラメーターで渡す

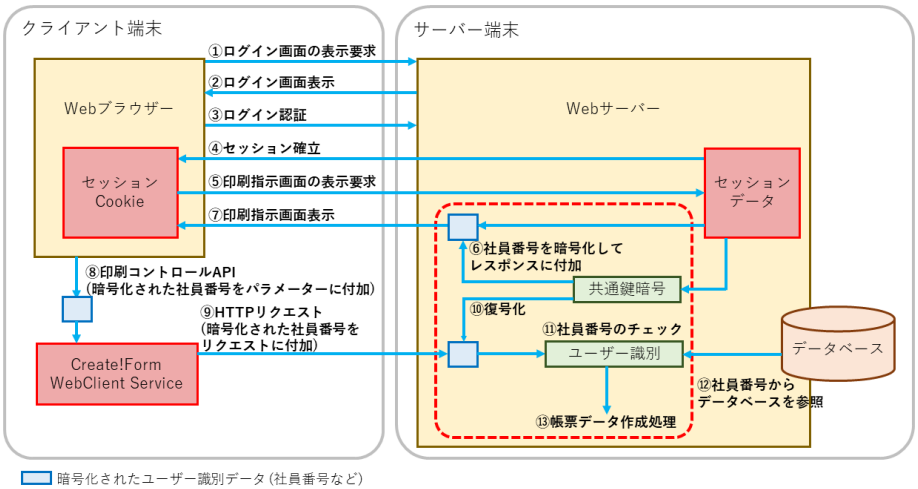
それぞれの仕組みについて説明します。

（a）ユーザー識別データを HTTP パラメーターで渡す



サーバーで印刷指示画面の表示要求を受けたタイミングでユーザー識別データをレスポンスで返します。印刷コントロール API では、SetRequestHeader または、FormRequestArray でこのユーザー識別データを付加して印刷実行を行います。HTTP リクエストでユーザー識別データがサーバーへ送信され、ユーザーが識別できます。その後は通常のランタイム実行と帳票データの作成処理を行います。

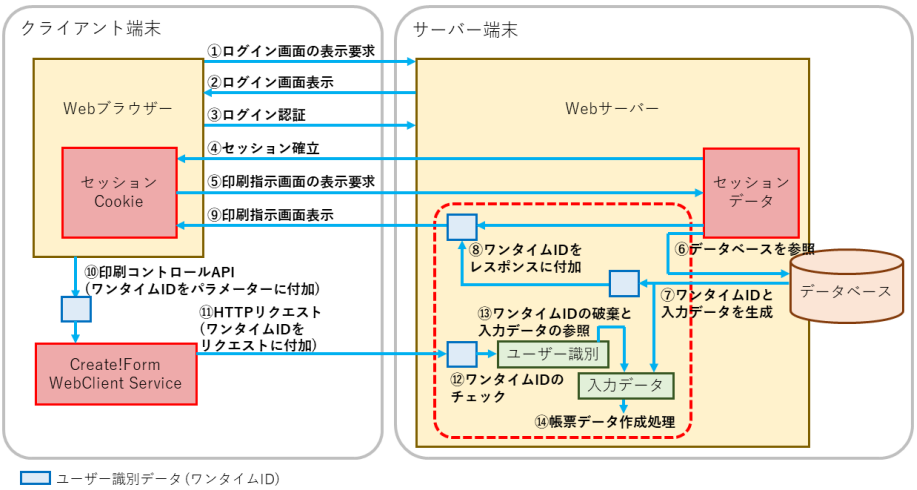
（b）暗号化したユーザー識別データを HTTP パラメーターで渡す



サーバーで印刷指示画面の表示要求を受けたタイミングで暗号化したユーザー識別データをレスポンスで返します。印刷コントロール API では SetRequestHeader または、FormRequestArray でこのユーザー識別データを付加して印刷実行を行います。HTTP リクエストでユーザー識別データがサーバーへ送信され、復号化とチェックによってユーザーが識別できます。その後は通常のランタイム実行と帳票データの作成処理を行います。

ユーザー識別データには社員番号のような単純なものだけでなく、現在時刻や接続元のクライアントの IP アドレスなどを含めて暗号化しておくことを推奨します。ユーザー識別データを複雑にしておくことで暗号化したユーザー識別データが他のクライアントから使いまわされたり改竄されたりすることへの対策となります。

(c) 入力データとワンタイム ID を生成してワンタイム ID を HTTP パラメーターで渡す



サーバーで印刷指示画面の表示要求を受けたタイミングでデータベースを参照し、有効期限付きでワンタイム ID と入力データを生成します。ワンタイム ID は入力データに紐づいており、レスポンスにはワンタイム ID を付加して返します。印刷コントロール API では SetRequestHeader または、FormRequestArray でこのワンタイム ID を付加して印刷実行を行います。HTTP リクエストでワンタイム ID がサーバーへ送信され、ユーザーの入力データが参照できます。その後は通常のランタイム実行と帳票データの作成処理を行います。

ワンタイム ID は規則性のない推測の難しい複雑な ID を生成することを推奨します。
また、ワンタイム ID の特徴としては一度その ID を使用すると使用済みとして扱われ、同じ ID は再度使用できなくなります。

ワンタイム ID による仕組みでは、ワンタイム ID が生成される前の状態ではクライアントからのリクエストは一切受け付けないため、総当たりによる攻撃に強くなります。
さらに、使用済みの ID は再度使用できないため、ID が流出したとしても他のクライアントから使いまわされることがありません。

15.PrintStageWeb Client ランタイムについて

印刷コントロールと PrintStageWeb Client ランタイムを連携させて行うことができる機能は以下の通りです。

・詳細な印刷情報の設定
・CCD ファイルを使用した印刷

それぞれの特徴について以下に説明します。

15-1. 詳細な印刷情報の設定

PDF 表示印刷や PDF 非表示印刷では、PrintStageWeb Client ランタイムを利用しない印刷となるため、詳細な印刷情報の設定はできません。
※ PDF 非表示印刷では、PDF 非表示印刷設定画面からプリンターの選択のみ可能です。
帳票出力業務においては、様々な種類の用紙や印刷設定を行う必要があります。PrintStageWeb Client ランタイムを導入することにより、以下の項目の設定を行うことが可能です。

・出力するプリンター
・給紙トレイ
・印刷部数
・両面印刷
・拡大／縮小
・解像度
・出力位置の補正

印刷情報は帳票ごとに保存することが可能なため、各帳票に適した印刷情報を設定できます。
印刷情報の設定は、印刷設定画面で行います。
※詳細は、「16. 印刷設定の利用方法」をご覧ください。

15-2.CCD ファイルを使用した印刷

PDF ファイルを用いた Web クライアント印刷の場合、これらのファイルでは帳票イメージの固定部分と可変部分が組み合わされているので、必ず全ての帳票イメージをダウンロードしなければなりません。PrintStageWeb Client ランタイムは PrintStageWeb ランタイムが出力する CCD ファイルを使用して印刷することが可能です。
CCD ファイルの内容は、帳票の固定部分（帳票資源データ）と可変部分（入力データ）です。この 2 つを切り分けているため、クライアントに帳票資源データを保存しておくことが可能となります。クライアントに帳票資源データが保存されている場合、印刷データをサーバーからダウンロードする際、可変部分のみダウンロードして印刷することになるため、インターネットのトラフィック量が軽減されます。

16. 印刷設定の利用方法

印刷情報の設定は、印刷設定画面から行います。
印刷設定画面は以下の方法で利用することができます。

・印刷コントロールのメソッド「ChangePrintInfo」の実行
・プログラムフォルダーの「PrtSTConf.exe」の実行

それぞれの起動方法について以下に説明します。

16-1. 印刷コントロールのメソッド「ChangePrintInfo」の実行

印刷コントロールを実行する Web ページにメソッド「ChangePrintInfo」記述し実行します。
以下に例を示します。

```
<!DOCTYPE html>
<html lang="ja">
<head>
<meta charset="UTF-8">
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
<script type="text/javascript" src="http://testsv:8080/createform/lib/
cwebclient.min.js?version=12.0.0" charset="utf-8"></script>
<script type="text/javascript">
function KickChangePrintInfo() {
   // 引数に帳票名（Job ファイル名）を指定します
   cwebclient.ChangePrintInfo("sheet");
}
</script>
</head>
<body>
   <!-- KickChangePrintinfo を実行するボタン等を配置します -->
   <input type="button" value="印刷設定" onclick="KickChangePrintInfo()"/>
</body>
</html>
```

メソッド「ChangePrintInfo」では引数に指定した帳票名の印刷設定を行うことが可能です。
印刷設定ファイルはユーザー設定（共通）フォルダーの「conf」フォルダーに作成されます。
ユーザー設定（共通）フォルダーはマネージャーの［ヘルプ］-［バージョン情報］-［バージョン情報詳細］ から確認できます。なお、メソッド「ChangePrintPdf」から PDF 非表示印刷の印刷設定を行うこともできます。

16-2. プログラムフォルダーの「PrtSTConf.exe」の実行

プログラムフォルダーの「PrtSTConf.exe」を実行します。
この方法で印刷設定を行う場合は、Web ページにメソッドを記述して実行することなく、任意の帳票名の印刷設定ファイルを作成することが可能となります。
主に、印刷設定ファイルを作成して複数のクライアントへ配布する際にこの方法を利用します。

17. 印刷設定画面

印刷設定画面は印刷コントロールのメソッドを利用するかどうかによって異なります。
以下では印刷コントロールのメソッドと印刷設定のプログラムの実行に分けて説明します。

17-1. 印刷コントロールのメソッド「ChangePrintInfo」の実行

Web ブラウザーから印刷コントロールのメソッド「ChangePrintInfo」を実行するとブラウザー任意指定印刷で利用する以下の印刷設定画面が表示されます。

図：印刷設定画面

①帳票名
印刷情報の設定変更を行う帳票名が表示されます。

②[保存] ボタン
各タブで設定された印刷情報を保存します。

③[設定の初期化] ボタン
各タブで設定された印刷情報をデフォルト値に戻します。

④[閉じる] ボタン
印刷設定画面を閉じます。

[プリンター設定]

図：印刷設定画面

①プリンター名

現在設定を行っている帳票を出力するプリンターを設定します。
Windows の通常使うプリンターまたはクライアントマシンにインストールされているプリンターから選択することが可能です。

<< 注意 >>

Windows 10 では通常使うプリンターの設定が 2 箇所あります。[コントロールパネル] の [デバイスとプリンター] と [Windows の設定] の [デバイス] です。[Windows の設定] の [デバイス] から [Windows で通常使うプリンターを管理する] を有効にした場合、通常使うプリンターは最後に使用したローカルプリンターのみ対応しています。ネットワークプリンターは対応していません。

②給紙トレイ

給紙トレイを設定します。
[プリンター名] で指定したプリンターの給紙トレイから設定可能です。
通常使うプリンターを選択した場合は給紙トレイの設定を行うことはできません。

③部数指定

出力部数を設定します。
最大値は "999" です。

④両面印刷

両面印刷の設定を行います。
[デフォルト] を選択した場合、出力するプリンターの設定に依存します。

[詳細設定]

図：印刷設定画面

①拡大 / 縮小

印刷時のスケールを元の出力イメージとの比率で設定します。

②解像度

解像度を dpi 単位で設定します。

③位置

印刷イメージを指定量だけ縦方向 / 横方向にずらすことが可能です。

[帳票資源データ設定]

図：印刷設定画面

①保存先

現在設定を行っている帳票の帳票資源データ（CCD ファイル）が保存されている、または保存するディレクトリのパスを設定します。

②最終更新日時

現在設定を行っている帳票の帳票資源データ（CCD ファイル）が保存されている場合、その帳票資源データのタイムスタンプを表示します。

③保存設定

現在設定を行っている帳票の帳票資源データ（CCD ファイル）の保存方法を設定します。
保存方法の設定は枠内のラジオボタンから行います。

・[保存しない] ラジオボタン

帳票資源データ（CCD ファイル）の保存を行いません。
従って、サーバーへ印刷要求を発行するごとに、帳票資源データと可変部データを合わせた CCD ファイルをダウンロードしなければなりません。

・[初回のみ保存する] ラジオボタン

サーバーへの初回印刷要求時のみ、ダウンロードした CCD ファイルから帳票資源データの部分のみをクライアントのローカルに保存します。
以降の印刷要求時は、可変部データのみダウンロードし、クライアントのローカルに保存された帳票資源データ（CCD ファイル）と合わせて出力します。

・[更新があれば保存する] ラジオボタン
クライアントのローカルに保存された帳票資源データとサーバーの帳票資源データを比較して、サーバーの帳票資源データが更新されていれば、帳票資源データと可変部データを合わせた CCD ファイルをダウンロードし、クライアントのローカルの帳票資源データ（CCD ファイル）を上書き・更新します。サーバーの帳票資源データに更新がなければ、可変部データのみダウンロードし、クライアントのローカルに保存された帳票資源データ（CCD ファイル）と合わせて出力を行います。

※帳票資源データの保存を有効に活用するためには、印刷データ生成スクリプトでのランタイム実行オプションを適切に記述する必要があります。
※詳細は、「19. CCD ファイル保存機能」をご覧ください。

17-2. 印刷コントロールのメソッド「ChangePrintPdf」の実行

Web ブラウザーから印刷コントロールのメソッド「ChangePrintPdf」を実行すると PDF 非表示印刷で利用する以下の印刷設定画面が表示されます。

図：PDF 非表示印刷設定画面

①プリンター名
PDF 非表示印刷で利用するプリンターを設定します。
Windows の通常使うプリンターまたはクライアントマシンにインストールされているプリンターから選択することが可能です。

<< 注意 >>
Windows 10 では通常使うプリンターの設定が 2 箇所あります。[コントロールパネル] の [デバイスとプリンター] と [Windows の設定] の [デバイス] です。[Windows の設定] の [デバイス] から [Windows で通常使うプリンターを管理する] を有効にした場合、通常使うプリンターは最後に使用したローカルプリンターのみ対応しています。ネットワークプリンターは対応していません。

② [保存] ボタン
設定された印刷情報を保存します。

③ [設定の初期化] ボタン
設定された印刷情報をデフォルト値に戻します。

④ [閉じる] ボタン
印刷設定画面を閉じます。

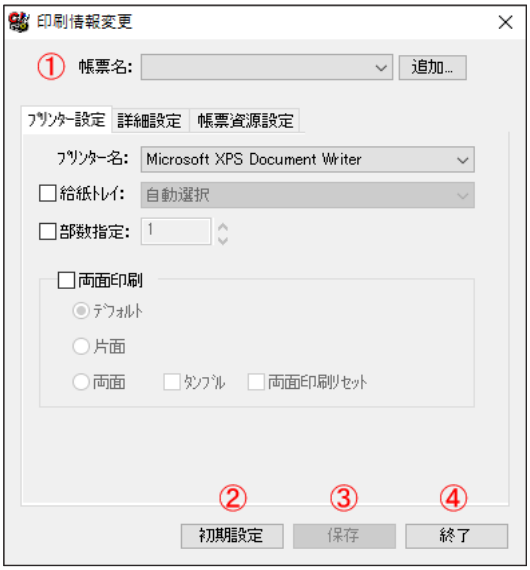
17-3. 印刷設定画面からの印刷実行

印刷コントロールのメソッド「PrintExecuteEx」や「PrintExecutePdf」の引数から印刷設定画面を表示するように指定することで印刷設定画面から印刷することもできます。
※詳細は、「10-2.API 仕様」をご覧ください。

17-4. プログラムフォルダーの「PrtSTConf.exe」の実行

プログラムフォルダーの「PrtSTConf.exe」を実行するとブラウザー任意指定印刷で利用する以下の印刷設定画面が表示されます。

図：印刷情報設定変更ダイアログ



①帳票名
印刷情報の作成を行う帳票名を指定します。

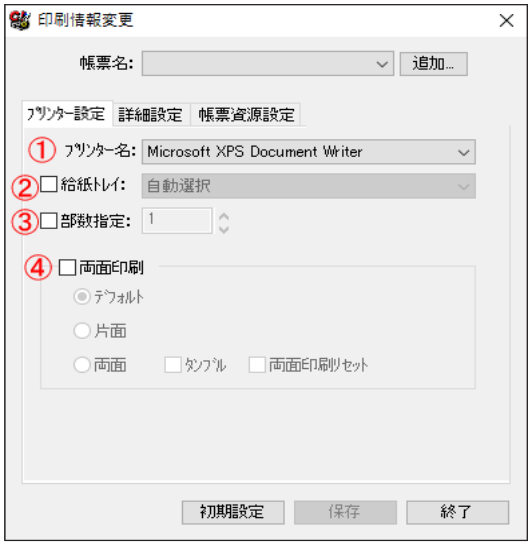
②[初期設定]ボタン
各タブで設定された印刷情報をデフォルト値に戻します。

③[保存]ボタン
各タブで設定された印刷情報を任意のディレクトリへ保存します。
帳票名が指定されていない場合は、印刷情報の保存を行うことはできません。

④[終了]ボタン
印刷情報設定変更ダイアログを終了します。
設定変更後にまだ保存が行われていない場合は、保存確認メッセージが表示されます。

[プリンター設定] タブ

図：印刷情報設定変更ダイアログ



①プリンター名

現在設定を行っている帳票を出力するプリンターを設定します。
Windows の通常使うプリンターまたはクライアントマシンにインストールされているプリンターから選択することが可能です。

<< 注意 >>

Windows 10 では通常使うプリンターの設定が 2 箇所あります。[コントロールパネル] の [デバイスとプリンター] と [Windows の設定] の [デバイス] です。[Windows の設定] の [デバイス] から [Windows で通常使うプリンターを管理する] を有効にした場合、通常使うプリンターは最後に使用したローカルプリンターのみ対応しています。ネットワークプリンターは対応していません。

②給紙トレイ

給紙トレイを設定します。
[プリンター名] で指定したプリンターの給紙トレイから設定可能です。
通常使うプリンターを選択した場合は給紙トレイの設定を行うことはできません。

③部数指定

出力部数を設定します。
最大値は "999" です。

④両面印刷

両面印刷の設定を行います。
[デフォルト] を選択した場合、出力するプリンターの設定に依存します。

[詳細設定] タブ

図：印刷情報設定変更ダイアログ

①拡大 / 縮小

印刷時のスケールを元の出力イメージとの比率で設定します。

②解像度

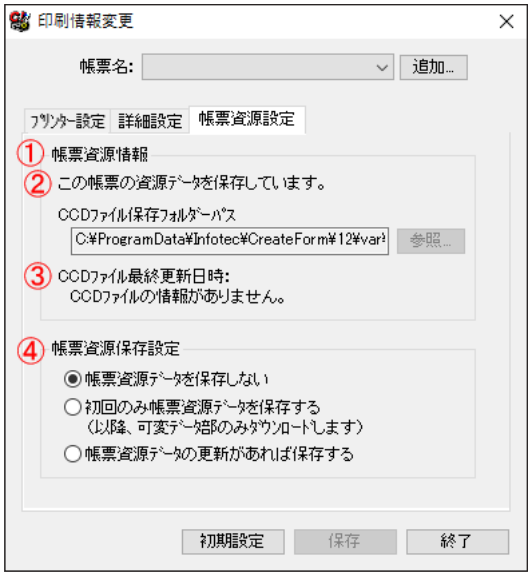
解像度を dpi 単位で設定します。

③位置

印刷イメージを指定量だけ縦方向 / 横方向にずらすことが可能です。

[帳票資源設定] タブ

図：印刷情報設定変更ダイアログ



①帳票資源情報

現在設定を行っている帳票の帳票資源データ（CCD ファイル）がクライアントのローカルに保存されているかどうかを表示します。

② CCD ファイル保存フォルダパス

現在設定を行っている帳票の帳票資源データ（CCD ファイル）が保存されている、または保存するディレクトリのパスを設定します。

③ CCD ファイル最終更新日時

現在設定を行っている帳票の帳票資源データ（CCD ファイル）が保存されている場合、その帳票資源データのタイムスタンプを表示します。

④帳票資源保存設定

現在設定を行っている帳票の帳票資源データ（CCD ファイル）の保存方法を設定します。
保存方法の設定は枠内のラジオボタンから行います。

・[帳票資源データを保存しない] ラジオボタン

帳票資源データ（CCD ファイル）の保存を行いません。
従って、サーバーへ印刷要求を発行するごとに、帳票資源データと可変部データを合わせた CCD ファイルをダウンロードしなければなりません。

・[初回のみ帳票資源データを保存する] ラジオボタン

サーバーへの初回印刷要求時のみ、ダウンロードした CCD ファイルから帳票資源データの部分のみをクライアントのローカルに保存します。
以降の印刷要求時は、可変部データのみダウンロードし、クライアントのローカルに保存された帳票資源データ（CCD ファイル）と合わせて出力します。

・[帳票資源データの更新があれば保存する] ラジオボタン

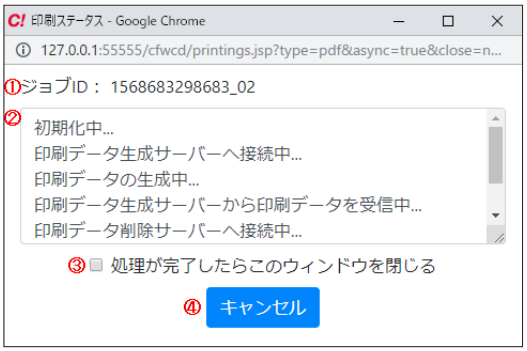
クライアントのローカルに保存された帳票資源データとサーバーの帳票資源データを比較して、サーバーの帳票資源データが更新されていれば、帳票資源データと可変部データを合わせた CCD ファイルをダウンロードし、クライアントのローカルの帳票資源データ（CCD ファイル）を上書き・更新します。サーバーの帳票資源データに更新がなければ、可変部データのみダウンロードし、クライアントのローカルに保存された帳票資源データ（CCD ファイル）と合わせて出力を行います。

※帳票資源データの保存を有効に活用するためには、印刷データ生成スクリプトでのランタイム実行オプションを適切に記述する必要があります。
※詳細は、「19.CCD ファイル保存機能」をご覧ください。

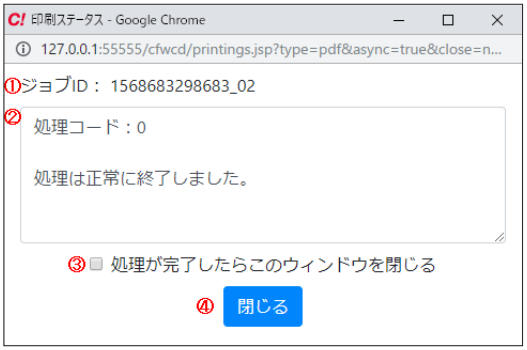
18. 印刷ステータス画面

印刷ステータス画面は印刷の処理状況の確認と印刷のキャンセルを行う画面です。初期設定では印刷処理を開始すると自動で表示されます。なお、印刷コントロールのプロパティ「MinimizedProgressDlg」に"1"が指定されている場合は印刷ステータス画面は表示されません。
※印刷コントロールの詳細は、「10. 印刷コントロール API 仕様」をご覧ください。

図：印刷ステータス画面（印刷中）



図：印刷ステータス画面（印刷終了）



①ジョブ ID
印刷処理ごとに割り当てられる一意のジョブ ID が表示されます。
ジョブ ID はログファイルにも出力されるため、印刷処理が終了したあとでもログファイルから印刷処理の内容を確認することができます。
※詳細は、「21-7. ログ出力」をご覧ください。

②印刷ステータス
現在の印刷処理状況が表示されます。印刷処理が終了（正常終了またはエラー終了）すると処理コードと処理結果のメッセージが表示されます。

③処理が完了したらこのウィンドウを閉じる
印刷処理が終了（正常終了またはエラー終了）すると印刷ステータス画面を自動で閉じます。
「SyncPrintExecuteEx」「SyncPrintExecutePdf」「SyncPreviewExecutePdf」のいずれかで印刷を行っている場合は常に閉じる動作となります。

④キャンセル / 閉じる

印刷処理をキャンセルします。印刷処理中にウィンドウを閉じた場合もキャンセルとなります。「SyncPrintExecuteEx」「SyncPrintExecutePdf」「SyncPreviewExecutePdf」のいずれかで印刷を行っている場合はキャンセルできません。印刷処理が終了したら［キャンセル］ボタンは［閉じる］ボタンに変わります。

19.CCD ファイル保存機能

CCD ファイル保存機能について説明します。

19-1. 印刷コントロールを実行する Web ページの構成

印刷コントロールを実行する Web ページを作成します。この Web ページでは印刷コントロールの情報とメソッド「PrintExecuteEx」「ChangePrintInfo」を実行するコードを記述しています。

```
------------------------------------------------------------------------------
<!DOCTYPE html>
<html lang="ja">
<head>
<meta charset="UTF-8">
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
<script type="text/javascript" src="http://testsv:8080/createform/lib/
cwebclient.min.js?version=12.0.0" charset="utf-8"></script>
<script type="text/javascript">
function KickPrintExecuteEx() {
   cwebclient.CreatePrintingDataScriptUrl = "http://testsv:8080/createform/run";
   cwebclient.DataSendMethod = "POST";
   cwebclient.PrintExecuteEx(form1.stylename.value);
}
function KickChangePrintInfo() {
   cwebclient.ChangePrintInfo(form1.stylename.value);
}
</script>
</head>
<body>
<form name="form1">
   <select name="stylename">
           <option value="syugyo">就業情報</option>
           <option value="time">タイムレポート</option>
           <option value="kotuhi">交通費申請</option>
   </seelct>
   <input type="button" value="印刷" onclick="KickPrintExecuteEx()"/>
   <input type="button" value="印刷設定" onclick="KickChangePrintInfo()"/>
</form>
</body>
</html>
------------------------------------------------------------------------------
```

印刷コントロールのメソッド「PrintExecuteEx」「ChangePrintInfo」の引数は、HTML フォームで選択された Job ファイル名を引数にしています。メソッド「ChangePrintInfo」は引数に指定された帳票の印刷設定画面を表示するので、HTML フォームで"タイムレポート"が選択された状態で、"印刷設定"ボタンをクリックすると、帳票名"time"の印刷設定を行うことができます。メソッド「PrintExecuteEx」の引数に指定された文字列は、印刷データ生成スクリプトに"StyleName=<引数に指定した文字列>"の形で送信されます。上記の例で"タイムレポート"が選択されていれば、印刷データ生成スクリプトには"POST"で"StyleName=time"が送信されます。

19-2. 印刷データ生成スクリプトの構成

前述の「18-2-1. 印刷コントロールを実行する Web ページの構成」で説明した Web ページに対応する印刷データ生成スクリプトの構成例を説明します。

```
import java.io.*;
import java.nio.file.*;

import javax.servlet.*;
import javax.servlet.http.*;

import net.createform.common.*;
import net.createform.sji.*;

public class RunServlet extends HttpServlet {
   @Override
   protected void doPost(HttpServletRequest req, HttpServletResponse res)
            throws ServletException, IOException {
            // 作業ディレクトリ
            File workDir = new File("/opt/resource/cwork");
            // ジョブファイル名
            String jobFileName = req.getParameter("StyleName") + ".sty";
            // 出力 PDF ファイル
            File outputFile = new File(
                     "/usr/local/tomcat/webapps/createform/printings",
                     req.getParameter("OutputFileName")
            );
            // 帳票資源データ設定
            String ccdFileOption = req.getParameter("CcdFileOption");
            // 入力データファイル
            File inputFile = new File("/opt/resource/data", "test.csv");

            RuntimeParam param = new RuntimeParam();
            param.setWorkDir(workDir.getPath());
            param.setStyleFile(jobFileName);
            param.setOutFile(outputFile.getPath());
            param.setStageOption(ccdFileOption);
            param.setDataFile(inputFile.getPath());

            CPrintSTCompress compress = new CPrintSTCompress();
            compress.executeRuntime(param);

            OutputStream out = res.getOutputStream();
            out.write(Files.readAllBytes(outputFile.toPath()));
   }
}
```

ランタイムの実行コマンドの生成では、出力する CCD ファイルの形式を設定する "-c" オプション（JavaAPI を利用する場合は RuntimeParam#setStageOption）を指定します。
"-c" オプションに指定する値は、GET 文字列あるいは POST 文字列から "CcdFileOption" で取得可能です。
※スクリプトの構築例では、POST 文字列で送信されています。
"CcdFileOption" で取得できる文字列は、クライアントの印刷設定の帳票資源データ設定「保存設定」の値が反映されます。クライアントの設定値と、"CcdFileOption" で送信される値の関係は以下の通りです。

帳票資源データ設定「保存設定」: CcdFileOption の値
　"保存しない" : "r"
　"初回のみ保存する" : 初回アクセス時のみ "r"、それ以降は "d"
　"更新があれば保存する" : クライアントが保存している CCD ファイルのタイムスタンプ

このオプションの値で出力される CCD ファイルの形式が変わります。
スクリプトの構築例の場合、クライアントがどのような設定を行っていても対応できる形になっています。ただし、帳票資源データ（CCD ファイル）をクライアントに保存させたくない場合には対応できません。保存させたくない CCD ファイルを出力するためには、"-c" オプションの値に "rb" を指定します。

```
--------------------------------------------------------------------------------
RuntimeParam param = new RuntimeParam();
param.setWorkDir(workDir.getPath());
param.setStyleFile(jobFileName);
param.setOutFile(outputFile.getPath());
param.setStageOption("rb");
param.setDataFile(inputFile.getPath());
--------------------------------------------------------------------------------
```

20. セキュリティ印刷

20-1. 概要

セキュリティ印刷とは、インターネットやイントラネットの Web システムにおいて、伝送中のデータを暗号化し、転送中の印刷データの盗聴を防ぐ機能です。
Create!Form のランタイムから出力された印刷データは暗号化され、クライアント側で復号化して処理します。暗号化のパスワードは印刷ジョブごとに異なるものが自動生成されるほか、伝送データの暗号化と復号化の処理もすべて自動で処理されますので、クライアント側でパスワードを指定する必要はありません。暗号化が可能な対象データは、Web クライアント印刷機能でサポートされている CCD ファイルと PDF ファイルになります。

20-2. 動作環境

20-2-1. サーバーセキュリティモジュール

対応 OS :
- Windows Server 2012 R2
- Windows Server 2016
- Windows Server 2019

※サーバーには Create!Form V12 ランタイム製品が導入されている必要があります。

20-2-2. クライアントセキュリティモジュール

対応 OS :
- Windows 10

※ 64bit の環境のみ対応しています。

20-2-3. 暗号化機能モジュールのインストール

セキュリティ印刷機能を使用する場合、サーバー、クライアント両方に、暗号化機能モジュールを追加でインストールする必要があります。「暗号化機能モジュール」は Create!Form ユーザーサポートサイトよりダウンロード可能です。インストール方法は Create!Form ユーザーサポートサイトをご覧ください。

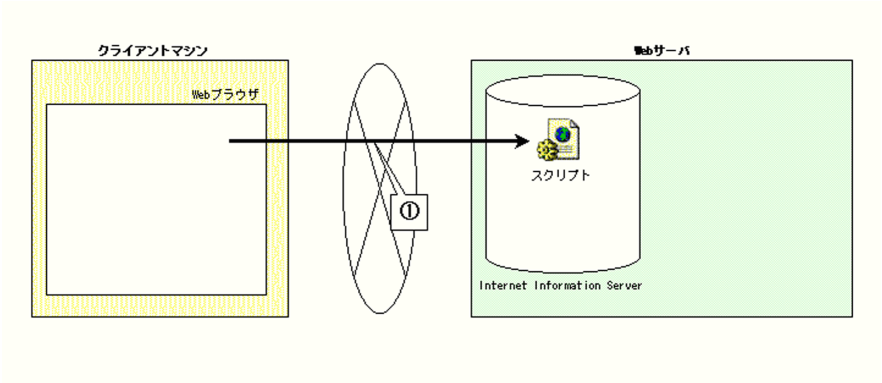
Create!Form ユーザーサポートサイト
https://support.createform.jp/

20-3. 処理の流れ

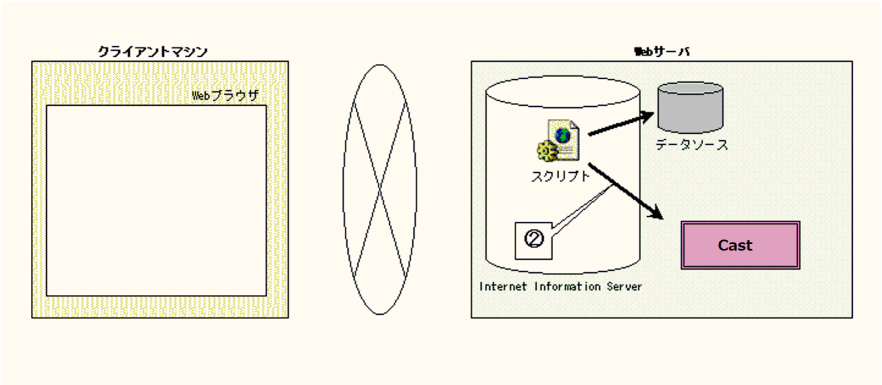
Web サーバーとして IIS、印刷データ生成スクリプトとしてクラシック ASP を利用した例で説明します。

※ Apache Tomcat と Java Servlet を利用した場合も同様の流れで処理されます。

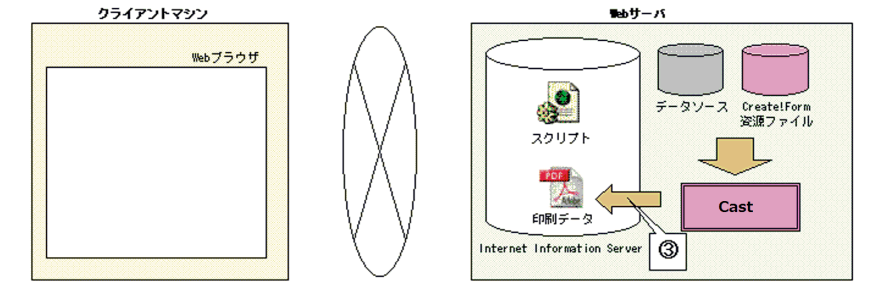
①クライアントは、Cast ランタイム（または PrintStage ランタイム）を実行する印刷データ生成スクリプトへアクセスします。このとき、クライアントの印刷コントロールは、暗号化に必要な暗号化キーを自動生成し、サーバーへ送信します。



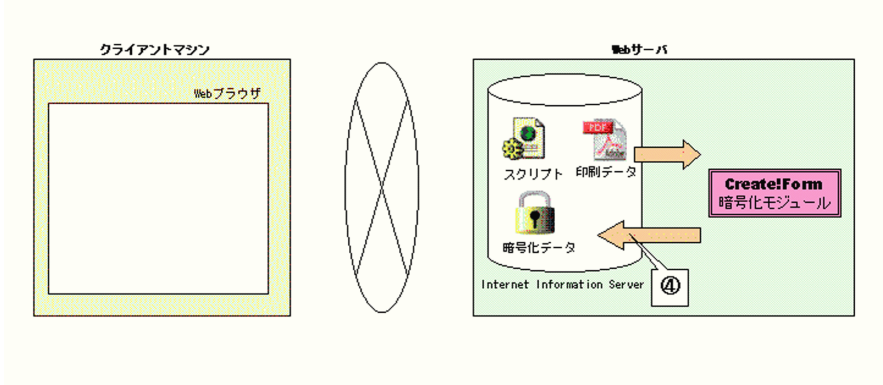
②リクエストを受けた印刷データ生成スクリプトは、ランタイムを実行します。



③ランタイムは、サーバー上の帳票資源ファイルを使用して印刷データを生成します。



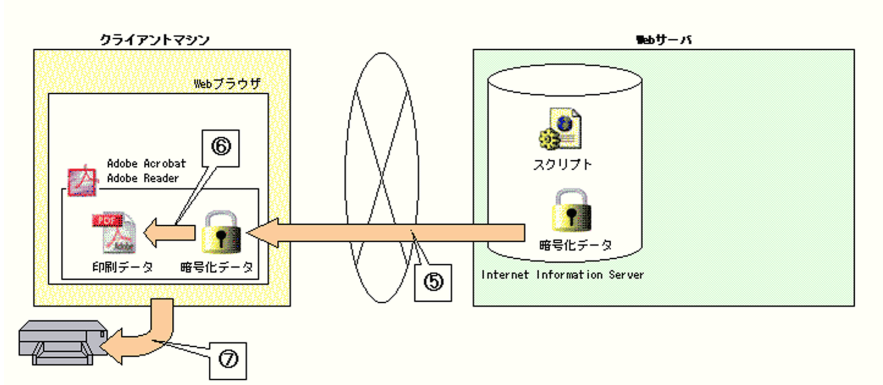
④印刷データ生成スクリプトは、暗号化機能モジュールを実行し、印刷データを暗号化します。



⑤暗号化された印刷データは、クライアントの印刷コントロールへ送信されます。

⑥印刷コントロールは、自動生成した復号化キーを使用して暗号化された印刷データを復号化します。

⑦印刷コントロールは、印刷データの印刷や PDF プレビューの表示を行います。



20-4. 送信クエリ文字列の予約キーワード

プロパティ「FormRequestArray」やGET文字列を使用して、プロパティ「CreatePrintingDataScriptUrl」で指定したスクリプトに対して、データ送信を行うことが可能ですが、使用するメソッドによって予約されたキーワードがあります。
このキーワードを設定してしまうと、印刷コントロールは正常に処理されません。

使用するメソッド	予約キーワード
PrintExecutePdf (SyncPrintExecutePdf)	"OutputFileName"、"enckey"
PreviewExecutePdf (SyncPreviewExecutePdf)	"OutputFileName"、"enckey"
PrintExecuteEx (SyncPrintExecuteEx)	"OutputFileName"、"StyleName"、"CcdFileOption"、"enckey"

20-5. 使用方法

20-5-1. 暗号化機能モジュール

サーバー

コアモジュール	CEncEdit.dll
インターフェースモジュール	
実行ファイル	CEnc.exe
COM コンポーネント	CEncCOM.dll
Java ネイティブ	cenc.jar CEncJNI.dll
.Net	CEncLib.dll
セキュリティログ設定モジュール	WCSecurityLog.exe
暗号化モジュール	CfSecurity_64_2.dll、CfSecurity_2.dll

クライアント

印刷コントロール	cwebclient.min.js
暗号化モジュール	CfSecurity_64_2.dll、CfSecurity_2.dll

20-5-2. 構築例

暗号化機能モジュールは Create!Form ランタイムの実行後に呼び出し、ランタイムから出力した印刷データを暗号化します。
インターフェースモジュールとしては「実行ファイル」「COM コンポーネント」「Java ネイティブ」が提供されていますので、サーバー環境に合わせて利用してください。
ここでは、Web クライアント印刷のブラウザー任意指定印刷のサーバースクリプトを例に説明します。その他、クライアント側の HTML ファイルの記述や環境設定等は、通常の Web クライアント印刷と同様です。

クライアント側 "index.html"

```
<!DOCTYPE html>
<html lang="ja">
<head>
<meta charset="UTF-8">
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
<title>ブラウザー任意指定印刷</title>
<script type="text/javascript" src="http://testsv:8080/createform/lib/
cwebclient.min.js?version=12.0.0" charset="utf-8"></script>
<script type="text/javascript">
function KickPrintExecuteEx() {
   cwebclient.SecurityOption = 1;
   cwebclient.CreatePrintingDataScriptUrl = "http://testsv:8080/createform/run";
   // 帳票名（Job ファイル名）を引数に指定します。
   cwebclient.PrintExecuteEx("SchoolLife");
}
</script>
</head>
<body>
   <input type="button" value="ブラウザー任意指定印刷"
          onclick="KickPrintExecuteEx()" />
</body>
</html>
```

サーバー側 "run.asp"（例 1：クラシック ASP）

```
<%@ Language=VBScript%>
<%
Dim objPrintST
Dim strCurPath      ' 実行ディレクトリ
Dim strTextFile     ' テキストファイル名
Dim strCWorkPath    ' Create!Form 作業ディレクトリ
Dim strCCDPath      ' CCD ファイルパス
Dim strCmd          ' PrintStageWeb ランタイム実行コマンドパラメーター
Dim ErrNum          ' 戻り値
strCurPath = "c:¥resource"
'============================================================
' PrintStageWeb ランタイム用テキストファイル作成
'============================================================
' 通常、ここで動的にテキストファイルを生成しますが、
' このサンプルでは既存のファイルを指定します。
strTextFile = strCurPath & "¥data¥test.csv"
'============================================================
' PrintStageWeb ランタイムの実行
'============================================================
Set objPrintST = CreateObject("CPrintSTCOM.CPrintST")
strCWorkPath = strCurPath & "¥cwork"
strCCDPath = "c:¥createform¥printings¥" & Request.Form("OutputFileName")

' コマンドの作成
strCmd = "-D" & strCWorkPath &  _
   " -s" & Request.Form("StyleName") & ".sty" & _
   " -o" & strCCDPath & _
   " -c" & Request.Form("CcdFileOption") & _
   " " & strTextFile
' PrintStageWeb ランタイム実行（第二パラメーターの 8705 は、
' CPW_HIDE | ERW_HIDE | ERL_ENABLE の指定と同じです。）
ErrNum = objPrintST.CPCompressExec( 0, 8705, "", strCmd )
' オブジェクトの解放
Set objPrintST = Nothing
' 印刷データの暗号化
Set objEnc = Server.CreateObject("CEncCOM.EncCOM")
RetVal = objEnc.Enc("key="&Request.Form("enckey")& _
   "&inputfile=" & strCCDPath & _
   "&outputfile=" & strCCDPath )
If RetVal <> 0 Then
   ' エラー処理
End If
Set objEnc = Nothing
' 生成された暗号化ファイルをリダイレクトで返す
Response.Redirect("printings/" & Request.Form("OutputFileName"))
%>
```

このクラシック ASP の例では、COM コンポーネントを利用して暗号化機能モジュールを実行しています。COM コンポーネントの Enc メソッドには実行引数を渡します。
※実行引数の詳細は、「20-5-3. インターフェースモジュール」をご覧ください。
暗号化機能モジュールの実行引数に渡す暗号化キーはPOST 文字列の"enckey"から取得します。
この POST 文字列は、クライアントの印刷コントロールで自動生成されてサーバーへ送信されます。

サーバー側 "CEncServlet.java" (例 2 : Java Servlet)

```
import java.io.*;
import java.nio.file.*;

import javax.servlet.*;
import javax.servlet.http.*;

import net.createform.*;
import net.createform.common.*;
import net.createform.sji.*;

public class RunServlet extends HttpServlet {
   @Override
   protected void doPost(HttpServletRequest req, HttpServletResponse res)
            throws ServletException, IOException {
            // 作業ディレクトリ
            File workDir = new File("/opt/resource/cwork");
            // ジョブファイル名
            String jobFileName = req.getParameter("StyleName") + ".sty";
            // 出力 PDF ファイル
            File outputFile = new File(
                     "/usr/local/tomcat/webapps/createform/printings",
                     req.getParameter("OutputFileName")
            );
            // 帳票資源データ設定
            String ccdFileOption = req.getParameter("CcdFileOption");
            // 入力データファイル
            File inputFile = new File("/opt/resource/data", "test.csv");

            RuntimeParam param = new RuntimeParam();
            param.setWorkDir(workDir.getPath());
            param.setStyleFile(jobFileName);
            param.setOutFile(outputFile.getPath());
            param.setStageOption(ccdFileOption);
            param.setDataFile(inputFile.getPath());

            CPrintSTCompress compress = new CPrintSTCompress();
            compress.executeRuntime(param);

            CEncJNI cenc = new CEncJNI();
            cenc.Enc("key=" + req.getParameter("enckey") +
                     "&inputfile=" + outputFile.getPath() +
                     "&outputfile=" + outputFile.getPath());

            OutputStream out = res.getOutputStream();
            out.write(Files.readAllBytes(outputFile.toPath()));
   }
}
```

この Java Servlet の例では、Java ネイティブを利用して暗号化機能モジュールを実行しています。Java ネイティブの Enc メソッドには実行引数を渡します。
※実行引数の詳細は、「20-5-3. インターフェースモジュール」をご覧ください。
暗号化機能モジュールの実行引数に渡す暗号化キーはPOST 文字列の"enckey"から取得します。
この POST 文字列は、クライアントの印刷コントロールで自動生成されてサーバーへ送信されます。

20-5-3. インターフェースモジュール

実行ファイル（CEnc.exe）

プログラムフォルダーに配置されています。

（書式）

CEnc［引数］

（引数）

文字列型：クライアントから送信された暗号化キー、暗号化する入力ファイルパス、暗号化した出力ファイルパスを以下のように＆（アンパサンド）で結合して指定します。
各値は”（ダブルクォート）で括ります。

```
key = ″暗号化キー″ & inputfile = ″暗号化する入力ファイルパス″
            & outputfile = ″暗号化した出力ファイルパス″
```

（例：コマンドライン）

```
CEnc key="ABCDEFG"&inputfile="C:¥data¥input.ccd"&outputfile="C:¥data¥output.ccd"
```

COM コンポーネント（CEncCOM.dll）

プログラムフォルダーに配置されています。
Windows の環境変数 PATH に Create!Form のプログラムフォルダーへのパスを設定してください。

（書式）

```
int CEnc(BSTR)
```

（引数）

文字列型：クライアントから送信された暗号化キー、暗号化する入力ファイルパス、暗号化した出力ファイルパスを以下のように＆（アンパサンド）で結合して指定します。
各値は”（ダブルクォート）で括ります。

```
key = ″暗号化キー″ & inputfile = ″暗号化する入力ファイルパス″
            & outputfile = ″暗号化した出力ファイルパス″
```

（戻り値）

正常終了：0
異常終了：0 以外

（例：クラシック ASP）

```
Dim objEnc
Set objEnc = Server.CreateObject("CEncCOM.EncCOM")
RetVal = objEnc.Enc("key=""&Request.Form("enckey")&"""&
   "&inputfile=""C:¥data¥"&Request.Form("OutputFileName")&"""&
   "&outputfile=""C:¥data¥"&Request.Form("OutputFileName")&""")
If RetVal <> 0 Then
   ' エラー処理
End If
Set objEnc = Nothing
```

Java ネイティブ（cenc.jar）

プログラムフォルダーの [lib] フォルダーに配置されています。
JAR ファイルにはクラスパスを設定し、java.library.path および Windows の環境変数 PATH に Create!Form のプログラムフォルダーへのパスを設定してください。

（書式）
```
int Enc(String)
```
（引数）

文字列型：クライアントから送信された暗号化キー、暗号化する入力ファイルパス、暗号化した出力ファイルパスを以下のように＆（アンパサンド）で結合して指定します。

```
key = 暗号化キー & inputfile = 暗号化する入力ファイルパス
            & outputfile = 暗号化ファイルパス
```
（戻り値）

正常終了：0
異常終了：0 以外

（例：Java）
```
import net.createform.CEncJNI;
   ...
CEncJNI cenc = new CEncJNI();
int ret = cenc.Enc("key=" + req.getParameter("enckey") +
   "&inputfile=C:¥¥data¥¥" + req.getParameter("OutputFileName") +
   "&outputfile=C:¥¥data¥¥" + req.getParameter("OutputFileName"));
```

.Net（CEncLib.dll）

プログラムフォルダーの [lib] フォルダーに配置されています。
この DLL ファイルを Visual Studio .NET の［プロジェクト］メニューの［参照の追加］から参照設定に追加してください。また、動作には [.Net Framework] が必要となります。
Windows の環境変数 PATH に Create!Form のプログラムフォルダーへのパスを設定してください。

（書式）
```
int Execute(CEncParam);
```
（引数）

CEncParam 型：クライアントから送信された暗号化キー、暗号化する入力ファイルパス、暗号化した出力ファイルパスを CEncParam 型に設定します。

（戻り値）

正常終了：0
異常終了：0 以外

（例）
```
using InfotecArchitects.CreateForm.Runtime;
using InfotecArchitects.CreateForm.Runtime.Enc;
   ...
CEncParam param = new CEncParam();
param.Key = "39c4･･･17d8efb0203010001";
param.InputFile = "inputFileName";
param.OutputFile = "outputFileName";

EncRuntime enc = new EncRuntime();
int ret = enc.Execute(param);
```

20-6. セキュリティログの出力

20-6-1. セキュリティログの種類

エラーログ
　暗号化機能モジュールでエラーが発生した場合にエラーの内容が出力されます。
　初期設定ではログファイル名は [ErrLog.txt] です。
実行ログ
　暗号化機能モジュールの処理の開始から終了までのログが出力されます。
　初期設定ではログファイル名は [IvtLog.txt] です。
実行詳細ログ
　暗号化機能モジュールの処理の開始から終了までの詳細ログが出力されます。
　初期設定ではログファイル名は [IvtDLog.txt] です。

セキュリティログの出力先は初期設定ではユーザーデータフォルダーの「log」フォルダーに出力されます。ユーザーデータフォルダーはマネージャーの [ヘルプ]-[バージョン情報]-[バージョン情報詳細] から確認できます。

20-6-2. セキュリティログの設定

セキュリティログの設定は、プログラムフォルダーのセキュリティログ設定モジュール [WCSecurityLog.exe] を使用します。
エラーログの出力の有無、出力ファイル名、ログの最大ファイルサイズを指定します。
実行ログ、実行ログ詳細も同様に指定します。

20-6-3. セキュリティログのメッセージ一覧

エラー番号：エラー内容
4101: メモリが不足しています。

4201: ファイルを作成できません。
　ファイルのアクセス権限を確認してください。

4202: ファイルを参照できません。
　ファイルのアクセス権限を確認してください。

4203: ファイルを操作できません。
　ファイルのアクセス権限を確認してください。

4204: ファイル内容がありません。
　ファイル転送中にエラーが起きた可能性があります。

4301: キー生成に失敗しました。

4302: 暗号化に失敗しました。

4303: 復号化に失敗しました。

4401: キー生成に失敗しました。

4402: 暗号化に失敗しました。

4403: 復号化に失敗しました。

5101: ログを出力できません。
　ログファイルのアクセス権限を確認してください。

5102: 詳細ログを出力できません。
　ログファイルのアクセス権限を確認してください。

21.Windows サービス

21-1. 概要

Web クライアント印刷の機能はクライアント側の以下の Windows サービス上で動作します。

Create!Form WebClient Service V12

そのため、この Windows サービスが開始している場合のみ Web クライアント印刷を行うことができます。デフォルトでは自動起動するように構成されてるため、Windows の起動時に自動で開始状態となります。

※ Windows サービスは Windows の [コントロールパネル]-[管理ツール]-[サービス] から確認できます。

導入されている Windows サービスの製品バージョンは以下より確認が可能です。
[Windows のスタートメニュー]-[すべてのアプリ]-[Create!Form V12]-[Web クライアントサービス バージョン]
なお、上記でバージョン確認が可能なのは V12.1.0 以降となります。

21-2. ログオンアカウント

Windows サービスにはログオンアカウントの設定があります。
ログオンアカウントは、サービスをどのユーザーのアカウントで実行させるかという設定になりますが、デフォルトでは「ローカルシステムアカウント」という特別なユーザーが設定されています。また、必要に応じて任意のアカウントをログオンユーザーとすることもできます。

以下にそれぞれのログオンユーザーの特徴について記述します。

21-2-1. ローカルシステムアカウント

サービスを実行させるための Windows が用意している特別なユーザー（"SYSTEM" と呼ばれるビルトインユーザー）です。このユーザーは Windows の Administrators グループに所属しているため、Administrators グループへのアクセス権限が与えられたプリンターやログオン中のユーザーのプリンターを利用することができます。一般的なプリンターは Administrators グループへのアクセス権限がデフォルトで与えられているため、特にプリンターのセキュリティ設定を変更することなく利用することができます。

なお、ローカルシステムアカウントを利用した場合は印刷時にダイアログが表示されるプリンター（Adobe Acrobat に付属の PDF プリンターの "Adobe PDF" など）もサービス設定ファイルを変更することで利用できます。
※サービス設定ファイルの詳細は、「21-5. サービス設定ファイル」をご覧ください。

21-2-2. 任意のアカウント

プリンターにアクセスできるユーザーが制限されている場合、ローカルシステムアカウントではプリンターへのアクセスが制限されてしまいます。この場合、任意のアカウントをサービスのログオンユーザーとすることでプリンターにアクセスできるようになります。

<< 注意 >>
任意のアカウントの場合、ローカルシステムアカウントのように印刷時にダイアログが表示されるプリンターは利用することができません。

21-3. 印刷スプーラー

サービスを起動すると印刷スプーラー（Print Spooler サービス）の再起動が行われます。
これはサービスからプリンターを正しく認識するためです。
Windows の起動直後に自動で印刷などを行っている場合はサービスの起動方法を手動起動や遅延開始するなどの構成変更が必要となりますので注意してください。

21-4. プリンタードライバー

プリンタードライバーはローカルプリンターとネットワークプリンターに対応しています。
サービスを起動後にプリンタードライバーの追加や削除を行った場合は、サービスの再起動後に内容が反映されます。

21-5. サービス設定ファイル

21-5-1. cwebclient.properties

サービス設定ファイルを編集することでサービスの設定を変更することができます。
サービス設定ファイルはユーザー設定（共通）フォルダーの「cwebclient.properties」です。
フォーマットは「key=value」の形式で記述し、文字コードは「UTF-8」を使用します。
サービス設定ファイルの内容はサービスの再起動後に反映されます。

cwebclient.properties

```
log.append=true
log.enable=true
log.directory=
log.filename=cwebclient.log
log.level=debug
cleaner.period=3600000
cleaner.expiration=86400000
service.idletimeout=86400000
connection.timeout=180000
connection.readtimeout=86400000
pdf.interactive=false
```

log.append
サービスのログの追記出力を設定します。
デフォルトは "true" です。
追記出力しない場合はサービスを起動するごとにログが初期化されます。

log.enable
サービスのログの出力有無を設定します。
デフォルトは "true" です。
※ログの詳細は、「21-7. ログ出力」をご覧ください。

log.directory
サービスのログの出力先を設定します。
デフォルトは未指定です。
未指定の場合はユーザーデータフォルダーの「log」フォルダーに出力されます。
パスの区切り文字は "/" または "¥¥" で指定してください。

log.filename
サービスのログのファイル名を設定します。
デフォルトは "cwebclient.log" です。

log.level
サービスのログのログレベルを設定します。
デフォルトは "debug" です。
ログレベルは "debug"、"info"、"warning"、"error" のいずれかを設定できます。
ログレベルは設定したレベル以上のログがすべて出力されます。
例えば、"info" を設定すると "info"、"warning"、"error" のログが出力対象となります。

cleaner.period
一時ファイルを自動削除する間隔を設定します。
デフォルトは "3600000" です。（単位：ミリ秒）
※一時ファイルの詳細は、「21-8. 一時ファイル」をご覧ください。

cleaner.expiration
一時ファイルを自動削除の対象にする経過時間を設定します。
デフォルトは "86400000" です。（単位：ミリ秒）
※一時ファイルの詳細は、「21-8. 一時ファイル」をご覧ください。

service.idletimeout
サービスとの接続を維持する時間を設定します。
デフォルトは "86400000" です。（単位：ミリ秒）

connection.timeout
サーバーへの接続タイムアウトの時間を設定します。
デフォルトは "180000" です。（単位：ミリ秒）

connection.readtimeout
サーバーへの接続後にレスポンスが返されるまでのタイムアウト時間を設定します。
デフォルトは "86400000" です。（単位：ミリ秒）

pdf.interactive
サービスのログオンアカウントがローカルシステムアカウントの場合に印刷時のダイアログ表示を許可するかを設定します。
デフォルトは"false"です。
許可した場合、印刷時にダイアログが表示されるプリンター（Adobe Acrobatに付属のPDFプリンターの"Adobe PDF"など）を利用することができます。許可しない場合は印刷時にダイアログが表示されてしまうと印刷が行われない状態となります。

21-5-2. HTTP設定ファイル

HTTP設定ファイルを編集することで通信時の設定を変更することができます。
HTTP設定ファイルはユーザー設定（共通）フォルダー¥confの「cwebclientHttp.properties」です。
フォーマットは「セクション」と「key=value」の形式で記述し、文字コードは「UTF-8」を使用します。
サービス設定ファイルの内容はサービスの再起動後に反映されます。
以下のフォーマットでファイルを作成し、各クライアント環境のユーザー設定（共通）フォルダー¥confへ配置してください。

cwebclientHttp.properties

```
[requestheader]
# 設定するHTTPリクエストヘッダを記載します。
request=header
date=20231006
...
```

21-6. 接続ポート番号

Webクライアント印刷はWindowsサービス上で動作しますが、クライアントの印刷コントロールとサービス間の通信にはHTTPを利用しています。
デフォルトではサービスの接続ポートは"55555"ですが、設定を変更することで接続ポート番号を変更することもできます。設定を変更する場合はサービスを停止した状態で以下のファイルをエディターで開いてください。

```
C:¥Program Files¥Infotec¥CreateForm¥12¥vender¥Jetty¥start.d¥http.ini
```

※製品をデフォルトのインストール先にインストールした場合のパスとなります。プログラムフォルダーを変更している場合はそのフォルダーに読み替えてください。

「jetty.port」の値を変更します。

```
## HTTP port to listen on
jetty.port=55555
```

接続ポート番号を"55555"以外に変更した場合は印刷コントロールAPIの「ServicePort」の値も変更する必要があります。
※詳細は、「10. 印刷コントロール API仕様」をご覧ください。

21-7. ログ出力

Web クライアント印刷の処理が行われるとサービスのログが出力されます。
ログはサービスの起動時や印刷の実行時に出力されます。
デフォルトではユーザーデータフォルダーの「log」フォルダーに「cwebclient.log」ファイルが出力されます。フォーマットは「CSV」、文字コードは「UTF-8」です。

cwebclient.log

```
--------------------------------------------------------------------------------
1 行目 : # Create!Form WebClient Service Ver. 製品バージョン (x64)
2 行目 : yyyy/mm/dd,hh:mm:ss.sss, ログレベル , ジョブ ID, 結果コード , メッセージ
...( 以降は 2 行目と同じ )...
--------------------------------------------------------------------------------
```

21-8. 一時ファイル

サービスの起動時や印刷の実行時に以下のテンポラリフォルダーに一時ファイルを生成します。

C:¥ProgramData¥Infotec¥CreateForm¥12¥temp¥cfwcd

※製品をデフォルトのインストール先にインストールした場合のパスとなります。テンポラリフォルダーを変更している場合はそのフォルダーに読み替えてください。

ウィルス対策ソフトなどの影響でテンポラリフォルダーへのファイルの読み書きが正常に行えない場合、Web クライアント印刷は正常に動作しません。この場合はテンポラリフォルダーを検知対象から除外するようにしてください。

21-9. PDF ファイルの関連付け

PDF 非表示印刷を利用する場合は PDF ファイルを Adobe Acrobat や Adobe Acrobat Reader に関連付ける必要がありますが、関連付けの設定についてはサービスの再起動後に内容が反映されます。

22. プロキシサーバーの利用

22-1. 概要

クロスブラウザー版 Web クライアント印刷は以下の Windows サービス上で動作します。

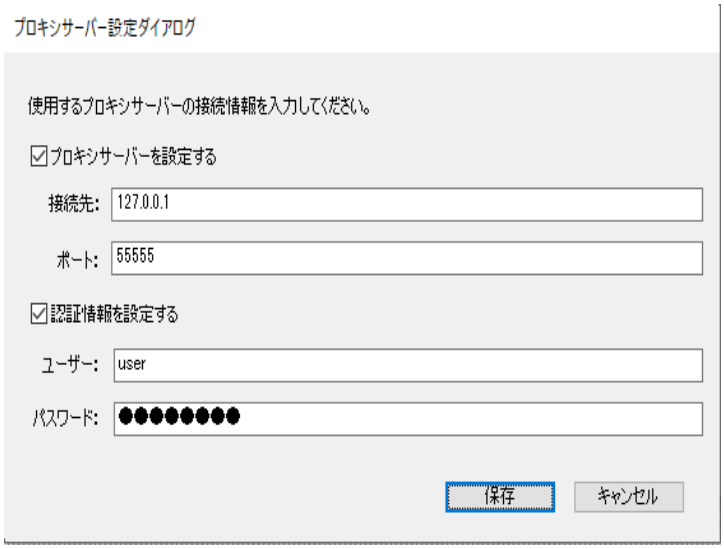
Creeate!Form WebClient Service V12

そのため、プロキシサーバーの設定はブラウザーや OS の設定ではなく、製品同梱のプロキシサーバー設定より行います。

22-2. プロキシサーバーの設定

<< 準備 >>
Windows のスタートメニューより、[すべてのアプリ]-[Create!Form V12]-[プロキシサーバー設定] を選択し、プロキシサーバー設定ダイアログを起動します。

図：プロキシサーバー設定ダイアログ



「プロキシサーバーを設定する」を有効にし、プロキシサーバーの接続先、ポート情報を入力してください。プロキシサーバーに認証設定がある場合は、「認証情報を設定する」を有効にし、ユーザー名、パスワードを入力してください。

入力が完了しましたら、保存ボタンを押下して「Create!Form WebClient Service V12」サービスを再起動してください。

※「Create!Form WebClient Service V12」サービスを再起動しない限り、設定は反映されません。

22-3. 制限事項

・プロキシサーバー設定はクライアント環境につき、1 つのみ設定が可能です。複数の接続先を管理することはできません。

・プロキシサーバー設定では、特定の接続先のみプロキシサーバーを経由しないような例外設定をすることはできません。

・プロキシサーバー設定ダイアログで使用可能な文字、文字数は以下となります。
接続先：100 文字以内の半角英数字、「-」、「.」
ポート：5 桁以内の半角数字
ユーザー：255 文字以内の半角英数字、記号
パスワード：255 文字以内の半角英数字、記号

・使用可能な認証設定は BASIC 認証または Digest 認証となります。

23. クライアント認証

23-1. 概要

クロスブラウザー版 Web クライアント印刷は以下の Windows サービス上で動作します。

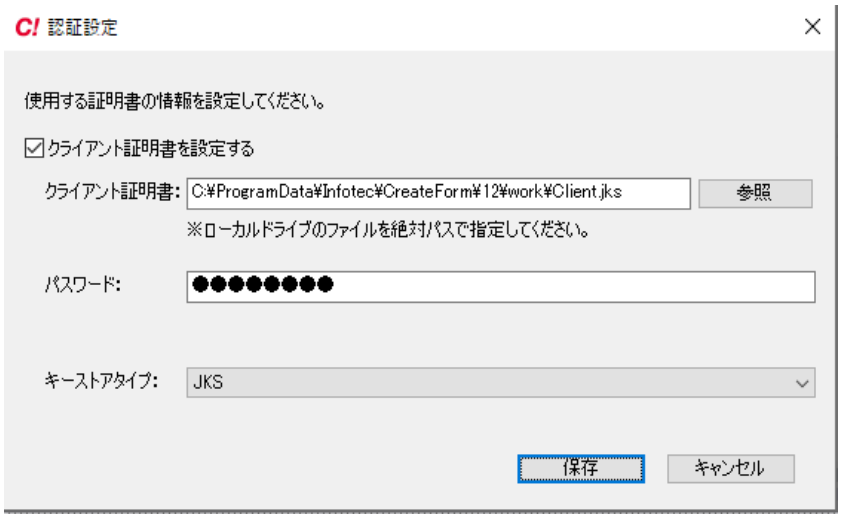
Creeate!Form WebClient Service V12

そのため、クライアント認証用の証明書の設定はブラウザーや OS の設定ではなく、製品同梱の認証設定より行います。

23-2. 認証設定

<< 準備 >>
Windows のスタートメニューより、[すべてのアプリ]-[Create!Form V12]-[認証設定] を選択し、認証設定ダイアログを起動します。

図：認証設定ダイアログ



「クライアント証明書を設定する」を有効にし、クライアント証明書、パスワード、キーストアタイプを入力してください。

入力が完了しましたら、保存ボタンを押下して「Create!Form WebClient Service V12」サービスを再起動してください。

※「Create!Form WebClient Service V12」サービスを再起動しない限り、設定は反映されません。

23-3. 制限事項

・クライアント証明書の設定はクライアント環境につき、1 つのみ設定が可能です。複数の証明書を管理することはできません。複数のクライアント証明書を管理したい場合は、複数の証明書を管理することが可能な JKS 形式に変換してご利用ください。

・認証設定ダイアログで使用可能な文字、文字数は以下となります。
　クライアント証明書：256 文字以内の文字列
　パスワード：255 文字以内の半角英数字、記号

・クライアント証明書へのパスにサロゲート文字は使用できません。

・クライアント証明書へのパスにショートファイルネームは使用できません。

・証明書の指定は絶対パスでの指定が必要です。ネットワークドライブは利用できません。

・リムーバブルディスク内の証明書は指定できません。

・使用可能な証明書形式は JKS または PKCS#12 形式となります。

24.HTTP リクエストヘッダ

24-1. 概要

クロスブラウザー版 Web クライアント印刷は以下の Windows サービス上で動作します。

Creeate!Form WebClient Service V12

そのため、CreatePrintingDataScriptUrl に指定した帳票サーバーと通信する際に HTTP リクエストヘッダを設定するためには、製品側での設定が必要となります。

24-2.HTTP リクエストヘッダの設定

HTTP リクエストヘッダの設定方法として、以下の 2 通りの方法があります。

SetRequestHeader による指定

API を使用して、HTTP リクエストヘッダの付与を行います。SetRequestHeader に設定したい HTTP リクエストヘッダ情報を設定します。API による設定のため、リクエストごとに動的な HTTP リクエストヘッダを指定可能です。
設定の詳細については 10-2.API 仕様をご覧ください。

設定ファイルによる指定

HTTP 設定ファイルにあらかじめ付与したい HTTP リクエストヘッダ情報を定義しておく方法です。
HTTP 設定ファイルを配布することで、複数のクライアント環境への適用が可能となります。
HTTP 設定ファイルでの定義方法については、21-5-2. HTTP 設定ファイルをご覧ください。

なお、HTTP 設定ファイルから HTTP リクエストヘッダ情報を反映させるためには、「Create!Form WebClient Service V12」サービスを再起動する必要があります。

24-3. 同一キーの扱い

同じキーが複数設定された場合は、カンマで連結された状態で送信されます。これは SetRequestHeader、HTTP 設定ファイルでそれぞれ設定した場合も同様です。

例：

SetRequestHeader の設定値

```
request=header
```

HTTP 設定ファイルの設定値

```
REQUEST=param
```

送信される値

```
request=header,param
```

24-4. 制限事項

・HTTP リクエストヘッダのキーに使用可能な文字は、半角英数字、「-」となります。

・SetRequestHeader を利用して HTTP リクエストヘッダを設定する場合、バリューには URL エンコードした値を設定する必要があります。HTTP 設定ファイルには URL エンコードは不要です。

・送信されるバリューは URL エンコードされた状態で送信されます。

・HTTP リクエストヘッダのキーでは、大文字、小文字の区別はされず、同一のキーとして扱われます。

・SetRequestHeader、HTTP 設定ファイルのそれぞれから同一のキー、バリューを設定した場合も、同じ値がカンマで連結された状態で送信されます。

25. ActiveX 版からの移行

25-1. 概要

旧バージョンの Web クライアント印刷には印刷コントロールとして「CWebClient.ocx」を利用する ActiveX 版があります。この ActiveX 版の印刷コントロールは現在のバージョンではサポートされないため、「cwebclient.min.js」の印刷コントロールへ移行するための作業が必要となります。

25-2. 移行手順

ActiveX 版からの移行は以下の流れで行います。

クライアントマシン側の作業

1. Web クライアント印刷のインストーラーを実行します。

2. マシンを再起動後、Web クライアント印刷の Windows サービスが開始状態となっていることを確認します。

サーバーマシン側の作業

3. Cast ランタイムや PrintStageWeb ランタイムをインストールします。

4. 印刷コントロール API(cwebclient.min.js) をプログラムフォルダーの「lib」フォルダーからコピーしてクライアントマシンの Web ブラウザーから参照できるように Web サーバー上に配置します。

5. Web サーバー上にある印刷要求を行う Web ページをエディターで開きます。

6. 印刷コントロール API(cwebclient.min.js) が実行されるように object タグから script タグへ変更します。

変更前

```
<object id="obj" classid="clsid:F58DEEA6-72E2-4811-8BE4-47796A7E804B"
   codebase="http://testsv/createform/lib/CWebClient.ocx#version=11,2,0,0">
</object>
```

変更後

```
<script type="text/javascript"
   src="http://testsv/createform/lib/cwebclient.min.js?version=12.0.0"
   charset="utf-8"></script>
```

7. 印刷コントロールオブジェクトの変数名を cwebclient に変更します。

変更前

```
<script type="text/javascript">
function KickPrintExecuteEx() {
   obj.CreatePrintingDataScriptUrl = "http://testsv/createform/run.asp";
   // 帳票名（Job ファイル名）を引数に指定します。
   obj.PrintExecuteEx("SchoolLife");
}
</script>
```

変更後

```
<script type="text/javascript">
function KickPrintExecuteEx() {
   cwebclient.CreatePrintingDataScriptUrl = "http://testsv/createform/run.asp";
   // 帳票名（Job ファイル名）を引数に指定します。
   cwebclient.PrintExecuteEx("SchoolLife");
}
</script>
```

なお、変数名を変更するのではなく cwebclient を参照するように変数を定義することもできます。

変更後

```
<script type="text/javascript">
function KickPrintExecuteEx() {
   var obj = cwebclient;
   obj.CreatePrintingDataScriptUrl = "http://testsv/createform/run.asp";
   // 帳票名（Job ファイル名）を引数に指定します。
   obj.PrintExecuteEx("SchoolLife");
}
</script>
```

Create!Form 12

Web クライアント印刷 第 4 版

発行日 2023 年 10 月
発行者 インフォテック株式会社
〒 160-0023 東京都新宿区西新宿 7-5-25