

Web クライアント印刷

1. 概要	3
2. PDF 表示印刷とは	4
3. PDF 非表示印刷とは	6
4. ブラウザ任意指定印刷とは	9
5. Web サーバの基本設定	12
6. クライアントの基本設定	14
6-1. IE のセキュリティ設定	14
7. PDF 表示印刷の構築例	16
8. PDF 非表示印刷の構築例	18
9. ブラウザ任意指定印刷の構築例	22
10. 印刷コントロール API 仕様	26
10-1. 起動方法	26
10-2. クラス情報	26
11. 印刷コントロール 結果コード一覧	30
12. 送信クエリ文字列の予約キーワード	37
13. 印刷データを削除する	38
13-1. バイナリを扱えないスクリプト (ASP 等) を使用している場合	38
13-2. バイナリを扱うスクリプト (Perl 等) を使用している場合	38
14. スクリプトヘッダを送信する	40
14-1. POST 文字列	40
14-2. GET 文字列	40
14-3. セッション変数	41
15. WebClient ランタイムについて	42
15-1. 詳細な印刷情報の設定	42
15-2. CCD ファイルを使用した印刷	42
16. 印刷設定ダイアログの起動方法	43
16-1. 印刷コントロールのメソッド「ChangePrintInfo」からの起動	43
16-2. 導入ディレクトリ直下の「PrtSTConf.exe」からの起動	43
17. 印刷設定ダイアログの画面説明	44
17-1. 印刷コントロールのメソッド「ChangePrintInfo」からの起動	44
17-2. 導入ディレクトリ直下の「PrtSTConf.exe」からの起動	48
18. CCD ファイル保存機能	52
18-1. 印刷コントロールを起動する Web ページの構成	52
18-2. 印刷データ生成スクリプトの構成	53
19. セキュリティ印刷	55
19-1. 概要	55
19-2. 導入環境	55
19-2-1. サーバセキュリティモジュール	55
19-2-2. クライアントセキュリティモジュール	55

19-3. 処理の流れ	55
19-4. 送信クエリ文字列の予約キーワード	57
19-5. 使用方法	57
19-6. ログの出力	63
19-6-1. ログの種類	63
19-6-2. ログの設定	63
19-6-3. メッセージ一覧	63

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（Internet Explorer（以下、IE）等）で印刷処理を行うことです。

この「Web ブラウザで印刷処理」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。

<< 参考 >>

Web ブラウザからの指示で印刷する場合でも、サーバから LAN 上にあるプリンタ（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）に印刷する場合は、Web クライアント印刷ではなくサーバ印刷となりますので、FormPrintStage ランタイム、または FormPrint ランタイムの印刷処理で行ってください。

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、あるいは Adobe Reader の機能で印刷する方法です。

この印刷方法は、サーバ側で作成された PDF ファイルを、ブラウザのプラグイン機能で Adobe Acrobat、あるいは Adobe Reader を起動して表示、印刷するだけです。特別な処理を実装する必要はありません。

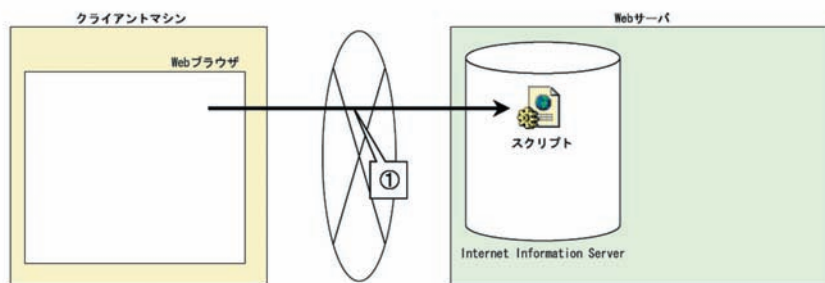
<< 注意 >>

印刷元となるデータは PDF ファイルとなりますので、サーバ側では PDF ファイルを生成する FormCast ランタイム（または FormCollect ランタイム）が必要です。

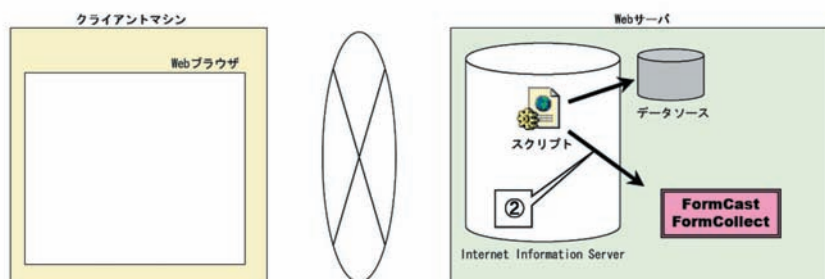
<< 注意 >>

Adobe Acrobat、あるいは Adobe Reader のバージョン 6 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。

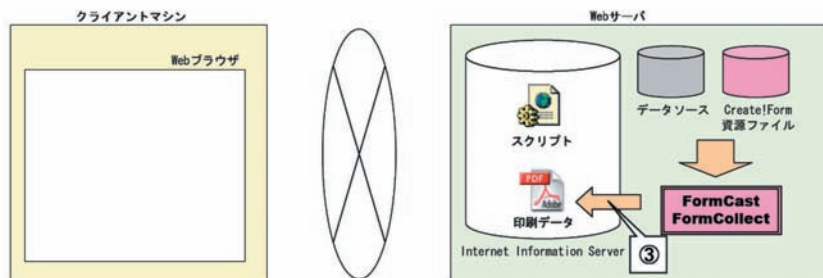
①クライアントは、FormCast ランタイム（あるいは FormCollect ランタイム）を起動するスクリプトにアクセスします。



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。

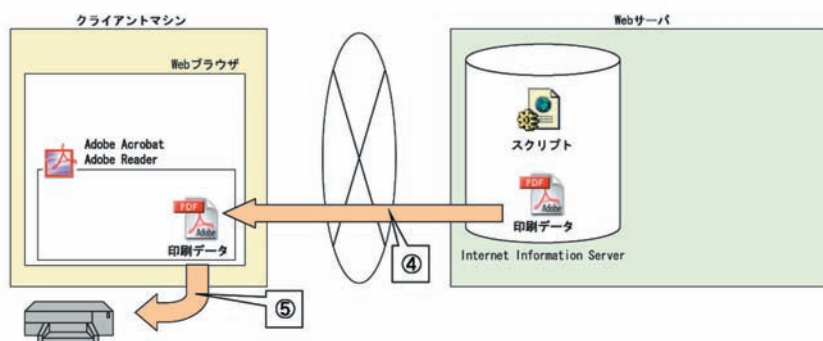


- ③ランタイムは、サーバ上のデータソースと資源ファイルから PDF ファイルを作成します。



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

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



またブラウザに関しては、Adobe Acrobat、あるいは Adobe Reader のアドインをサポートする Web ブラウザ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

ブラウザ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へ印刷する方法です。

この非表示での印刷方法は、サーバ側で作成された PDF ファイルを表示せずに Windows 環境で「通常使用するプリンタ」に設定されたプリンタ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「通常使用するプリンタ」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（ActiveX）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、サーバ側では PDF ファイルを生成する FormCast ランタイム（または FormCollect ランタイム）が必要です。

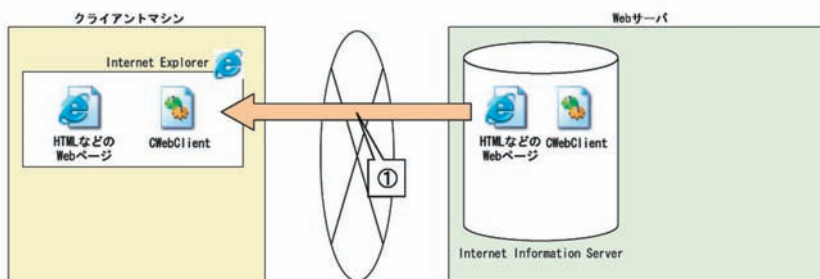
<< 注意 >>

Adobe Acrobat、あるいは Adobe Reader のバージョン 6 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 >>

印刷コントロール（ActiveX コントロール）を使用するため、使用するブラウザは IE（バージョン 5 以上）が必須となります。

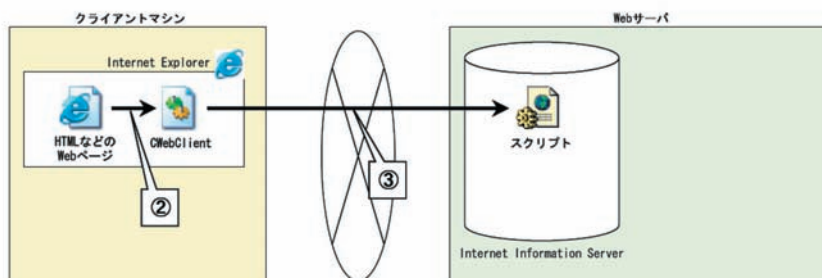
① object タグに印刷コントロールが埋め込まれた Web ページをクライアントが IE で参照すると、印刷コントロールはクライアントマシンにダウンロード、インストールされます。



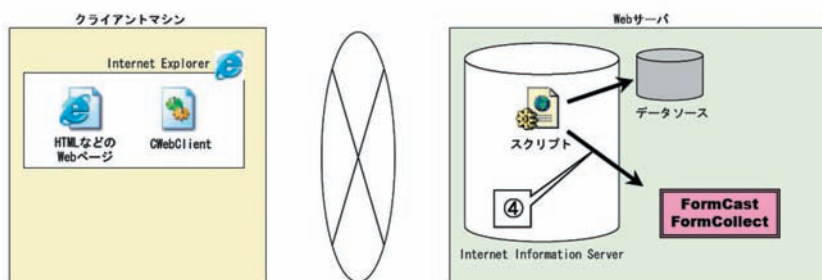
②その Web ページから印刷要求を発生させます。

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。

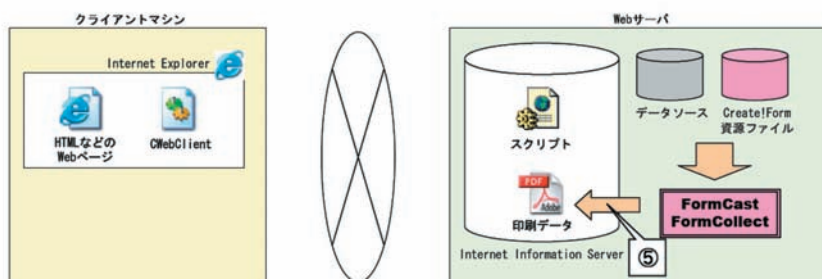
③印刷コントロールは設定されたプロパティ値を元に、Web サーバ上のスクリプトにアクセスします。



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。

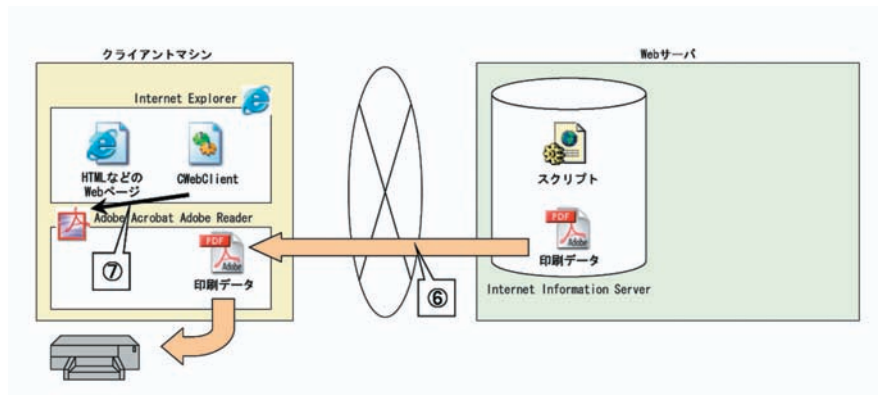


⑤ランタイムは、サーバ上のデータソースと資源ファイルから PDF ファイルを作成します。



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、あるいは Adobe 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. ブラウザ任意指定印刷とは

ブラウザでの印刷指示により任意のプリンタ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へ印刷する場合に利用します。

プリンタが複数の場合にはプリンタの設定、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 (ActiveX) を使用する必要があります。

<< 注意 >>

ブラウザ任意指定印刷を行うにあたり、サーバ側では FormPrintStage Web ランタイム、クライアント側では FormPrintStage WebClient ランタイムが必要です。

<< 注意 >>

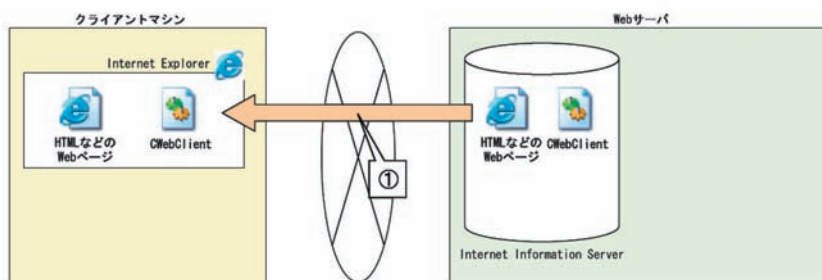
印刷コントロール (ActiveX コントロール) を使用するため、使用するブラウザは IE (バージョン 5 以上) が必須となります。

ブラウザ任意指定印刷では、サーバで生成された 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、FormPrintStage WebClient ランタイムが必要となります。

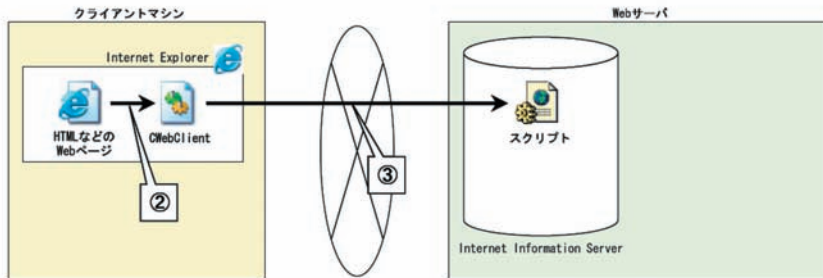
① object タグに印刷コントロールが埋め込まれた Web ページをクライアントが IE で参照すると、印刷コントロールはクライアントマシンにダウンロード、インストールされます。



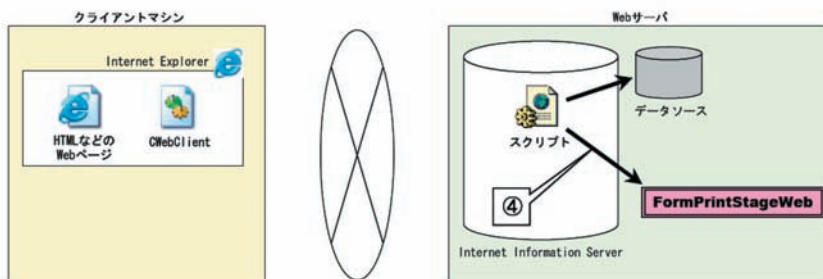
②その Web ページから印刷要求を発生させます。

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。

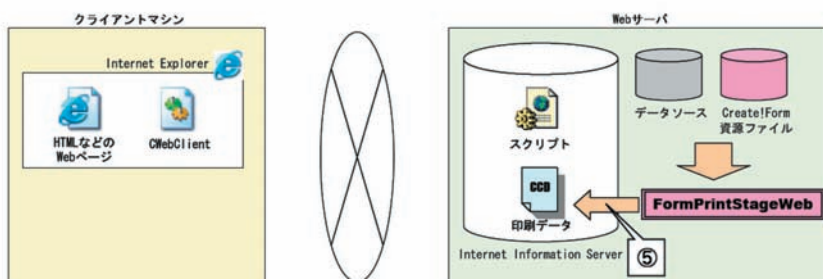
③印刷コントロールは設定されたプロパティ値を元に、Web サーバ上のスクリプトにアクセスします。



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。

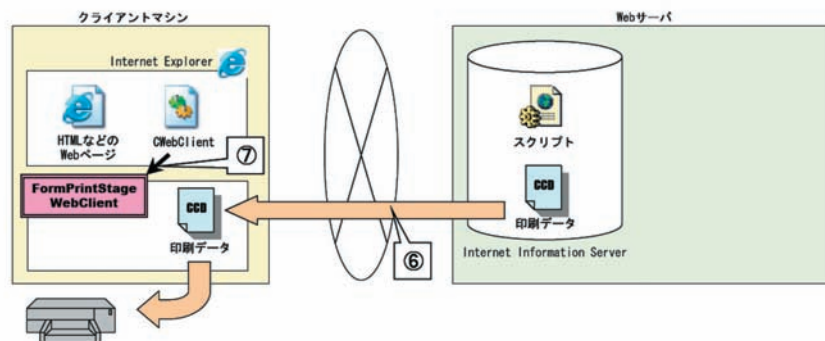


⑤ ランタイムは、サーバ上のデータソースと資源ファイルから CCD ファイルを作成します。



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、FormPrintStage WebClient 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へ印刷を実行します。



以上がブラウザ任意指定印刷の基本的な流れになります。

FormPrintStage WebClient 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“B”に、帳票“C”はプリンタ“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
- ・ 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 サーバの基本設定

サーバマシンの構成

Web サーバの設定を行うにあたり、Windows 2003 Server + IIS 6.0 + ASP 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。

1. 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。



※ 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。

※ 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 FormCast ランタイム（または FormCollect ランタイム）、「ブラウザ任意指定印刷」であれば FormPrintStage Web ランタイムをスクリプトで起動します。

※ 3: CWebClient.ocx は印刷コントロールのファイル名です。Create!Form 導入ディレクトリ直下の「lib」ディレクトリに配置されていますので、「C:¥createform¥lib」ディレクトリにコピーしてください。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.ocx を配置する必要はありません。

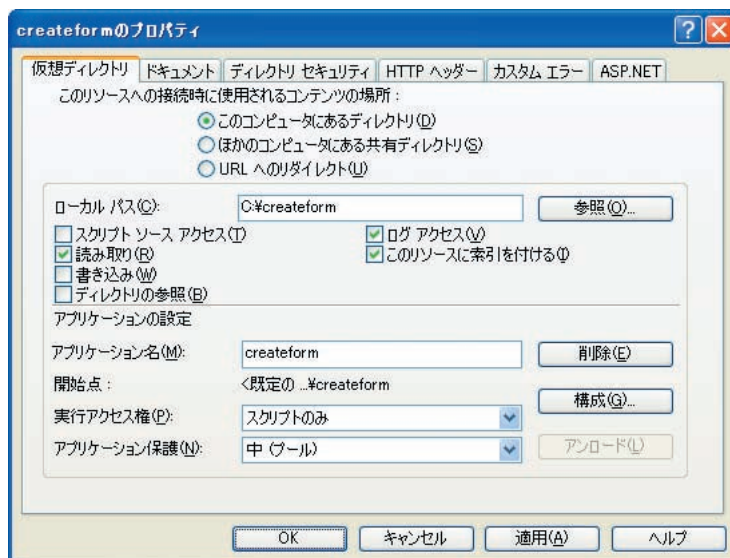
※ 4: 印刷データ生成ディレクトリには、各ランタイムが生成するファイル（PDF ファイル、EMF ファイル、GCD ファイル）が格納されます。

2. 仮想ディレクトリの作成

「インターネット サービス マネージャ」を使用して仮想ディレクトリを作成します。

testsv サーバ上の C:¥createform ディレクトリをエイリアスとして「createform」で設定します。

図：IIS マネージャ



6. クライアントの基本設定

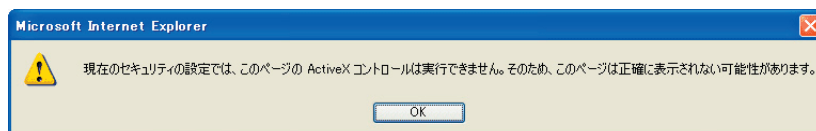
6-1. IE のセキュリティ設定

※ここで説明する内容は、「PDF 非表示印刷」、「ブラウザ任意指定印刷」での Web クライアント印刷を実装する場合のみ必要です。「PDF 表示印刷」での実装の場合はクライアントの設定を行う必要はありません。

印刷コントロールを使用した Web クライアント印刷において、クライアントマシンは初めに IE で印刷コントロールを起動する Web ページにアクセスします。

この際、クライアントマシンの IE の設定によっては次のようなメッセージが表示されます。

図：警告ダイアログ

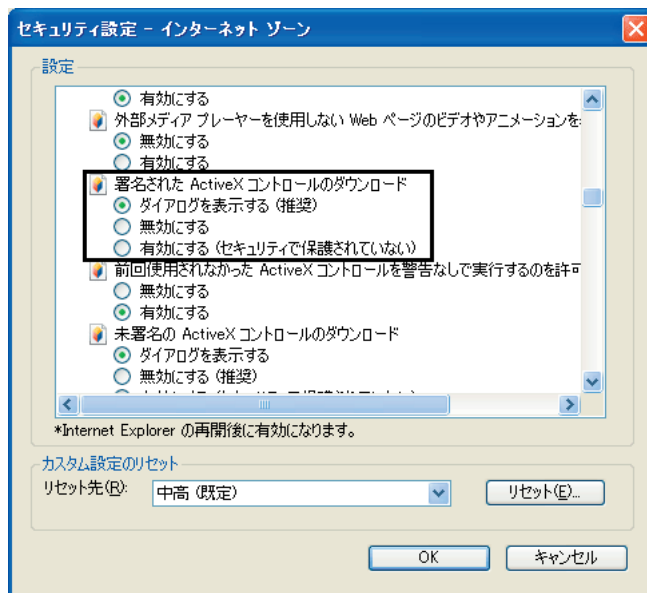


この場合、印刷コントロールはインストールされますが、IE の設定を変更する必要があります。IE（以下の説明では IE8 を使用しています）の [ツール] メニューから [インターネット オプション] を選択します。

[セキュリティ] タブを開き、適切なゾーンを選択して、[レベルのカスタマイズ] をクリックします。このとき、ゾーンが正しく選択されていることを確認してください。

[ActiveX コントロールとプラグイン] の [署名された ActiveX コントロールのダウンロード] の設定を [ダイアログを表示する] か [有効にする] に設定します。

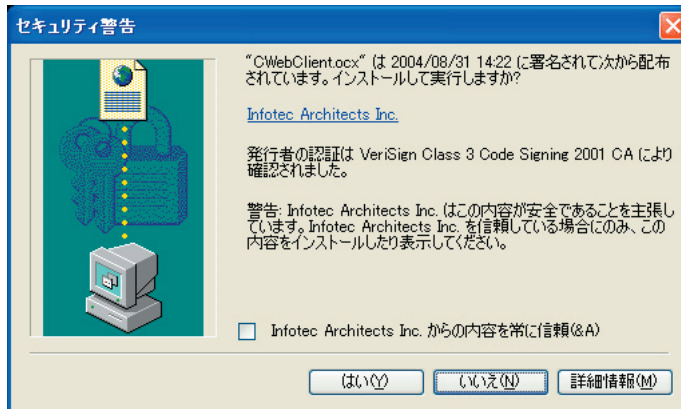
図：セキュリティの設定



[有効にする] を選択した場合は、自動的にインストールされます。

[ダイアログを表示する] を選択した場合、インストールする前に次のようなメッセージが表示されます。

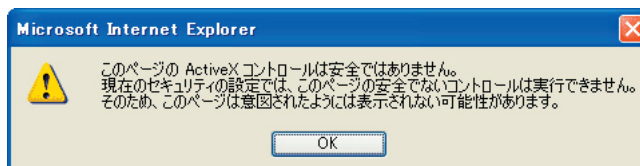
図：セキュリティ警告ダイアログ



ここで、[はい]を選択すると印刷コントロールがインストールされます。

また、印刷コントロールのプロパティやメソッドにアクセスした際に、次のような警告が出る場合もあります。

図：警告ダイアログ



この場合も、セキュリティの設定を変更する必要があります。

IE（以下の説明では IE8 を使用しています）の [ツール] メニューから [インターネット オプション] を選択します。

[セキュリティ] タブを開き、適切なゾーンを選択して、[レベルのカスタマイズ] をクリックします。このとき、ゾーンが正しく選択されていることを確認してください。

[ActiveX コントロールとプラグイン] の [署名された ActiveX コントロールのダウンロード] の設定を [ダイアログを表示する] か [有効にする] に設定します。

次節からは、各印刷方法に応じた構築例を説明します。

7. PDF 表示印刷の構築例

① "index.htm" の作成

```
<html>
<head>
<title>PDF 表示印刷</title>
<script language="JavaScript">
function ShowPdf() {
    window.open("http://testsv/createform/run.asp?OutputFileName=test.pdf",
        "_blank");
}
</script>
</head>
<body>
    <input type="button" value="PDF 表示印刷" onClick="ShowPdf()" />
</body>
</html>
```

"index.htm" は、クライアントがブラウザに読み込んで、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、FormCast ランタイムを実行するためのスクリプト "run.asp" に画面遷移するためのものです。これにより、ブラウザ上から "PDF 表示印刷" ボタンをクリックすることで、サーバに対して PDF 生成要求を発行することが可能です。

"index.htm" は c:\%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、サーバの FormCast ランタイムが生成する PDF ファイルの名前として、run.asp 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 "run.asp" の作成

```
<%@ Language=VBScript%>
<%
Dim objCast
Dim strCurPath      ' 実行ディレクトリ
Dim strTextFile     ' テキストファイル名
Dim strWorkPath     ' Create!Form 作業ディレクトリ
Dim strPDFPath      ' PDF ファイルパス
Dim strCmd          ' FormCast ランタイム実行コマンドパラメータ
Dim ErrNum          ' 戻り値
strCurPath = "c:\%resource"
```

```

' =====
' FormCast 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"
' =====
' FormCast ランタイムの実行
' =====

Set objCast = CreateObject("CCastCOM.CCast")
strWorkPath = strCurPath & "%cwork"
strPDFPath = "c:%createform%printings%" & Request.QueryString("OutputFileName")

' コマンドの作成
strCmd = "-D" & strWorkPath & " -sSchoolLife.sty -o" & strPDFPath & " " & _
    strTextFile
' FormCast ランタイム実行（第二パラメータの 8705 は、
' CPW_HIDE | ERW_HIDE | ERL_ENABLE の指定と同じです。）
ErrNum = objCast.CCastExec( 0, 8705, "", strCmd )
' オブジェクトの解放
Set objCast = Nothing
' 生成された PDF ファイルをリダイレクトで返す
Response.Redirect("printings/" & Request.QueryString("OutputFileName"))
%>

```

“run.asp” は、FormCast ランタイムを呼び出すスクリプトです。

FormCast ランタイムは COM コンポーネントを利用して呼び出しています。

“index.htm” 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。FormCast ランタイム実行後、生成された PDF ファイルをリダイレクトでクライアントに返しています。これにより、クライアントは PDF ファイルをブラウザで表示できることになります。

“run.asp” は C:%createform 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。

ブラウザから “http://testsv/createform/index.htm” にアクセスします。ブラウザに表示された “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、“run.asp” にアクセスします。サーバ側で FormCast ランタイムの実行が終了すると、作成された PDF ファイルがリダイレクトされるので、クライアント側に PDF ファイルが表示されます。

※ Adobe Acrobat、あるいは Adobe Reader がインストールされており、アドイン機能が有効なブラウザを使用している場合のみです。

Adobe Acrobat、あるいは Adobe 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.htm" の作成

```
<html>
<head>
<title>PDF 非表示印刷</title>
<object id="obj" classid="clsid:F58DEEA6-72E2-4811-8BE4-47796A7E804B"
  codebase="http://testsv/createform/lib/CWebClient.ocx#version=10,0,0,0">
</object>
<script language="JavaScript">
function KickPrintExecutePdf() {
  obj.CreatePrintingDataScriptUrl = "http://testsv/createform/run.asp";
  obj.PrintExecutePdf();
}
</script>
</head>
<body>
  <input type="button" value="PDF 非表示印刷" onClick="KickPrintExecutePdf()" />
</body>
</html>
```

“index.htm”は、クライアントがIE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にダウンロード・インストールするため、object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。

id:

印刷コントロールオブジェクトの変数名として使用され、任意の変数名を指定できます。

classid:

固定値で「clsid:F58DEEA6-72E2-4811-8BE4-47796A7E804B」を記述します。

codebase:

印刷コントロールが置かれているディレクトリのURLと、バージョン番号を追加した形式で記述します。現在ご使用のバージョン番号を記述してください。

objectタグを記述することにより、クライアントマシンに印刷コントロールがインストールされていない場合、またはcodebaseで記述したバージョンよりも古いコントロールがインストールされている場合、Webサーバより印刷コントロールがダウンロードされクライアントマシンにインストールされます。“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、FormCast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、“run.asp”がFormCastランタイムを実行するスクリプトなので、“http://testsv/createform/run.asp”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「CreatePrintingData ScriptUrl」で指定されたスクリプトにアクセスし、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については、後述の「印刷コントロール API 仕様」を参照してください。「index.htm」は c:\createform ディレクトリ直下に配置します。

② "run.asp" の作成

```
<%@ Language=VBScript%>
<%
Dim objCast
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 strPDFPath      ' PDF ファイルパス
Dim strCmd          ' FormCast ランタイム実行コマンドパラメータ
Dim ErrNum          ' 戻り値
strCurPath = "c:\%resource"
' =====
' FormCast 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"
' =====
' FormCast ランタイムの実行
' =====

Set objCast = CreateObject("CCastCOM.CCast")
strCWorkPath = strCurPath & "%cwork"
strPDFPath = "c:\createform%printings%" & Request.Form("OutputFileName")

' コマンドの作成
strCmd = "-D" & strCWorkPath & " -sSchoolLife.sty -o" & strPDFPath & " " & _
    strTextFile
' FormCast ランタイム実行（第二パラメータの 8705 は、
' CPW_HIDE | ERW_HIDE | ERL_ENABLE の指定と同じです。）
ErrNum = objCast.CCastExec( 0, 8705, "", strCmd )
' オブジェクトの解放
Set objCast = Nothing
' 生成された PDF ファイルをリダイレクトで返す
Response.Redirect("%printings/" & Request.Form("OutputFileName"))
%>
```

“run.asp” は、FormCast ランタイムを呼び出すスクリプトです。

FormCast ランタイムは COM コンポーネントを利用して呼び出しています。

変数 “strPDFPath” に注目してください。

POST 文字列で “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 “run. asp” 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については、後述の「印刷コントロール API 仕様」を参照してください。

FormCast ランタイム実行後、生成された PDF ファイルをリダイレクトでクライアントに返しています。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。“run. asp” は c:\%createform 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。

IE から “http://testsv/createform/index. htm” にアクセスします。IE 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、“run. asp” にアクセスします。サーバ側で FormCast ランタイムの実行が終了すると、作成された PDF ファイルがリダイレクトされ、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、あるいは Adobe 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「通常使用するプリンタ」に設定されているプリンタに対して実行されます。

印刷情報も「通常使用するプリンタ」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、あるいは Adobe Reader がインストールされている必要があります。

<< 注意 >>

“index. htm” を IE でアクセスした際 (印刷コントロールのインストール時) や、印刷コントロールのプロパティやメソッドにアクセスした際に、警告ダイアログが表示されることがあります。この警告ダイアログは、IE のセキュリティ設定により表示されます。詳しくは、前述の「クライアントの基本設定」を参照してください。

<< 注意 >>

印刷設定はすでに設定されている Adobe Acrobat 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で選択されている用紙サイズが異なる場合、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 4 の場合>

Acrobat の [印刷] ダイアログボックスにある [用紙サイズに合わせる] チェックボックスをチェックします。

<Acrobat 5 の場合>

Acrobat の [印刷] ダイアログボックスにある [用紙サイズに合わせてページを拡大] と [用紙サイズに合わせてページを縮小] チェックボックスをチェックします。

<Acrobat 6、7、8、9 の場合>

Acrobat の [印刷] ダイアログボックスにある [ページ処理] の [ページの拡大 / 縮小] リストから [用紙に合わせる] を選択します。

<Acrobat 10、11 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 [サイズ] ボタンからサイズオプション「合わせる」を選択します。

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仕様については、後述の「印刷コントロール API 仕様」を参照してください。

9. ブラウザ任意指定印刷の構築例

① "index.htm" の作成

```
<html>
<head>
<title> ブラウザ任意指定印刷 </title>
<object id="obj" classid="clsid:F58DEEA6-72E2-4811-8BE4-47796A7E804B"
  codebase="http://testsv/createform/lib/CWebClient.ocx#version=10,0,0,0">
</object>
<script language="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>
</head>
<body>
  <input type="button" value=" ブラウザ任意指定印刷 " onClick="KickPrintExecuteEx()" />
</body>
</html>
```

"index.htm" は、クライアントが IE 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にダウンロード・インストールするため、object 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。

id:

印刷コントロールオブジェクトの変数名として使用され、任意の変数名を指定できます。

classid:

固定値で「clsid:F58DEEA6-72E2-4811-8BE4-47796A7E804B」を記述します。

codebase:

印刷コントロールが置かれているディレクトリの URL と、バージョン番号を追加した形式で記述します。現在ご使用のバージョン番号を記述してください。

object タグを記述することにより、クライアントマシンに印刷コントロールがインストールされていない場合、または codebase で記述したバージョンよりも古いコントロールがインストールされている場合、Web サーバより印刷コントロールがダウンロードされクライアントマシンにインストールされます。" ブラウザ任意指定印刷 " 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、FormPrintStage Web 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、"run.asp" が FormPrintStage Web ランタイムを実行するスクリプトなので、"http://testsv/createform/run.asp" 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「CreatePrintingData ScriptUrl」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については、後述の「印刷コントロール API 仕様」を参照してください。
 “index.htm”は c:\createform ディレクトリ直下に配置します。

② "run.asp" の作成

```

<%@ 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          ' FormPrintStageWeb ランタイム実行コマンドパラメータ
Dim ErrNum          ' 戻り値
strCurPath = "c:\resource"
' =====
' FormPrintStage Web 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"
' =====
' FormPrintStage Web 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
' FormPrintStageWeb ランタイム実行（第二パラメータの 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
' 生成された CCD ファイルをリダイレクトで返す
Response.Redirect("printings/" & Request.Form("OutputFileName"))
%>

```

“run.asp”は、FormPrintStage Web ランタイムを呼び出すスクリプトです。

FormPrintStage Web ランタイムは COM コンポーネントを利用して呼び出しています。変数 “strCCDPath” に注目してください。POST 文字列で “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 “run.asp” 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の EMF 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については、後述の「印刷コントロール API 仕様」を参照してください。

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.htm” 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、FormPrintStage Web 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” オプションの詳細については、後述の「CCD ファイル保存機能」を参照してください。

FormPrintStage Web ランタイム実行後、生成された CCD ファイルをリダイレクトでクライアントに返しています。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。“run.asp” は c:\createform 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。

IE から “http://testsv/createform/index.htm” にアクセスします。IE に表示された “ブラウザ任意指定印刷” ボタンをクリックします。印刷コントロールが起動し、“run.asp” にアクセスします。サーバ側で FormPrintStage Web ランタイムの実行が終了すると、作成された CCD ファイルがリダイレクトされ、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、FormPrintStage WebClient ランタイム

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。

印刷情報の設定に関しては、後述の「印刷設定ダイアログの起動方法」を参照してください。

<< 注意 >>

“index.htm” を IE でアクセスした際（印刷コントロールのインストール時）や、印刷コントロールのプロパティやメソッドにアクセスした際に、警告ダイアログが表示されることがあります。この警告ダイアログは、IE のセキュリティ設定により表示されます。詳しくは、前述の「クライアントの基本設定」を参照してください。

以上で、ブラウザ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仕様については、後述の「印刷コントロール API 仕様」を参照してください。

「ブラウザ任意指定印刷」は印刷コントロールと FormPrintStage WebClient 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。

詳しくは後述の「WebClient ランタイムについて」を参照してください。

10. 印刷コントロール API 仕様

10-1. 起動方法

印刷コントロールの実行には次のファイルを使用します。

CWebClient.ocx

このファイルは Create!Form 導入ディレクトリ直下の「lib」ディレクトリ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. クラス情報

プロパティ :BSTR 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。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。詳細はメソッド「PrintExecuteEx」の項目、及び後述の「WebClient ランタイムについて」を参照してください。

プロパティ :BSTR 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。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 :BSTR 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。また、このプロパティ値は必須です。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 :BSTR 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。

```
obj. FormRequestArray = "name=create&product=form"
```

プロパティ :Integer MinimizedProgressDialog

概要：

メソッド「PrintExecuteEx」「PrintExecutePdf」「PreviewExecutePdf」実行時に表示される処理進行ダイアログ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、処理進行ダイアログは最小化で表示されます。指定なし、及び“0”を指定すると、ダイアログは通常サイズの大きさで表示されます。

メソッド :PrintExecuteEx BSTR StyleName

引数：

StyleName

文字列型。

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。

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、サーバのランタイムは FormPrintStage Web 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 FormPrintStage WebClient ランタイムが導入されている必要があります。

※詳細は、後述の「WebClient ランタイムについて」を参照してください。

メソッド :PrintExecutePdf

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、サーバのランタイムは FormCast ランタイム（あるいは FormCollect 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（あるいは Adobe Reader）が必要です。

メソッド:PreviewExecutePdf

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、サーバのランタイムはFormCast ランタイム（あるいはFormCollect 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（あるいは Adobe Reader）が必要です。

メソッド:Init

概要：

印刷コントロールクラスの各プロパティを全て初期化します。

メソッド:ChangePrintInfo BSTR StyleName

引数：

StyleName

文字列型。

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。このメソッドを実行すると、印刷情報設定変更ダイアログが起動します。このメソッドを実行するには、クライアントに FormPrintStage WebClient ランタイムが導入されている必要があります。

※印刷情報設定変更ダイアログの詳細は、後述の「印刷設定ダイアログの起動方法」を参照してください。

メソッド:ResetClosedFinishing

概要：

メソッド「PrintExecuteEx」「PrintExecutePdf」「PreviewExecutePdf」実行時に表示される処理進行ダイアログの「処理終了後、このダイアログボックスを閉じる」で、閉じる設定（オンにしている状態）を指定している場合、これを強制的に解除するメソッドになります。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処理進行ダイアログの「処理終了後、このダイアログボックスを閉じる」で、閉じない設定（オフにしている状態）を指定している場合、これを強制的に閉じる設定に変更するメソッドになります。処理進行ダイアログを自動的に閉じる場合に使用します。

プロパティ:BSTR 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。

メソッド :long retval SyncPrintExecuteEx BSTR StyleName

戻り値：

retval

数値型。

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

文字列型。

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。

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。処理中はブラウザを操作することができません。

メソッド :long retval SyncPrintExecutePdf

戻り値：

retval

数値型。

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。処理中はブラウザを操作することができません。

メソッド :long retval SyncPreviewExecutePdf

戻り値：

retval

数値型。

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。処理中はブラウザを操作することができません。

メソッド :Integer 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。この項目についての詳細は、後述のセキュリティ印刷をご覧ください。

11. 印刷コントロール 結果コード一覧

印刷コントロールのメソッド「PrintExecuteEx」「PrintExecutePdf」「PreviewExecutePdf」（及び「SyncPrintExecuteEx」「SyncPrintExecutePdf」「SyncPreviewExecutePdf」）の処理結果は、各メソッドとも共通です。

以下にその一覧と対処法、あるいは詳細説明を記述します。

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 が正しいか、サーバ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で指定したディレクトリの権限を確認してください。

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 FormPrintStagWeb 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。詳細情報はマニュアルで確認してください。

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：Adobe Acrobat、Adobe Reader のプロセス終了に失敗しました。

対処：Adobe Acrobat、あるいは Adobe Reader のプロセスが残っているため、タスクマネージャ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：「通常使用するプリンタ」を取得することができません。「通常使用するプリンタ」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。CGD ファイル保存フォルダ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 V10 の導入ディレクトリを取得できませんでした。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。

2106

説明：DLL ファイルを参照できませんでした。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。

2107

説明：関数「〈関数名〉」を参照できませんでした。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。

2108

説明：プリンタハンドルの取得に失敗しました。プリンタの設定等を確認してください。

2109

説明：プリンタ情報の取得に失敗しました。プリンタの設定等を確認してください。

2111

説明：クライアント印刷の初期化に失敗しました。

対処：クライアントランタイムが正しくインストールされていない可能性があります。
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。

2112

説明：クライアント印刷の実行に失敗しました。

対処：クライアントランタイムが正しくインストールされていない可能性があります。
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。

2113

説明：強制終了しました。処理結果を確認して下さい。

対処：処理進行ダイアログの「×」ボタンを押して終了した場合にこのエラーメッセージが表示されます。

2114

説明：一時ファイルの作成に失敗しました。

対処：テンポラリフォルダが正しく設定されていることを確認してください。

2115

説明：一時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への書き込みに関するアクセス権を確認してください。

2116

説明：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へのアクセス権を確認してください。

2117

説明：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へのアクセス権を確認してください。

2200

説明：URL でない文字列が指定されています。CreatePrintingDataScriptUrl プロパティの設定値を確認してください。

対処：CreatePrintingDataScriptUrl に、URL でない値が代入されていないか確認してください。

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。

2202

説明：サーバへの接続に失敗しました。サーバの環境を確認してください。

2203

説明：サーバスクリプトへの要求に失敗しました。

対処：サーバ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。

2206

説明：インターネットに接続できません。環境を確認してください。

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。

2207

説明：アクセス設定情報の取得に失敗しました。

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。

2208

説明：サーバへのアクセス設定に失敗しました。

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。

2209

説明：HTTP レスポンスの取得に失敗しました。サーバの設定等を確認してください。

2210

説明：サーバスクリプトへの接続に失敗しました。

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。

2211

説明：転送量の取得に失敗しました。サーバの設定等を確認してください。

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。

2301

説明：デバイスコンテキスト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。

2302

説明：印刷処理が開始できませんでした。プリンタの設定等を確認してください。

2303

説明：印刷処理が実行できません。サーバで使用しているランタイムを確認してください。

2304

説明：印刷処理を終了できませんでした。プリンタの設定等を確認してください。

2307

説明：印刷処理の続行に失敗しました。プリンタの設定等を確認してください。

2308

説明：印刷処理の続行に失敗しました。プリンタの設定等を確認してください。

2310

説明：用紙の向きを設定することができませんでした。プリンタの設定等を確認してください。

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。

2403

説明：ダウンロードした CCD ファイルを正常に閉じることができませんでした。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。

2404

説明：ダウンロードした CCD ファイルのヘッダを読み込むことができません。サーバ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。サーバで実行されたランタイム製品が「FormPrintStageWeb」か確認してください。

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 ファイルのヘッダが正しくありません。サーバ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「FormPrintStageWeb」か確認してください。

2410

説明 : 実行コマンドオプションの生成に失敗しました。

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。

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。詳細情報はマニュアルで確認してください。

対処 : FormPrintStageWeb Client が印刷処理中にエラーを返しました。FormPrintStageWeb 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。

2412

説明 : クライアントランタイムから不明な処理結果が返されました。

クライアントランタイムが印刷処理中にエラーを返す、あるいは異常終了した可能性があります。

2413

説明 : FormPrintStageWeb 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 FormPrintStageWeb 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。

2500

説明 : Adobe Acrobat、Adobe Reader の情報を取得できませんでした。

対処 : Adobe Acrobat、あるいは Adobe Reader を再インストールしてください。

2501

説明 : Adobe Acrobat、Adobe Reader の情報を取得できませんでした。

対処 : Adobe Acrobat、あるいは Adobe Reader を再インストールしてください。

2502

説明 : Adobe Acrobat、Adobe Reader の起動に失敗しました。

対処 : Adobe Acrobat、あるいは Adobe 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。

2503

説明 : 印刷ジョブの監視に失敗しました。プリンタの設定等を確認してください。

2504

説明 : 印刷ジョブ情報の取得に失敗しました。プリンタの設定等を確認してください。

2505

説明 : 印刷ジョブへの送信に失敗しました。プリンタの設定等を確認してください。

2506

説明 : 現在導入されている Adobe Acrobat、Adobe Reader のバージョンでは本製品は使用
できません。

対処 : 対応するバージョンの Adobe Acrobat、Adobe Reader を導入してください。

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、印刷コントロールが正常に終了できない可能性があります。

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"

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（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、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（Perl 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（Perl 等）を使用している場合

バイナリを扱える Perl 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。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。Perl 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。印刷データ削除スクリプトの例は以下の通りです。

```
#!/C:/usr/Perl/bin/perl.exe
:
:
# ランタイム実行など
:
:
# 印刷データファイルオープン
open(FILE, "$OutputFilePath");
binmode FILE;
```

```
print "Content-type: application/x-compress¥n¥n";
print while (<FILE>);
close(FILE);
# 印刷データを削除
unlink("$OutputFilePath");
```

また、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。

14. スクリプトヘータを送信する

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「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「FormRequestArray」に送信する文字列を指定します。

```
obj.FormRequestArray = "name= 帳票 太郎 &id=A10256";
```

例えば、上記のようにプロパティ「FormRequestArray」の値を設定すると（例は JavaScript です）、プロパティ「CreatePrintingDataScriptUrl」で指定されたスクリプト（例では ASP です）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。

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

プロパティ「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。具体的なフォーマットは前述の「印刷コントロール API 仕様」を参照してください。

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」を使用している場合は“OutputFileName”、メソッド「PrintExecuteEx」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ない可能性がありますので注意してください。

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。

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

例えば、上記のようにプロパティ「CreatePrintingDataScriptUrl」を設定すると（例は JavaScript です），“run.asp”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。

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

<< 注意 >>

メソッド「PrintExecutePdf」「PreviewExecutePdf」を使用している場合は「OutputFileName」、メソッド「PrintExecuteEx」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ない可能性がありますので、注意してください。

14-3. セッション変数

Web システムでは、ユーザ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。印刷コントロールでも、セッション変数を使用することが可能です。

セッション変数に関しては、印刷コントロールで特別な設定を行う必要はありませんが、セッション変数を使用できるのは、以下の戻り値を返すメソッドを使用した場合です。

- ・ SyncPrintExecuteEx
- ・ SyncPrintExecutePdf
- ・ SyncPreviewExecutePdf

以下のメソッドを使用した場合には、セッション変数を使用することができませんのでご注意ください。

- ・ PrintExecuteEx
- ・ PrintExecutePdf
- ・ PreviewExecutePdf

プロパティ「CreatePrintingDataScriptUrl」「DeletePrintingDataScriptUrl」で指定したスクリプトでは、通常通りにセッション変数を参照することが可能です。

15. WebClient ランタイムについて

印刷コントロールと FormPrintStage WebClient 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、プリンタの選択やトレイの選択等、詳細な印刷情報の設定はできませんでした。しかし、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。FormPrintStage WebClient 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
- ・ 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、「印刷情報設定変更ダイアログ」で行います。起動方法や詳細については、後述の「印刷設定ダイアログの起動方法」を参照してください。

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。FormPrintStage WebClient ランタイムは FormPrintStage Web 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. 印刷設定ダイアログの起動方法

印刷情報の設定は、「印刷情報設定変更ダイアログ」から行います。印刷情報設定変更ダイアログの起動方法については以下の通りです。

- ・印刷コントロールのメソッド「ChangePrintInfo」からの起動
- ・導入ディレクトリ直下の「PrtSTConf.exe」からの起動

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」からの起動

印刷コントロールを起動する 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。

```
-----  
<html>  
<head>  
<object id="obj" classid="CLSID:F58DEEA6-72E2-4811-8BE4-47796A7E804B"  
    codebase="http://testsv/createform/lib/CWebClient.ocx#version=10,0,0,0">  
</object>  
<script language="JavaScript">  
function KickChangePrintInfo() {  
    // 引数に帳票名 (Job ファイル名) を指定します  
    obj.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="印刷情報設定ダイアログ"  
        onclick="KickChangePrintInfo"/>  
</body>  
</html>  
-----
```

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導入ディレクトリ直下の「conf」ディレクトリに作成されます。

16-2. 導入ディレクトリ直下の「PrtSTConf.exe」からの起動

導入ディレクトリ直下の「PrtSTConf.exe」を起動します。

この方法で印刷設定ダイアログを起動することで、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起動方法を使用します。

17. 印刷設定ダイアログの画面説明

印刷設定変更ダイアログは前述の「印刷設定ダイアログの起動方法」によって動作が異なります。それぞれの起動方法ごとに画面説明を行います。

17-1. 印刷コントロールのメソッド「ChangePrintInfo」からの起動

【メイン】

図：印刷情報設定変更ダイアログ - 【メイン】



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 [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保存します。

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。

[プリント設定] タブ

図：印刷情報設定変更ダイアログ - [プリント設定] タブ

① プリンタ名

現在設定を行っている帳票を出力するプリンタを設定します。
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から選択することが可能です。

② 給紙トレイ

給紙トレイを設定します。
[プリンタ名] で指定したプリンタの給紙トレイから設定可能です。

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 - [詳細設定] タブ



① 拡大 / 縮小

印刷時のスケールを元の出カイメージとの比率で設定します。

② 解像度

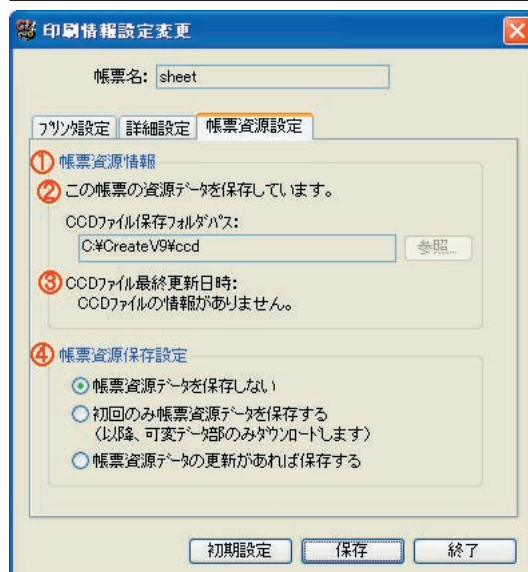
解像度を dpi 単位で設定します。

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 - [帳票資源設定] タブ



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の帳票資源データを比較して、サーバ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。

後述の「CCD ファイル保存機能」を参照してください。

17-2. 導入ディレクトリ直下の「PrtSTConf.exe」からの起動

[メイン]

図：印刷情報設定変更ダイアログ - [メイン]



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。

[プリント設定] タブ

図：印刷情報設定変更ダイアログ - [プリント設定] タブ

**① プリンタ名**

現在設定を行っている帳票を出力するプリンタを設定します。
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から選択することが可能です。

② 給紙トレイ

給紙トレイを設定します。
[プリンタ名] で指定したプリンタの給紙トレイから設定可能です。

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 - [詳細設定] タブ



① 拡大 / 縮小

印刷時のスケールを元の出カイメージとの比率で設定します。

② 解像度

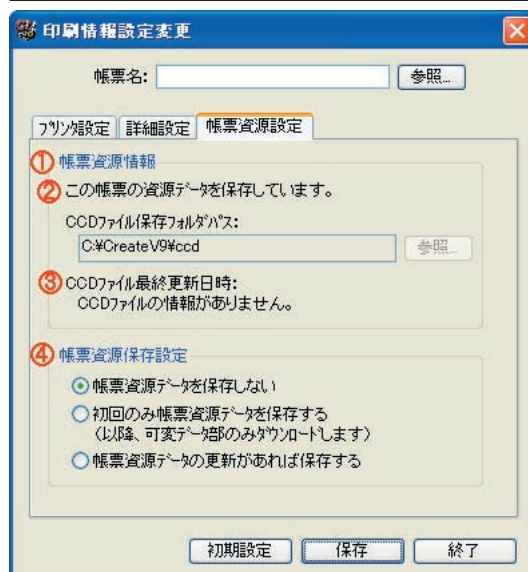
解像度を dpi 単位で設定します。

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 - [帳票資源設定] タブ



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の帳票資源データを比較して、サーバ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。

後述の「CCD ファイル保存機能」を参照してください。

18. CCD ファイル保存機能

CCD ファイル保存機能を利用する場合の例を示しながら説明します。

18-1. 印刷コントロールを起動する Web ページの構成

印刷コントロールを起動する Web ページを作成します。この Web ページには、印刷コントロールの情報と、メソッド「PrintExecuteEx」「ChangePrintInfo」を実行させるコードを記述しています。

```
<html>
<head>
<object id="obj" classid="CLSID:F58DEEA6-72E2-4811-8BE4-47796A7E804B"
    codebase="http://testsv/createform/lib/CWebClient.ocx#version=10,0,0,0">
</object>
<script language="JavaScript">
function KickPrintExecuteEx() {
    obj.CreatePrintingDataScriptUrl = "http://testsv/createform/run.asp";
    obj.DataSendMethod = "POST";
    obj.PrintExecuteEx(form1.stylename.value);
}
function KickChangePrintInfo() {
    obj.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>
    </select>
    <input type="button" name="ccdprt" value="印刷" onClick='KickPrintExecuteEx(
    )' />
    <input type="button" name="chgprt" value="変更" onClick='KickChangePrintInfo(
    )' />
</form>
</body>
</html>
```

印刷コントロールのメソッド「PrintExecuteEx」「ChangePrintInfo」の引数に注目してください。どちらも 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情報を変更するダイアログを起動するので、HTML フォームで「タイムレポート」が選択された状態で、「変更」ボタンをクリックすると、帳票名「time」の印刷情報を変更、もしくは新規作成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、「run.asp」には「POST」で「StyleName=time」が送信される、ということです。

18-2. 印刷データ生成スクリプトの構成

前述の「18-1. 印刷コントロールを起動する 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。

```
<%@ 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          ' FormPrintStageWeb ランタイム実行コマンドパラメータ
Dim ErrNum          ' 戻り値
strCurPath = "c:\$resource"
' =====
' FormPrintStage Web 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"
' =====
' FormPrintStage Web 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 & _                ' 作業フォルダの指定
"-s" & Request.Form("StyleName") & ".sty" & _    ' Job ファイルの指定
"-o" & strCCDPath & _                             ' 出力ファイルの指定
"-c" & Request.Form("CcdFileOption") & _         ' CCD ファイルの形式を指定
" " & strTextFile                                ' データソースの指定
' FormPrintStage Web ランタイム実行（第二パラメータの 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
' 生成された EMF ファイルをリダイレクトで返す
Response.Redirect("printings/" & Request.Form("OutputFileName"))
%>
```

コマンドの作成では、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（上記の例では、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情報「帳票資源保存設定」の値が反映されます。クライアントの帳票資源保存設定の値と、"CcdFileOption" で送信される値の関係は以下の通りです。

帳票資源保存設定 : CcdFileOption の値

” 帳票資源データを保存しない ” : ”r”

” 初回のみ帳票資源データを保存する ” : 初回アクセス時のみ ”r”、それ以降は ”d”

” 帳票資源データの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。

上記の例であれば、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、上記の例では対応できません。保存することのできない CCD ファイルを出力するには、”-c” オプションの値に ”rb” を指定します。

’ コマンドの作成

strCmd = ”-D” & strWorkPath & _ ’ 作業フォルダの指定

” -s” & Request.Form(”StyleName”) & “.sty” & _ ’ Job ファイルの指定

” -o” & strCCDFile & _ ’ 出力ファイルの指定

” -crb” & _ ’ 保存禁止の CCD ファイル出力の指定

” ” & strTextFile ’ データソースの指定

※スクリプト例内の「 _ 」はスクリプトを改行して記述するため使用しています。

19. セキュリティ印刷

19-1. 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、転送中のデータ印刷データの盗用を防ぐ機能です。Create!Form により出力された印刷データを暗号化し、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生成されます。伝送データの暗号化と復号化の処理は自動で行われますので、クライアントがパスワードを指定する必要はありません。暗号化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 GCD ファイルと PDF ファイルです。

19-2. 導入環境

19-2-1. サーバセキュリティモジュール

対応 OS (32bit)

Windows Server 2003 R2

対応 OS (64bit)

Windows Server 2003 R2、Windows Server 2008 R2、Windows Server 2012

※いずれも、Create!Form V10 ランタイム製品が導入されていること。

19-2-2. クライアントセキュリティモジュール

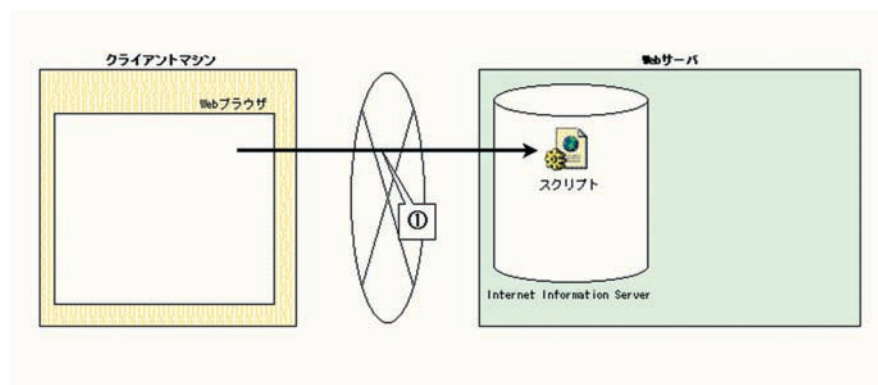
対応 OS (32bit)

Windows XP、Windows Vista、Windows 7、Windows 8

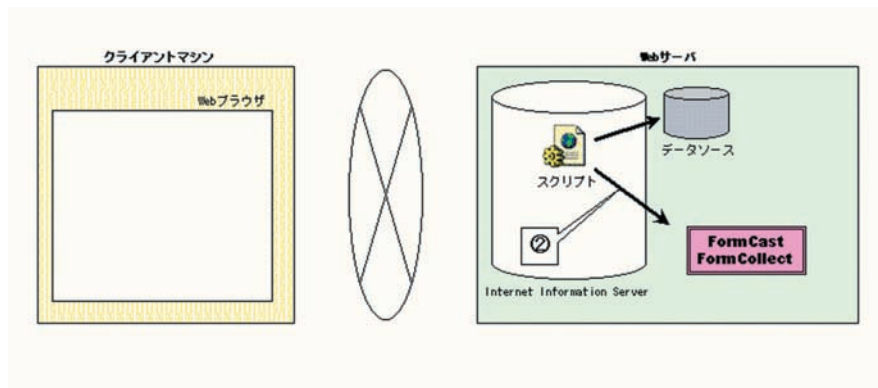
※いずれも、Internet Explorer 6.0 以上および、ActiveX モジュールのインストールと実行権限があること。

19-3. 処理の流れ

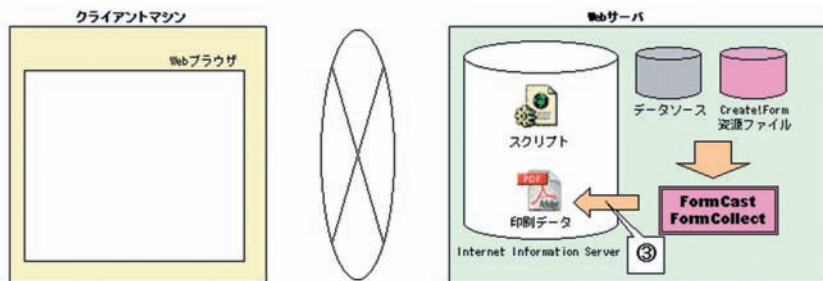
①クライアントは、FormPrintStage ランタイム（あるいは FormCast、FormCollect ランタイム）を起動するスクリプトへアクセスします。このとき、クライアントの印刷コントロールは、暗号化に必要な暗号化キーを生成し、サーバへ転送します。



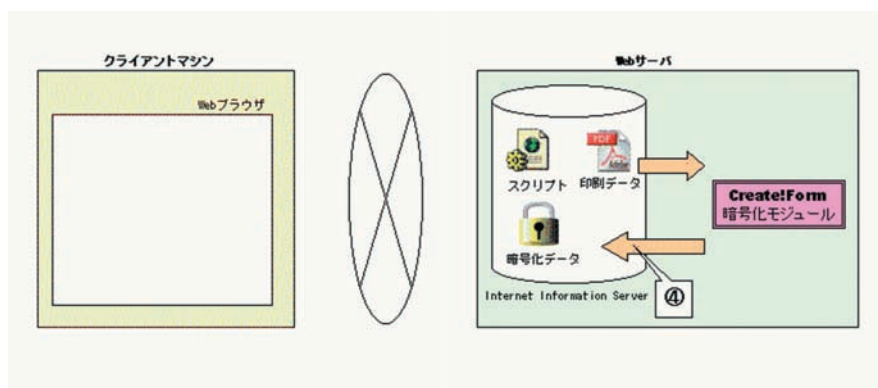
- ②リクエストを受けたスクリプトは、ランタイムを実行します。



- ③ランタイムは、サーバ上の資源ファイルから印刷データを作成します。



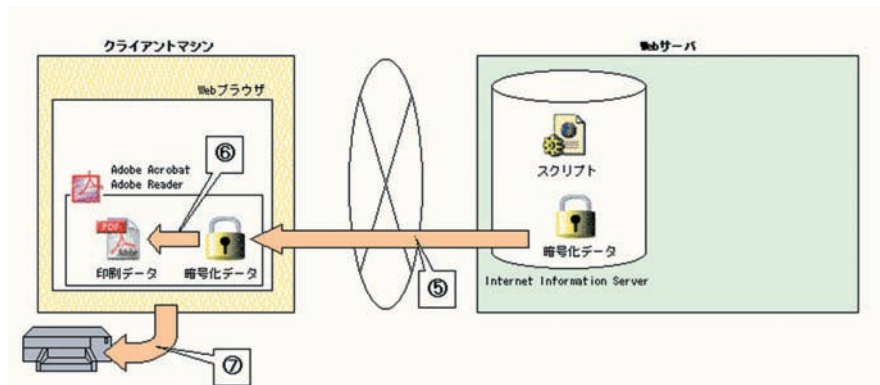
- ④スクリプトは、暗号化ランタイムを実行し、印刷データを暗号化します。



- ⑤生成された暗号化ファイルは、クライアントの印刷コントロールへ転送されます。

- ⑥印刷コントロールは、保持していた復号化キーで、転送された暗号化ファイルを復号化します。

⑦印刷コントロールは、データの印刷や表示を行います。



19-4. 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、印刷コントロールが正常に終了できない可能性があります。

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"

19-5. 使用方法

1. セキュリティモジュール

サーバセキュリティモジュール	
ログファイル設定モジュール	WCSecurityLog.exe
インタフェースモジュール	
実行ファイル	CEnc.exe
コンポーネント	CEncCOM.dll
Java ネイティブ呼び出し	cenc.jar CEncJNI.dll
コアモジュール	
CEncEdit.dll	

クライアントセキュリティモジュール

ActiveX

CWebClient.ocx

2. 呼び出しサンプル

暗号化モジュールは、Create!Form ランタイムが出力した印刷データを暗号化します。

暗号化モジュールは Create!Form ランタイムを呼び出した後に呼び出します。また、呼び出しインタフェースモジュールは、導入環境に合わせて選択し、ご利用ください。

ここでは、Web クライアント印刷のブラウザ任意指定印刷のサーバ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。詳しくは、前述の Web クライアント印刷での構築例をご覧ください。

クライアント側 "index.html"

```

<html>
<head>
<title> ブラウザ任意指定印刷 </title>
<object id="obj" classid="clsid:F58DEEA6-72E2-4811-8BE4-47796A7E804B"
  codebase="http://testsv/createform/lib/CWebClient.ocx#version=10,0,0,0">
</object>
<script language="JavaScript">
function KickPrintExecuteEx() {
  obj.SecurityOption = 1;
  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>
</head>
<body>
  <input type="button" value=" ブラウザ任意指定印刷 " onClick="KickPrintExecuteEx()" />
</body>
</html>

```

サーバ側 例1 "run.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          ' FormPrintStageWeb ランタイム実行コマンドパラメータ
Dim ErrNum          ' 戻り値
strCurPath = "c:\resource"
' =====

```

```

' FormPrintStage Web 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"

' =====
' FormPrintStage Web ランタイムの実行
' =====
Set objPrintST = CreateObject("CPrintSTCOM.CPrintST")
strWorkPath = strCurPath & "%cwork"
strCCDPPath = "c:%createform%printings%" & Request.Form("OutputFileName")

' コマンドの作成
strCmd = "-D" & strWorkPath & _
        "-s" & Request.Form("StyleName") & ".sty" & _
        "-o" & strCCDPPath & _
        "-c" & Request.Form("CcdFileOption") & _
        "" & strTextFile

' FormPrintStageWeb ランタイム実行（第二パラメータの 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
' =====
' 印刷データの暗号化
Set objEnc = Server.CreateObject("CEncCOM.EncCOM")
RetVal = objEnc.Enc("key=" & Request.Form("enckey") & _
        "&inputfile=" & strCCDPPath & _
        "&outputfile=" & strCCDPPath )

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"))
%>
=====

```

このサンプルでは、COM コンポーネントを利用してセキュリティモジュールを呼び出しています。COM コンポーネントの Enc メソッドに、実行引数を渡します。
 実行引数の詳細は、後述の「3. 呼び出しインタフェース」をご覧ください。
 暗号化モジュールの実行引数に受け渡す暗号化キーは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 ActiveX モジュールで自動的に生成されてサーバへ送られます。

サーバ側 例2 “run.java”

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.util.*;
import net.createform.sji.CPrintST;
import net.createform.common.DispConst;
import net.createform.CEncJNI;

public class CEncServlet extends HttpServlet{
public void service(HttpServletRequest request, HttpServletResponse
response)throws ServletException, IOException {

    String EncKey;
    String OutputFileName;
    EncKey = "";
    OutputFileName = "";
    Enumeration enum = request.getParameterNames();

    while(enum.hasMoreElements()){
        String name = (String)enum.nextElement();
        if(name.equals("enckey")){
            EncKey = request.getParameterValues(name)[0];
        }
        if(name.equals("OutputFileName")){
            OutputFileName = request.getParameterValues(name)[0];
        }
    }

    //-----
    // FormPrintStage ランタイムの実行
    //-----
    String commandline = "-DC:%%resource%%workdir -stest.sty -oC:%%web%%
input%%"+OutputFileName+" -c C:%%resource%%workdir%%sample.csv";
    CPrintST cprintst = new CPrintST();
    int ret = cprintst.execCPrintSTS(commandline);

    //-----
    // 印刷データの暗号化
    //-----
    CEncJNI cenc = new CEncJNI();
```

```

cenc. Enc ("key="+EncKey+"&inputfile=c:¥¥web¥¥input¥¥"
          +OutputFileName+"&outputfile=c:¥¥web¥¥output¥¥"+OutputFileName);
response. sendRedirect ("http://testsv/output/" +OutputFileName);
}

```

このサンプルでは、Java ネイティブ呼出しを利用してセキュリティモジュールを呼び出しています。cenc. jar の Enc メソッドに、実行引数を渡します。

実行引数の詳細は、後述の「3. 呼び出しインタフェース」をご覧ください。

暗号化モジュールの実行引数に受け渡す暗号化キーは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 ActiveX モジュールで自動的に生成されてサーバへ送られます。

3. 呼び出しインタフェース

3-1. 実行ファイル (EXE)

CEnc. exe を使用します。

(書式)

CEnc [引数]

(引数)

文字列型：クライアントから受け取った暗号化キー、暗号化したい生データパス、暗号化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 = " 暗号化キー文字列 " & inputfile = " 生データファイルパス "
      & outputfile = " 暗号化ファイルパス "
```

(例)

コマンドライン

```
CEnc key="ABCDEF6"&inputfile="C:¥Data¥data1. emf"&outputfile="C:¥Enc¥data1. emf"
```

3-2. コンポーネント (COM)

CEncCOM. dll を使用します。

(書式)

int CEnc (BSTR p_arg)

(引数)

p_arg

文字列型：クライアントから受け取った暗号化キー、暗号化したい生データパス、暗号化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 = " 暗号化キー文字列 " & inputfile = " 生データファイルパス "
      & outputfile = " 暗号化ファイルパス "
```

(戻り値)

0：正常終了 0 以外：異常終了

(例)

■ ASP

```
Dim objEnc
```

```
Set objEnc = Server. CreateObject ("CEncCOM. EncCOM")
```

```
RetVal = objEnc. Enc ("key="+Request. Form ("enckey")&
                    "&inputfile="&"C:¥Data¥data1. emf"&outputfile="&"C:¥Enc¥"&Request. Form ("OutputFil
                    eName")&"")
```

```

If RetVal <> 0 Then
    ' エラー処理
End If
Set objEnc = Nothing

```

3-3. Java ネイティブ呼び出し (JNI)

cenc.jar を使用します。導入ディレクトリ直下の [lib] ディレクトリに存在しますので、この JAR ファイルに対してクラスパスを設定して使用します。

(書式)

```
int Enc(String p_arg)
```

(引数)

p_arg

文字列型: クライアントから受け取った暗号化キー、暗号化したい生データパス、暗号化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 = "暗号化キー文字列" & inputfile = "生データファイルパス" & outputfile = "暗号化ファイルパス"

(戻り値)

0: 正常終了 0 以外: 異常終了

(例)

```

import net.createform.CEncJNI;
...

CEncJNI enc = new CEncJNI();
enc.Enc("key="+EncKey+"&inputfile="+InputFileName+"&outputfile="+OutputFileName);

```

3-4. .Net 呼び出し

CEncLib.dll を使用します。導入ディレクトリ直下の [lib] ディレクトリに存在しますので、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。

(書式)

```
int Execute(CEncParam);
```

(引数)

CEncParam

CEncParam 型: クライアントから受け取った暗号化キー、暗号化したい生データパス、暗号化ファイルパスを CEncParam 型に設定します。詳しくは以下の例をご覧ください。

(戻り値)

0: 正常終了 0 以外: 異常終了

(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);

```

19-6. ログの出力

19-6-1. ログの種類

エラーログ

エラー発生時に出力されます。システム異常時に参照します。

実行ログ

ジョブの発生と終了が出力されます。ジョブの管理時に参照します。

実行詳細ログ

ジョブの処理内容が出力されます。システム異常時のデバッグ時に参照します。

19-6-2. ログの設定

ログの設定は、Create!Form V10 の導入ディレクトリ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。

19-6-3. 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。

Create!Form

Web クライアント印刷 第 2 版

発行日	2015 年 2 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