Paper 009-30

Instant SAS® Applications With VBScript, Jscript, and dHTML

Brian Fairfield-Carter, PRA International, Victoria, BC
Tracy Sherman, PRA International, Victoria, BC
Stephen Hunt, PRA International, Victoria, BC

ABSTRACT

For a programmer with limited experience outside of Base SAS, the objective of setting up an application that runs SAS as a 'back-end', and that works with SAS source code outside of the SAS system, can be somewhat daunting. Naturally, the first thing you'd ask would be 'what is the simplest, cheapest environment that I can get started with?' A good answer seems to lie in the combination of Windows Scripting technologies (VBScript and Jscript), dynamic HTML, and SAS integration with the Windows Component Object Model (COM). HTML provides a very simple environment in which to create a graphical user interface, which can be made dynamic (as in, responsive to user actions) by placing VBScript and Jscript components between <SCRIPT> and </SCRIPT> tags. The script components can, in turn, launch SAS sessions, run SAS code, and retrieve and display SAS output and data. This environment is attractive as a starting point for applications development because it is simple and cheap: it requires no additional software (all the necessary language interpreter/compiler facilities already exist in the Windows operating system and web browser), languages (particularly VBScript and HTML) are very easy to learn, and components can be easily borrowed from other HTML pages.

INTRODUCTION

Getting started is usually the most difficult phase of any programming venture, particularly when it involves the application of new ideas and/or technology. This paper is intended to introduce some of the basics of SAS/ActiveX automation, and to provide some functioning building blocks to enable programmers with little or no familiarity with applications development to start creating their own dynamic applications.

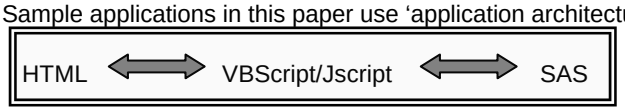
The examples in this paper are meant to address a couple of different themes:

1. **Providing SAS functionality to non-SAS users**. For example, to allow non-SAS-programmers to run programs, and to filter and display SAS data.
2. **Accelerating the process of SAS code-writing by using 'dynamically-generated' code**. In some cases, you may want to edit and test the code generated by another program before importing or using it, or parse the code in a point-and-click interface to ensure certain parameters are met.

For simplicity, Visual Basic Scripting Edition (VBScript) comprises the bulk of the sample script code, though be forewarned that while Internet Explorer supports both VBScript and Jscript, Netscape Navigator only supports Jscript.

APPLICATION ARCHITECTURE

Sample applications in this paper use 'application architecture' that can be illustrated as:



In other words, a graphical user interface is written in HTML (and launched in a web browser), scripts are fired in response to user actions (button clicks, selections, etc.), and these scripts launch SAS sessions and submit SAS statements.

SOME VERY SIMPLE HTML

HTML is a 'mark-up' language, in that text and graphical attributes are set using <tag></close tag> syntax. For example, to display text in bold, you would use:

```
<b>bold text</b>
```

Objects such as buttons, text entry fields, and 'text areas' can be added to a page via:

```
<INPUT TYPE="Button" NAME="MyButton" VALUE="MyButton">
<INPUT TYPE=text NAME="data_library" size=60>
<TEXTAREA class="formField" NAME="Syntax" rows="20" cols="90"></TEXTAREA>
```

To make an application dynamic, scripts are added to a page within <SCRIPT></SCRIPT> tags:

```
<SCRIPT FOR="MyButton" EVENT="onClick" LANGUAGE="VBScript">
</SCRIPT>
```

Scripts associated with objects (button, text fields) are 'event scripts', in that they respond to events in the user interface.

VBSCRIPT AND JSCRIPT

Though 'object.method' syntax and program organization in VBScript and Jscript may seem unfamiliar at first, there are actually many analogies to Base SAS, particularly in the areas of looping, conditional statements, and the

similarity in definition and use between user-defined functions and SAS macros. Documentation and language references can be found on the Microsoft Developers Network (MSDN) Website (http://msdn.microsoft.com).

SAMPLE APPLICATIONS

SAMPLE APPLICATION 1 – THE BASICS

This first example is pared down to the very basics, and consists of an HTML page with just 1 button. This button fires an 'on-click' script that launches a SAS session and runs a pre-written SQL query.

Say we had a program, saved as 'c:\sas\sasuser\sae.sas', that refreshed a Serious Averse Event dataset:

```
libname source "\\server\sas";
libname temp "c:\sas\sasuser";
proc sql;
  create table temp.sae as select * from source.ae where aeser=1;
quit;
```

To run this program from a web interface, we'd create an html page with a button:

```
<INPUT TYPE="Button" NAME="RunSAS" VALUE="Update SAEs">
```

We'd then add an 'on-click' event script:

```
<SCRIPT FOR="RunSAS" EVENT="onClick" LANGUAGE="VBScript">
Dim objSAS
Set objSAS = CreateObject("SAS.Application.8") 'Launch a SAS session
objSAS.Submit("%include 'c:\sas\sasuser\sae.sas';endsas;") 'Submit statements
</SCRIPT>
```

Clicking the button now launches a SAS session, runs the SAS program (which refreshes the SAE dataset), and then ends the SAS session.

SAMPLE APPLICATION 2 – A SAS 'DATA BROWSER'

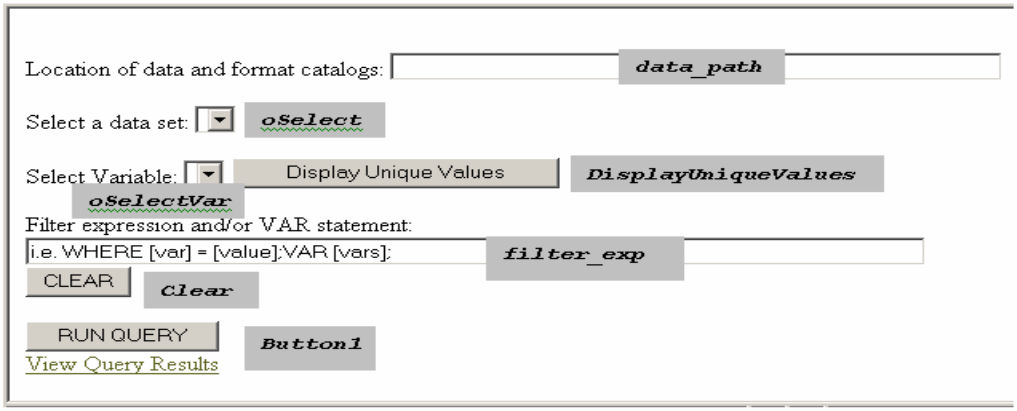
The above example of course raises a number of questions: what if you want to be able to select from among a number of different programs? What if you want to view output? What if you want to be able to build and submit SAS statements dynamically?

The following example allows a user to enter the path to a data library into a text field, then select a dataset in this library from a scrolling list, and type 'where' and 'var' statements to be applied to the dataset in the PRINT procedure. Results of the PROC PRINT are written to an HTML page (via ODS), which is linked to the main page.

Creation of the page starts with the addition of objects (text entry boxes, buttons, selection lists, and links):

```
<FORM NAME="Form1">
<br> Location of data and format catalogs: <input type=text name="data_path"
size=60>
<br><br>Select a data set: <select ID="oSelect"></select><br><br>
Select Variable:<select ID="oSelectVar"></select>
<INPUT TYPE="Button" NAME="DisplayUniqueValues" VALUE="Display Unique Values">
<br><br>Filter expression and/or VAR statement: <br>
<input type=text name="filter_exp" size=90> <br>
<INPUT TYPE="Button" NAME="Clear" VALUE="CLEAR"> <br><br>
<INPUT TYPE="Button" NAME="Button1" VALUE="RUN QUERY"><td height="58">
<div align="left"><a href="C:\SAS\SASUSER\QUERY.html">View Query
Results</a></div></td>
```

These statements are rendered in Internet Explorer as:



USER INTERFACE EVENTS

The following sequence of events can now occur:

1. The user enters the path name for a data library in 'data_path'
2. An 'on-change' script fires for 'data_path', which determines the number of data sets in the target directory, creates a VB array populated with the file name of each dataset, and then loops through the array assigning each element as an entry in the scrolling selection list 'oSelect'
3. The user selects a dataset, causing an 'on-click' script for 'oSelect' to fire, which in turn submits statements to a SAS session that write a list of the variables on the selected dataset to a temporary text file. This text file is then used to populate a VB array, which in turn is used to assign each element (variable name) as an entry in the 'oSelectVar' scrolling selection list
4. The user types 'where' and/or 'var' statement(s) (for use in PROC PRINT) in the 'filter_exp' text field. Any variable selected from 'oSelectVar' is appended to the text in 'filter_exp'; if the 'display unique values' button is clicked, statements are submitted to the SAS session to write a list of all unique values of the selected variable to a temporary text file, the contents of which are then displayed in a dialog box
5. Once the 'where' and/or 'var' statements are created, the user clicks the 'run query' button, which submits ODS/PROC PRINT statements to the SAS session, producing ODS HTML PROC PRINT output. This output is then viewed by clicking the 'view query results' link.

STARTUP SCRIPTS

Startup scripts (not associated with events) are executed when the page is launched; these may be used to populate text fields with data, launch other applications, and compile user-defined functions. The following startup script (1) creates an instance of the File System Object for text-streaming and file handling operations, (2) creates a reference to the folder to be used to hold temporary text files, (3) deletes any temporary files left over from previous runs, (4) launches a SAS session, (5) initializes the 'query string' with an example, (6) compiles a function to determine the number of data sets in the target library, (7) compiles a function to write path/file names for all data sets in the target library to an array, (8) compiles a string-manipulation function to separate 'file.extension' strings into file name and file extension, and check file extension to determine if the file is a SAS data set, (9) compiles a function to determine the size of an array (this is required because Jscript can only call *user-defined* VBScript functions, meaning the built-in VBScript function that returns array size can't be called from a Jscript function), and (10) compiles a function to determine the number of variable names the 'variables' selection list will need to hold:

```
<SCRIPT LANGUAGE="VBScript">
 Const SasFolder = "C:\SAS\SASUSER"
 Dim FSO, SasPth, objSAS, Wshell
 Set FSO = CreateObject("Scripting.FileSystemObject") '(1)
 Set SasPth = FSO.GetFolder(SasFolder)                '(2)
 Set Wshell=CreateObject("WScript.Shell")

 '(3) Clear any files left over from last run
 On Error Resume Next
 FSO.DeleteFile SasPth & "\UNIQUE.txt"
 FSO.DeleteFile SasPth & "\ATTRB.txt"

 Set objSAS = CreateObject("SAS.Application.8")        '(4)
 objSAS.Visible=True

 '(5) Initialize query string with example
 Form1.filter_exp.value="i.e. WHERE [var] = [value];VAR [vars]; "
 '(6) Function to determine required array size for data sets list
 Function GetArraySize()
  Dim Pth1, i
  Set Pth1 = FSO.GetFolder(Form1.data_path.value)
  i=0
     For Each File In Pth1.Files
        i=i+1
     Next
   GetArraySize = i
 End Function
 '(7) Function to create array to hold full path names for data sets
 Function CreateVBArray()
  On Error Resume Next
  Dim Pth, iter, iNewSize
  Set Pth = FSO.GetFolder(Form1.data_path.value)
  iNewSize = GetArraySize()
```

```
 iter=0
 Dim a()                 'Declare array
 ReDim a(1, iNewSize)    'Re-size to correct dimensions
    For Each File In Pth.Files
       iter=iter+1
       a(1,iter) = File.Name  'Write file name to array element
    Next
  CreateVBArray = a
End Function
'(8) Function to separate data set name and file extension, and confirm SAS dataset
Function GetFileNm(FileRef)
 Dim extens, prefix, FileShort
 extens=split(strreverse(FileRef),".",-1,1) 'Split into array of 2 elements
 prefix=strreverse(extens(lbound(extens))) 'Capture the file extension
 FileShort=trim(replace(FileRef,"." & prefix," ")) 'Capture the data set name
 If ucase(prefix)="SAS7BDAT" or ucase(prefix)="SD2" Then
   GetFileNm = FileShort  'If SAS data set, return the data set name
 Else
   MsgBox("This is not a SAS data set")
   GetFileNm = "_null_"  'Otherwise, return _null_
 End If
End Function
'(9) Function to determine the size of an array on a given dimension
Function ArrayDim(ArrayName)
   ArrayDim=UBound(ArrayName,2)
End Function
'(10) Function to determine the number of var names which the vars array will need
to hold
Function GetNVars()
   Dim S, File, i
    i=0
   Set File = FSO.GetFile(SASPth & "\ATTRB.txt")
   Set TextStream = File.OpenAsTextStream
   Do While Not TextStream.AtEndOfStream
      S = TextStream.ReadLine & NewLine
      i=i+1
   Loop
   TextStream.Close
   GetNVars=i
End Function
</SCRIPT>
```

The following (Jscript) startup script (1) compiles a function to loop through the array of data set names and write the value of each element to an option in the ‘data sets’ selection list, (2) compiles a function to clear the ‘variables’ selection list (otherwise if a new data set is selected, variable names are appended to the existing list), and (3) compiles a function to loop through the array of variable names and write the value of each element to an option in the ‘variables’ selection list:

```
<SCRIPT LANGUAGE="JScript">
 //(1) Function to populate data sets list
 function GetItemTest(vbarray)
 {
   var i;
   var a = new VBArray(vbarray);
   var iNewSize = GetArraySize();
   for (i = 1; i <= iNewSize; i++)
   {
          var oOption = document.createElement("OPTION");
          Form1.oSelect.options.add(oOption);
          oOption.innerText = (a.getItem(1, i));
          oOption.Value = (a.getItem(1, i));
    }
 }
 //(2) Function to clear vars selection list
 function ClearVarList(NumEntries)
```

```
  {
    var i;
    for (i = 1; i <= NumEntries; i++)
    {
      var oOption = document.createElement("OPTION");
      Form1.oSelectVar.options.remove(oOption);
    }
  }
  //(3) Function to populate vars selection list
  // Note that user-defined VB Function 'ArrayDim' is used because
  // VB functions like 'UBound' can't be called directly in Jscript
  function GetVarList(vbarray)
  {
    var i;
    var a = new VBArray(vbarray);
    var iNewSize = ArrayDim(vbarray);
    for (i = 1; i <= iNewSize; i++)
    {
           var oOption = document.createElement("OPTION");
           Form1.oSelectVar.options.add(oOption);
           oOption.innerText = (a.getItem(1, i));
           oOption.Value = (a.getItem(1, i));
     }
  }
</SCRIPT>
```

EVENT SCRIPTS

Event scripts run in response to user actions such as button clicks, selections, and text entry.

"**data_path**": The following script fires each time the value in 'data_path' changes. The inner function call occurs first – CreateVBArray creates the array of SAS dataset names, which is then passed to GetItemTest, which populates the 'data sets' selection list 'oSelect' with the contents of the array. Note that VBScript functions can call Jscript functions, and that references to VB arrays can be passed as arguments to Jscript functions.

```
<SCRIPT FOR="data_path" EVENT="onChange" LANGUAGE="VBScript">
 GetItemTest(CreateVBArray())
</SCRIPT>
```

"**oSelect**": This script is called when a data set is selected in the 'data sets' selection list; it submits SAS (PROC CONTENTS) statements to get the names of variables on the selected data set and to write these to a text file. The text file is then used to populate a VB array which is in turn used to populate the 'variables' selection list.

```
<SCRIPT FOR="oSelect" EVENT="onClick" LANGUAGE="VBScript">
 'Run PROC CONTENTS on selected dataset and write results to vars selection list
 Dim iSelectedIndex, FileNm
 iSelectedIndex = Form1.oSelect.SelectedIndex  'Get index ref to selected item
 FileNm = GetFileNm(Form1.oSelect.options(iSelectedIndex).text) 'Get text

   'Run the PROC CONTENTS SAS program, and write list of variables to text file
   objSAS.Submit("LIBNAME LIBRARY '" & Form1.data_path.value & "';LIBNAME IN '" &
    Form1.data_path.value & "';")
   objSAS.Submit("PROC CONTENTS DATA=IN." &
    GetFileNm(Form1.oSelect.options(iSelectedIndex).text) & " OUT=ATTRB
    NOPRINT;RUN;")
   objSAS.Submit("DATA _NULL_;SET ATTRB;FILE 'C:\SAS\SASUSER\ATTRB.txt';IF _N_=1
    THEN DO;B='--Select Variable--';PUT B;PUT NAME;END;IF _N_ NE 1 THEN DO;PUT
    NAME;END;RUN;")
   Dim AttrbExists, iNewSize, iter1
   iter1=0
   iNewSize=0
   AttrbExists=False
     AttrbExists=FSO.FileExists(SASPth & "\ATTRB.txt") 'Process-timing problem
     If AttrbExists=True Then                          '(still needs work)
      iNewSize=GetNVars()
        If iNewSize>0 Then
         Wshell.PopUp "This data set contains " & iNewSize-1 & " variables", 1
```

```
          Dim b()                 'Declare array
          ReDim b(1, iNewSize)    'Re-size to correct dimensions
          Set File = FSO.GetFile(SASPth & "\ATTRB.txt")
          Set TextStream = File.OpenAsTextStream 'Open variable-list text file
          Do While Not TextStream.AtEndOfStream
           iter1=iter1+1
           S = TextStream.ReadLine & NewLine   'Read each line...
           b(1, iter1) = S                     '...and write to array element
          Loop
          TextStream.Close
          ClearVarList(100)      'Clear the 'variables' selection list
          GetVarList(b)          'Call Jscript function to re-populate
         End If                  'the selection list with the VB array 'b'
      End If
   </SCRIPT>
```

"**oSelectVar**": The following script is called when a variable is selected in the 'variables' selection list; it appends the variable name to the current contents of the 'filter_exp' text field, and then submits a SAS (PROC SORT NODUPKEY) program to capture unique values of the selected variable, and writes these to a text file.

```
   <SCRIPT FOR="oSelectVar" EVENT="onChange" LANGUAGE="VBScript">
    'Add var name to query string
    Dim VarSelectedIndex, FileSelectedIndex
    VarSelectedIndex = Form1.oSelectVar.SelectedIndex
    FileSelectedIndex = Form1.oSelect.SelectedIndex
    Form1.filter_exp.value= Form1.filter_exp.value &
     Form1.oSelectVar.options(VarSelectedIndex).text & " "

    'Run SAS program to get unique values of the selected variable
    objSAS.Submit("LIBNAME LIBRARY '" & Form1.data_path.value & "';LIBNAME IN '" &
     Form1.data_path.value & "';")
    objSAS.Submit("DATA IN(KEEP=" & Form1.oSelectVar.options(VarSelectedIndex).text &
     " RENAME=(" & Form1.oSelectVar.options(VarSelectedIndex).text & " = VAR_));
     SET IN." & GetFileNm(Form1.oSelect.options(FileSelectedIndex).text) & ";
     PROC SORT NODUPKEY DATA=IN;BY VAR_;RUN;")
    objSAS.Submit("DATA _NULL_;SET IN;FILE 'C:\SAS\SASUSER\UNIQUE.txt';PUT VAR_;RUN;")
   </SCRIPT>
```

"**DisplayUniqueValues**": This script is called when the 'display unique values' button is clicked, and displays the contents of the text file created by the 'onChange' script for 'oSelectVar' in a message box.

```
   <SCRIPT FOR="DisplayUniqueValues" EVENT="onClick" LANGUAGE="VBScript">
    Dim UniqueExists
    UniqueExists=False
     UniqueExists=FSO.FileExists(SASPth & "\UNIQUE.txt")
     If UniqueExists=True Then
       Set File = FSO.GetFile(SASPth & "\UNIQUE.txt")
       Set TextStream = File.OpenAsTextStream
       Do While Not TextStream.AtEndOfStream
          S=S & TextStream.ReadLine & NewLine & ", "
       Loop
       TextStream.Close
       MsgBox(S)
     End If
   </SCRIPT>
```

"**Button1**": The following script captures the contents of the 'filter_exp' text field, and incorporates this into a SAS (PROC PRINT) program, writing the results to an ODS/HTML output destination, which is referenced by the 'view query results' link.

```
   <SCRIPT FOR="Button1" EVENT="onClick" LANGUAGE="VBScript">
    'Submit a simple PROC PRINT/ODS HTML SAS program, based on user selections
    Dim iSelectedIndex
    iSelectedIndex = Form1.oSelect.SelectedIndex

    'Run the SAS program
    objSAS.Submit("LIBNAME LIBRARY '" & Form1.data_path.value & "';LIBNAME IN '" &
```

```
  Form1.data_path.value & "';")
 objSAS.Submit("ODS HTML BODY='QUERY.html' PATH='C:\SAS\SASUSER' RS=NONE;")
 objSAS.Submit("PROC PRINT DATA=IN." &
  GetFileNm(Form1.oSelect.options(iSelectedIndex).text) & ";")
 objSAS.Submit("" & Form1.filter_exp.value & ";RUN;ODS HTML CLOSE;")
</SCRIPT>
```

While this application may not in of itself appear particularly useful, it does illustrate many common and reusable components, particularly data entry fields, selection lists, buttons, and interaction with external applications.

SAMPLE APPLICATION 3 – A PROC REPORT STATEMENT GENERATOR

This example uses a pre-written SAS macro that captures 'meta-data' (variable names and attributes) for the last (most recent) data set added to the WORK directory, and writes PROC REPORT statements based on this meta-data to a text file. The PROC REPORT statements can then be edited in a text area and submitted to the SAS session. A quick sum of column widths set in 'DEFINE' statements (plus spacing between columns) gives 'column space used' at the click of a button. A second set of buttons and associated text area allow the user to generate dummy data (such a feature might be adapted for uses such as the production of table shells).

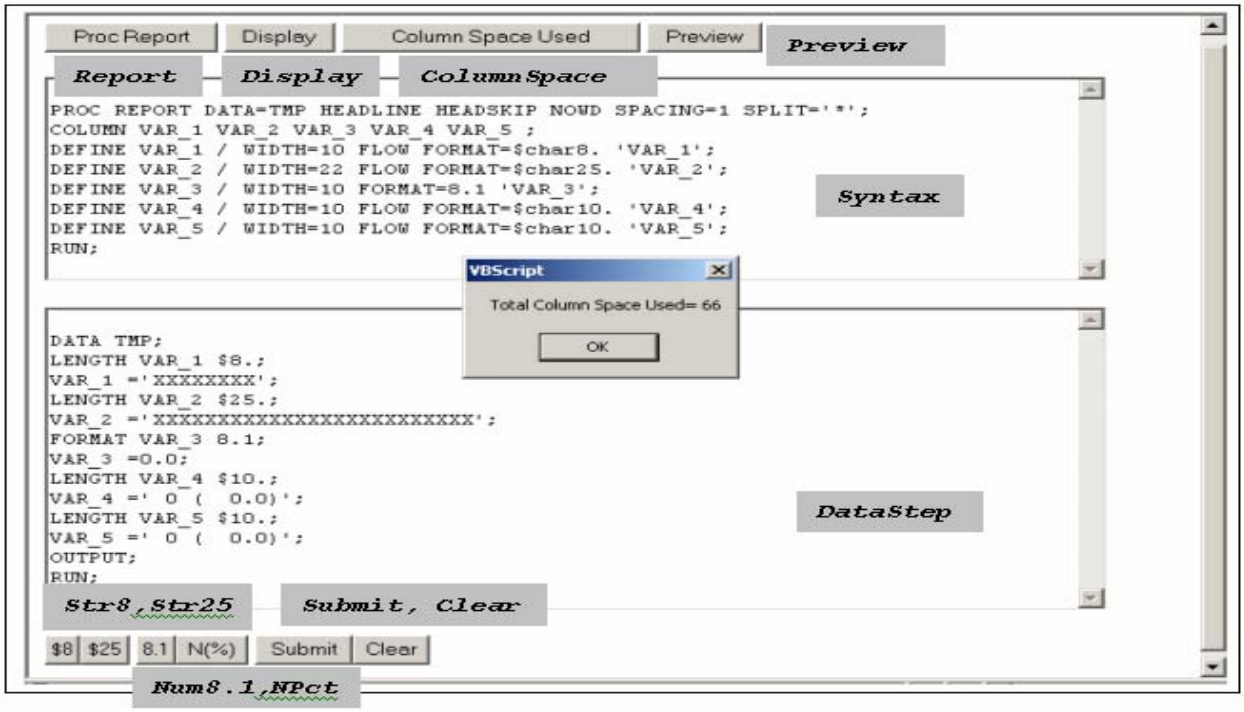
First, here is the SAS 'back-end', used to generate PROC REPORT statements (saved as "procreport.sas"):

```
%macro procreport;
proc sql noprint;
/*First, get the last dataset added to the WORK library*/
   select trim(left(memname)) into :dsetnm from
    (select libname, memname, memtype, crdate, modate from sashelp.vtable
      where libname="WORK" and memtype="DATA" and memname ne "ATTRB"
      group by libname having modate=max(modate));
/*Create lists of variables and attributes*/
   select count(*), name, length,
          case when label^="" then compress(label,"'") else "none" end,
          case when format^="" then format else "none" end, type, varnum into
    :nvars_,:name_ separated by "~",:length_ separated by "~",
    :label_ separated by "~",:format_ separated by "~",:type_ separated by "~",
    :varnum_ separated by "~" from dictionary.columns where libname="WORK"
    and memname="&dsetnm"; quit;
/*Create proc report statements*/
data _null_;file "c:\sas\sasuser\report_.txt";
  put "PROC REPORT DATA=&dsetnm HEADLINE HEADSKIP NOWD SPACING=1 SPLIT='*';";
  put "COLUMN " %do i=1 %to &nvars_; "%scan(&name_,&i,'~') " %end; ";";
  %do i=1 %to &nvars_;
   %if "%scan(&label_,&i,'~')"="none" %then %let lab_=%scan(&name_,&i,'~');
   %else %let lab_=%scan(&label_,&i,'~');
   /*(If label is missing, use variable name)*/
   %if "%scan(&type_,&i,'~')"="char" %then %do;
     %if "%scan(&format_,&i,'~')"="none" %then %let
      form_=$%scan(&type_,&i,'~')%scan(&length_,&i,'~').;
     %else %let form_=%scan(&format_,&i,'~'); %end;
   %else %if "%scan(&type_,&i,'~')"="num" %then %do;
     %if "%scan(&format_,&i,'~')"="none" %then %let form_=%scan(&length_,&i,'~').;
     %else %let form_=%scan(&format_,&i,'~'); %end;
   /*(If format is missing, use combination of data type and length)*/
   %if "%scan(&type_,&i,'~')"="char" %then %do;
     %if %eval(%scan(&length_,&i,'~') > 25)=1 %then %let len_=25;
     %else %if %eval(%scan(&length_,&i,'~') < 10)=1 %then %let len_=10;
     /*(Arbitrarily adjust widths of very long and very short text variables)*/
     %else %let len_=%scan(&length_,&i,'~');
       put "DEFINE %scan(&name_,&i,'~') / WIDTH=&len_ FLOW FORMAT=&form_ '" "&lab_"
"';" ; %end;
   %else %if "%scan(&type_,&i,'~')"="num" %then %do;
     put "DEFINE %scan(&name_,&i,'~') / WIDTH=10 FORMAT=&form_ '" "&lab_" "';" ;
  %end;%end;
  put "RUN;";
run;
data attrb; format name $15.; %do i=1 %to &nvars_;
  name="%scan(&name_,&i,'~')"; output; %end; run;
```

```
x 'copy "c:\sas\sasuser\report_.txt" "c:\sas\sasuser\report_.sas"';
%mend procreport;
%procreport;
```

The HTML user Interface consists of the following components:

```
<FORM NAME="Form1">
<INPUT TYPE="Button" NAME="Report" VALUE="Proc Report">
<INPUT TYPE="Button" NAME="Display" VALUE="Display">
<INPUT TYPE="Button" NAME="ColumnSpace" VALUE="Column Space Used">
<INPUT TYPE="Button" NAME="Preview" VALUE="Preview">
<br><br><textarea class="formField" name="Syntax" rows="5" cols="45"></textarea>
<br><br><textarea class="formField" name="DataStep" rows="5" cols="45"></textarea>
<br><br><INPUT TYPE="Button" NAME="Str8" VALUE="$8"><INPUT TYPE="Button"
NAME="Str25" VALUE="$25">
<INPUT TYPE="Button" NAME="Num8.1" VALUE="8.1"><INPUT TYPE="Button" NAME="NPct"
VALUE="N(%)">
<INPUT TYPE="Button" NAME="Submit" VALUE="Submit"><INPUT TYPE="Button" NAME="Clear"
VALUE="Clear">
```



STARTUP SCRIPTS

The following (VBScript) startup script (1) creates an instance of SAS, (2) creates a File System Object and deletes old temporary files (3) gets a reference to the folder to store temporary files, and (4) compiles a generic function to write the content of either the 'syntax' or the 'datastep' text area to a text file.

```
<SCRIPT LANGUAGE="VBScript">
  Dim objSAS
  Set objSAS = CreateObject("SAS.Application.8")        '(1)
  objSAS.Visible = True
  objSAS.Submit("OPTIONS NOXWAIT NOXSYNC LS=120 PS=41;")

  Dim FSO, SasPth
  Set FSO = CreateObject("Scripting.FileSystemObject")  '(2)
  If (FSO.FileExists("C:\SAS\SASUSER\REPORT_.sas"))=True Then
    FSO.DeleteFile "C:\SAS\SASUSER\REPORT_.sas"
  End if
  Const SasFolder = "C:\SAS\SASUSER"
  Set SasPth = FSO.GetFolder(SasFolder)                 '(3)
  '(4) Function to write text-area contents to a text file
  Function Capture(FileNm)
   Dim TxtFile, writetxt
```

```
    If FileNm="REPORT_" Then
      writetxt = Form1.Syntax.value
    Else
      writetxt = Form1.DataStep.value
    End If
    Dim TextStream
    Set TextStream = SasPth.CreateTextFile(FileNm & ".txt")
    TextStream.Write(writetxt)
    TextStream.Close
  End Function
</SCRIPT>
```

The following (Jscript) startup script compiles functions to display text in the text areas. Note that the text areas are referenced through 'element' numbers, where the 'proc report' button is element 0, the first element in the page, and the text areas are elements 4 and 5.

```
<SCRIPT LANGUAGE = "JScript">
  var cr="\r\n"
  var textout=" ";
  function cleartext()
  {
  textout=" ";
  }
  function addtext(textin)
  {
  textout=textout + cr + textin; //Concatenate carriage return plus input text
  self.document.forms[0].elements[4].value = textout; //Update the displayed value
  }
  function addtext1(textin)
  {
  textout=textout + cr + textin;
  self.document.forms[0].elements[5].value = textout;
  }
</SCRIPT>
```

EVENT SCRIPTS

"**Report**": This script runs the 'procreport' macro, creating PROC REPORT statements for the most recent WORK dataset:

```
<SCRIPT FOR="Report" EVENT="onClick" LANGUAGE="VBScript">
 objSAS.Submit("%INCLUDE 'C:\SAS\SASUSER\procreport.sas';")
</SCRIPT>
```

"**Display**": This script then displays PROC REPORT statements created by the 'procreport' macro in the text area

```
<SCRIPT FOR="Display" EVENT="onClick" LANGUAGE="VBScript">
 cleartext()   'Re-set base string to null
 Dim File
   If (FSO.FileExists("C:\SAS\SASUSER\REPORT_.sas"))=True Then
    Set File = FSO.GetFile(SASPth & "\REPORT_.sas")
    Set TextStream = File.OpenAsTextStream  'Open PROC REPORT text file
    Do While Not TextStream.AtEndOfStream
       S = TextStream.ReadLine & NewLine    'Read each line
       addtext(S)                           'Concatenate to string displayed
    Loop                                    ' in text area, via 'addtext' Jscript
    TextStream.Close                        ' function
    End if
</SCRIPT>
```

"**ColumnSpace**": This script calculates column space used by the report, and displays the value in a message box.

```
<SCRIPT FOR="ColumnSpace" EVENT="onClick" LANGUAGE="VBScript">
 ' First, capture the current text-area contents
   Capture("REPORT_")
 ' Next, capture each column width value, and calculate total
 Dim File, S, SearchChar,i, CharPos, Lng, Rgt, Lft, BetweenColSpace, Total
  SearchChar = "="
  i=0
```

```
    Total=0
    BetweenColSpace=0
    Set File = FSO.GetFile(SasPth & "\REPORT_.txt")
    Set TextStream = File.OpenAsTextStream
      Do While Not TextStream.AtEndOfStream
         S = TextStream.ReadLine & NewLine
         i=i+1
         If i > 2 Then
         CharPos = 0
         CharPos = Instr(1, S, SearchChar, 1)
          If CharPos > 0 Then
           Lng = Len(S)
           Rgt = Right(S,Lng-CharPos)
           Rgt = Trim(Rgt)
           CharPos=Instr(1, Rgt, " ",1)
           Lft = Left(Rgt,CharPos)
           Total=Total+Lft
           BetweenColSpace=BetweenColSpace+1
          End If
         End If
      Loop
      TextStream.Close
      Total=Total+BetweenColSpace-1
      MsgBox("Total Column Space Used= " & Total)
</SCRIPT>
```

"**Preview**": This script submits the PROC REPORT statements displayed in the text area to the SAS session.

```
<SCRIPT FOR="Preview" EVENT="onClick" LANGUAGE="VBScript">
 Capture("REPORT_") 'Write text area contents to 'report_.txt'
 objSAS.Submit("%INCLUDE 'C:\SAS\SASUSER\REPORT_.txt';")
</SCRIPT>
```

The screen-shot above shows the application in use, where a dummy data step has been generated using the lower set of buttons, and PROC REPORT statements have been generated by the '%procreport' macro. The 'column space' event script has been called, to display total column space used.

CONCLUSION

This paper introduced the basic concepts behind running SAS as a 'back-end' to a dynamic HTML application. This introduction, in combination with the working examples provided, will hopefully give SAS programmers with little or no prior applications development experience a good start toward implementing their own applications.

Because of its power and simplicity, the dHTML environment is useful for the rapid development of custom utilities, and for the delivery of SAS functionality to non-SAS-users.

ACKNOWLEDGMENTS

The authors wish to thank first and foremost Jeff Carter of Equinox Software Design (www.equinox.ca) for introducing the possibilities offered by Windows Scripting technologies and for providing substantial technical advice. We would also like to thank Tim Williams for his support of creativity and innovation, and Cara, Kelly, Gavin, Kurtus, PJ, Nicholas, and our friends in Biostatistical Services for their encouragement and support.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the authors at:

Brian Fairfield-Carter, Tracy Sherman, Stephen Hunt
PRA International
600-730 View Street
Victoria, BC, Canada V8W 3Y7
Email: FairfieldCarterBrian@PRAIntl.com; ShermanTracy@PRAIntl.com; HuntStephen@PRAIntl.com
Web: www.prainternational.com