

Generate PDF Report on the Web

Desheng(Ben) Xu, CGS, Northern Illinois University, De Kalb, IL

ABSTRACT

PDF file gets more and more popular because of its portability on different platforms and unique features on information presentation. This paper discusses how to produce PDF report from the web application through combination of Output Delivery System (ODS) and Application Dispatcher. JavaScript language is involved to approach more desirable result. This paper is appropriate for intermediate programmers who know SAS/Intrnet as well as JavaScript language.

INTRODUCTION

In order to better demonstrate the examples, a modified data set from SASHELP.CLASS is used. Only four observations are used so that the figure size can be very small. I assume the readers have good understanding on how to use Application Dispatcher. Also, you know the basics about JavaScript language. To view the PDF file, Adobe Acrobat Reader 5.0 or higher version is installed on your machine. For the free version, go to web site www.adobe.com for more information.

All the examples are created with SAS 8.2 on Windows 2000. Internet Explorer 6.0 is used.

A WARMUP EXAMPLE

Sending the procedure output to PDF destination is pretty straightforward:

```
ODS PDF file = 'outpath\filename.pdf' <options>;
SAS Procedures...
ODS PDF close;
```

Like HTML destination, you can also use BODY= to specify the output file name. Although the syntax is very simple, it provides an implication for web application. You can generate the PDF file to the web folder beforehand and then use a link that points to this PDF file on the web page. It is quite useful for static PDF report. For ODS PDF options and tricks, please refer to a really good paper by Kevin P. Delaney (Paper 146 in SUGI 28).

Figure 1 shows a web page that contains a button. If you click the button, a PDF report is created on the fly.

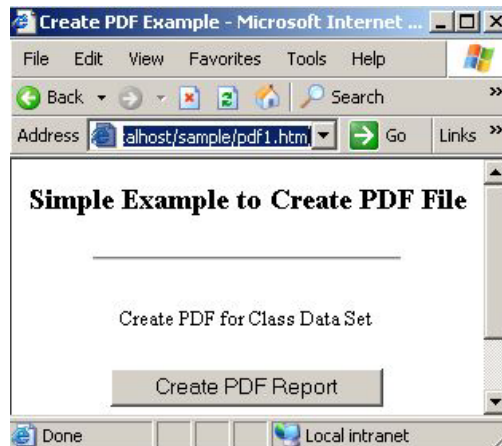


Figure 1. A simple web page to create PDF file

The HTML form behind the scene is the following:

```
<form name="PDFform" method="post" action="/cgi-bin/broker.exe">
  <center><input type="submit" name="PDFButton" value="Create PDF Report"></center>
  <input type="HIDDEN" name="_SERVICE" value="default">
  <input type="HIDDEN" name="_PROGRAM" value="pdfsrc.CreatePDF1.sas">
  <!--input type="hidden" name="_DEBUG" value=131-->
</form>
```

CreatePDF1.sas is also very simple:

```
options NODATE NONUMBER;
options TOPMARGIN="0.5 in" BOTTOMMARGIN="0.4 in"
        LEFTMARGIN="0.7 in" RIGHTMARGIN="0.7 in";
```

```
ods pdf body=_webout;
title Bold height=1 font=arial 'Simple Example for PDF';
proc print data=pdfdata.class noobs;
run;
ods pdf close;
```

TOPMARGIN, BOTTOMMARGIN, LEFTMARGIN, and RIGHTMARGIN are options used to set the margins for the generated PDF file. You can also use option ORIENTATION= to set up the output orientation. You can easily guess the choice is PORTRAIT or LANDSCAPE. By default, ODS PDF destination will create the bookmark on the left; you may use NOTOC option to turn it off. To output PDF to the web, the special file reference _WEBOUT is used instead of giving a physical file name in the local drive.

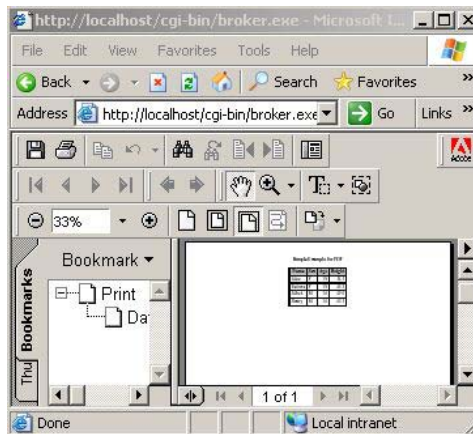


Figure 2. PDF file generated by warmup example

As you can see, the PDF file is displayed within the Internet Explorer. If you click the BACK button of the browser, it goes back the web page of Figure 1. Sometime, it is more preferable that the table is displayed on the top and there is a button or link text at the bottom that can be used to open or save as PDF file. The key issue becomes how to append a button or link text to the HTML table.

USE NOBOT OPTION AND DATA_NULL_STATEMENT

By default, ODS HTML will automatically write the HTML tags at the bottom. In some scenario, you may want to append your own HTML codes. Option NO_BOTTOM_MATTER makes it possible and allows you to add your own codes to the existing file. NOBOT is the alias of this option. Similarly, NO_TOP_MATTER (alias NOTOP) option allows adding HTML codes on the top.



Figure 3. Button created with NOBOT option

```
ods html body=_webout(nobot);
title 'Display Class Data Set';
proc print data=pdfdata.class noobs;
run;
ods html close;
```

```

data _null_;
  file _webout;
  put '<br>';
  put '<form name="pdfform" method="post" action="/cgi-bin/broker.exe">';
  put '<center>';
  put '<input type="submit" name="Button" value="Save As PDF">';
  put '</center>';
  put '<input type="HIDDEN" name="_SERVICE" value="default">';
  put '<input type="HIDDEN" name="_PROGRAM" value="pdfsrc.CreatePDF1.sas">';
  put '</form>';
  put '</body>';
  put '</html>';
run;

```

The PUT statements are used here to add more HTML tags to the generated HTML page. Don't forget the close tags of </body> and </html>. If the "Save As PDF" button is clicked, the program CreatePDF1.sas is called to create the PDF file.

The result is the same as Figure 2. The PDF file still embeds within the browser. However, an interesting thing occurs if you click the BACK button of the browser. Instead of backing to the HTML table page, it goes back to the previous web page that generates the HTML table. Apparently, it is not what we want since we still want to see the HTML table. After clicking the BACK button, it is supposed to go back to the parent URL of the history. The generated PDF still thinks the previous web page as its parent.

USE FOOTNOTE TO ADD A LINK

The FOOTNOTE or TITLE statement allows you to insert some HTML tags to create a link. Therefore, it provides a good solution that you may use the method mentioned early. The PDF file can be generated in advance and sent to the web folder. Then you can create a link that points to this PDF file.

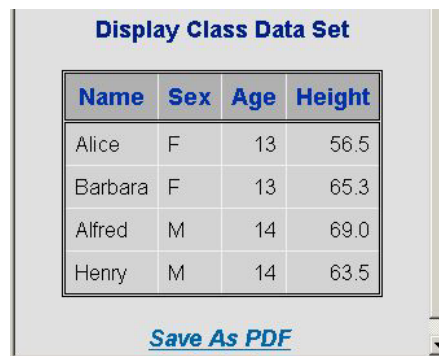
```

options nodate nonumber;
ods pdf body='c:\website\sample\class.pdf' NOTOC;
title 'Display Class Data Set';
proc print data=pdfdata.class noobs;
run;
ods pdf close;

%let fn = '<A HREF="http://localhost/sample/Class.pdf">Save As PDF</A>';
ods html body=_webout;
proc print data=pdfdata.class noobs;
title Bold height=2 font=arial 'Display Class Data Set';
footnote height=2 &fn;
run;
ods html close;

```

The resulting HTML page looks like:



Name	Sex	Age	Height
Alice	F	13	56.5
Barbara	F	13	65.3
Alfred	M	14	69.0
Henry	M	14	63.5

[Save As PDF](http://localhost/sample/Class.pdf)

Figure 4. Link created by footnote

If you click on the link, the PDF report is opened within the browser. If you click on the BACK button of the browser, it goes back to the HTML table page. This is the big difference from the way of NOBOT option. Also, please note that the link text "Save As PDF" is outside of the HTML table.

USE COMPUTE BLOCK OF PROC REPORT

The LINE statement in PROC REPORT can be used to write customized summary. But at here, it is used to create a link on the HTML page. Like the above footnote method, the PDF file is produced beforehand. At the end of the report, COMPUTE block is applied to prepare the link text. Apparently this link becomes part of the HTML table.

By using the TARGET attribute, the PDF file is opened in a new pop-up window if you click on the link. The value of “_blank” always causes the link to be loaded in a new, unnamed blank window.

```
ods html body=_webout(title='Fun with Class') style=styles.statdoc;
title Bold height=2 font=arial 'Display Class Data Set';
proc report data=pdfdata.class nowd;
  column Sex Name Age Height;
  define Sex / Group;
  define Name / Display;
  define Age / Display;
  define Height / mean ;
  break after sex / summarize;
  compute after sex;
    name = 'Average: ';
  endcomp;
  compute after;
    line '<A HREF="http://localhost/sample/class.pdf" target="_blank">Save As PDF</A>';
  endcomp;
run;
ods html close;
```

Sex	Name	Age	Height
F	Alice	13	56.5
	Barbara	13	65.3
<i>F</i>	<i>Average:</i>		<i>60.9</i>
M	Alfred	14	69
	Henry	14	63.5
<i>M</i>	<i>Average:</i>		<i>66.25</i>

[Save As PDF](#)

Figure 5. Link created by compute block

USE POSTHTML OPTION IN TEMPLATE

In the style definition, POSTHTML option is used to insert HTML codes after the table or cell. Obviously it can be used to append a link that will open a PDF file.

```
proc template;
  define style Styles.Mystatdoc;
    parent = Styles.statdoc;
    style body from output /
    posthtml = '<BR><CENTER><A HREF="../sample/Class.pdf">Save As PDF</A></CENTER>';
  end;
run;
```

First, run the PROC TEMPLATE to produce a style called Mystatdoc and it is stored under SASUSER.TEMPLAT. When you apply this style, just use STYLE option on the ODS HTML statement:

```
ods html body=_webout style=styles.mystatdoc;
SAS procedure to produce report...
ods html close;
```

The result is quite similar to that of figure 4.

USE APPSRV_HEADER FUNCTION

All output created by Application Dispatcher programs must have an HTTP header. The most basic HTTP header is the content-type header, such as: Content-type: text/html. It informs the browser what type of output should be produced.

In order to generate PDF file, the content-type is defined to: Application/PDF.

```
%LET RV=%SYSFUNC(APPSRV_HEADER(CONTENT-TYPE, APPLICATION/PDF));
ods pdf body=_webout;
  SAS procedures to produce report...
ods pdf close;
```

The result is the same as the one generated in figure 2. However, if you use the Content-Disposition header field with type "attachment", it forces a file download dialog pop up and gives you the choice to view the file or store it to local machine.

```
%LET RV=%SYSFUNC(APPSRV_HEADER(CONTENT-DISPOSITION, %STR(ATTACHMENT;
FILENAME=CLASS.PDF)));
options NODATE NONUMBER;
options TOPMARGIN="0.5 in" BOTTOMMARGIN="0.4 in"
  LEFTMARGIN="0.7 in" RIGHTMARGIN="0.7 in";
ods pdf body=_webout NOTOC;
title 'Simple Example for PDF';
proc print data=pdfdata.class noobs;
run;
ods pdf close;
```

The parameter FILENAME specifies the generated PDF file name. You can see on the file download dialog (Figure 6), the file is called class.pdf. If you click on "Open" button, the PDF file is opened directly in Acrobat Reader. This time it is not embedded in browser at all. It is quite different from the above methods since all the previous PDF files are embedded with browser. If you click on "Save" button, the save as dialog pops up and then you can save it to anywhere on your machine. The default file name is class.pdf. You can cancel view or save action by clicking "Cancel" button.

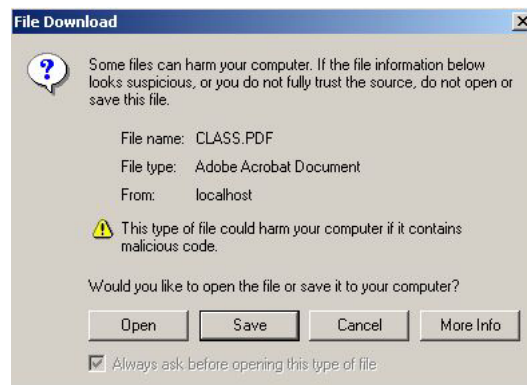


Figure 6. File download dialog

GENERATE A DYNAMIC PDF REPORT ON POP-UP WINDOW

In some situations, a dynamic PDF report with pop-up window is more desirable. When the resulting PDF file created by the previous ways is displayed within the browser, the problem arises if you accidentally click on the close icon of the browser. You intend to only close the PDF file, but the whole web application is also shut down. You have to restart the browser to continue viewing activities. By using a pop-up window, you can close the PDF file without affecting the web application.

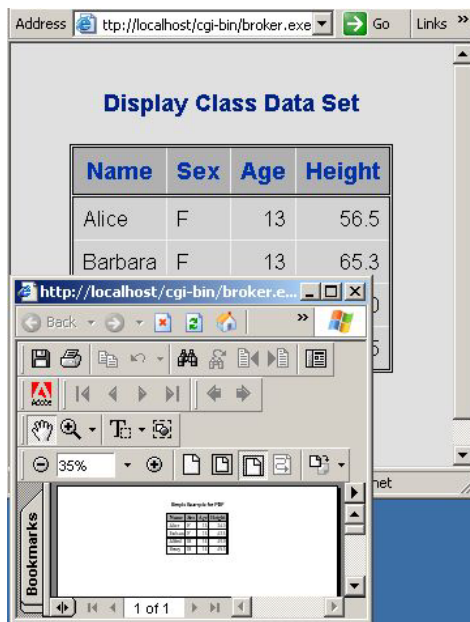


Figure 7. PDF file on pop-up window

Figure 7 shows a PDF file is generated dynamically in a pop-up window. The PDF file is opened in a new window on top of the HTML page and no PDF file is created in advance. It is further modification of the method using NOBOT option. The DATA_NULL_ steps have been changed to the following:

```
data _null_;
  file _webout;
  put '<SCRIPT LANGUAGE="JavaScript">';
  put 'function OpenWin()';
  put '{';
  put 'window.open("http://localhost/cgi-bin/broker.exe?_Service=default&_Program=pdfsrc.createPDF1.sas", "NewWin", "toolbar,width=400,height=400, resizable");';
  put '}';
  put '</SCRIPT>';
  put '<br><br>';
  put '<center><button onclick="OpenWin();">Save As PDF</button></center>';
run;
```

When the button "Save As PDF" is clicked, the onclick event triggers a user-defined function OpenWin(), which in turn, calls the JavaScript function window.open(). The syntax of window.open() function is:

```
window.open("URL", "Window Name", "Options");
```

For the first argument, the broker's URL together with service name and program name are passed. You can also add other variable and value pairs separated by the symbol "&". The second argument specifies the new window name. The third argument controls the features of the new window. The available choices are:

- menubar - the row that appears on top of the browser, including File, Edit, View.
- toolbar - browser toolbar including Back, Forward, Stop, Refresh, Home.
- location - the text area to type address.
- width - the window width in pixels.
- height - the window height in pixels.
- resizable - determines if the window size can be changed.

This method provides more controls on the pop-up window. The drawback is the title of browser shows the service name and program name. If "Save A Copy" button of Acrobat Reader is clicked, this long string will be used as the default file name. Of course, you can not really hide those information. All the information can be seen from view source menu of the browser.

CONCLUSION

This paper discusses several ways to present PDF report from the web application. Advantages and disadvantages of each method have also been addressed. It's up to you to decide what type of PDF file to generate, either static or dynamic before you start the project. In addition, most of the skills can be applied to generate other common file formats, such as EXCEL or CSV.

REFERENCES

SAS Institute Inc. (1999), *SAS Online Doc, Version 8*, Cary, NC: SAS Institute Inc.
K. P. Delaney. *ODS PDF: It's not just for printing anymore*, SUGI 28, Seattle, WA, 2003.
P. M. McGown. *Using SAS ODS to Create Adobe PDFs from SAS/GRAPH Output*, SUGI 27, Orlando, FL, 2002.

ACKNOWLEDGMENTS

The author would like to thank Mr. John Baj for reviewing this paper and providing good comments.

CONTACT INFORMATION

Please send your comments and questions to:

Ben Xu
Center for Governmental Studies
Northern Illinois University
148 N 3rd St
De Kalb, IL 60115
Work Phone: (815)753-4790
Email: dxu@niu.edu

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.