WORKING TOWARD A PAPERLESS OFFICE

WEB MIGRATION OF SAS® REPORTS

Arline Loh, Sheila Hobeck

Scotiabank, Toronto, Canada

Abstract

Decision support information is crucial in the banking environment. The following article focuses on the latest development toward a paperless office for the conservative banking industry. Utilizing SAS® software and Intranet technology, paper reports are now converted to electronic formats for easy access and delivery. Intranets provide both rapid report delivery and the ability to seamlessly convert live reports into spreadsheets for further analysis. The SAS web macros provide the means to construct standardized reports, while maintaining control over the information being delivered and centralizing the data. SAS software integrated with an Intranet renders an efficient means of data delivery.

The first step of web migration is to move paper-based, SAS generated reports onto an Intranet site. Using the SAS web publishing macros, various techniques for formatting, and web integration, reports are seamlessly constructed for data delivery over the Internet or an Intranet.

Introduction

In the current frenzy of web development, the research and development department at the SAS Institute has issued a set of web publishing tools. These macros are the first step in a line of web tools for bringing SAS functionality to Intranets and the Internet, and these macros can be used as either a primary conversion tool, or as the path to dynamic web page creation. These macros, in conjunction with the rest of the SAS macro language, provide a great deal of power to the SAS developer.

Before the availability of the SAS web publishing tools, reports were printed. Even if they were electronically stored, they were never well integrated into HTML. Since HTML is a presentation-oriented language, delivery of data is now both easy and eye pleasing. This article will explore the capabilities of the SAS web publishing macros in the conversion of traditional printed reports. It will discuss all the topics involving SAS web page creation and delivery, and the progression from SAS data sets to the web. It will start with the tools themselves, then progress to the usage and application of the tools, and finally into the fine tuning of the HTML reports to make them as easy to maintain as possible.

Methods

Conditions of development

- For conversion to occur smoothly and to produce the desired web pages, it was required that minimal changes be made to existing programs. Development needed to be quick and easy.
- It was essential that the programs dynamically create the file names for the reports. Due to the number of reports produced (several hundred and growing), the names had to be unique to each report, product and location.

Macros

In many programming languages, the most useful tools available to developers are the macro components of the language. This is absolutely true of the SAS language. By stepping outside of standard code, solutions to baffling problems can be efficient, elegant and time saving.

As a further enhancement to the SAS macro language, the web publishing tools are macros themselves. The current web tools consist of three macros: OUT2HTM, DS2HTM, and TAB2HTM. While all the macros are very

useful, the first two seem to have the most practical application.

Challenges were encountered in producing the reports, which is expected in any development situation. These were overcome easily with the use of macro conditional logic and the web publishing tools. Details of the challenges and the solutions found are covered in the *Discussion* section.

OUT2HTM

The OUT2HTM macro is the most fundamental of the web publishing tools. Taking the output from either the log window or the output window, all captured material is processed and placed in pretext form inside an HTML file. This is an extremely basic function for SAS and a very valuable tool for insertion of non-SAS related information. HTML files created by SAS require no further modification, and thus OUT2HTM is ideal for inserting scripts, tags and links without manual modification.

Report Index, Drill Path and Link Legend

The original printed reports were organized according to report type, and then products within locations (see Figure 1) and locations within products (see Figure 2). Index pages on the web site allow the user to climb down the report tree structure in one of two ways. The first route is to choose a report type and then choose one of the available locations. Each location page consists of all applicable products for the report type for that location. The second route is to choose a report type and then choose one of the available products. Each product page consists of data for each location for that product. On a monthly basis, business requirements may change, and numbers of locations and products can fluctuate. Creation of an index page for all locations and products was incorporated into each of the programs. The code in Example 1 demonstrates how the OUT2HTM macro creates a link legend of products for the produced reports.

Example 1

```
/* Create .html with product filenames as links*/
%OUT2HTM(CAPTURE=ON,
         WINDOW=OUTPUT,
         RUNMODE=B);

DATA _NULL_;
   SET UNIQUE;
   LENGTH N $ 4;
   N = _N_;
   FILE PRINT NOTITLES  LS=174;
   PUT  @1 '<A HREF="QV_PRD'
        @16 _N_
        @(16 + LENGTH(TRIM(LEFT(N))))  '.HTM">'
        @(22 + LENGTH(TRIM(LEFT(N))))  DESC
        @(22 + LENGTH(TRIM(LEFT(N)))
         + LENGTH(DESC))'</a>';
RUN;

%OUT2HTM(HTMLFILE=D:\HTML\PRDNDX.HTM,
         CAPTURE=OFF,
         OPENMODE=REPLACE,
         ENCODE=N,
         WINDOW=OUTPUT,
         RUNMODE=B,
         CTEXT=BLACK);
```

Export HTML Report to Spreadsheet

The static web pages needed functionality that HTML alone could not provide such as editing the page, using the data to draw graphs and charts, and cutting and pasting the data into other applications. Enhancements were made using ActiveX scripting which enabled users to access the HTML data through Excel spreadsheets to provide the functionality listed above. (The Excel software must exist on the user's PC). Details of the ActiveX component will not be discussed here. The ActiveX script is appended to the bottom of the HTML file and is the last stage in the process. A button labelled "View Report in Spreadsheet" appears at the bottom left-hand corner of each page. Note that in SAS, the macro option 'ENCODE' is set to 'NO' to turn off any encoding to HTML. The following sample code (see Example 2) illustrates how SAS generates the code for the HTML-to-spreadsheet capability.

DS2HTM

The DS2HTM macro is quite useful for presenting tables of data. It takes a SAS data set and creates one cell for each piece of data. Included with the macro are the title and footnote variables, which are very useful for quick formatting, without going to another step like the OUT2HTM macro. The programmer is given a lot of control over the table attributes. Although full control is not possible without coding HTML, the degree of control is enough to produce eye-pleasing tables. There are ways to

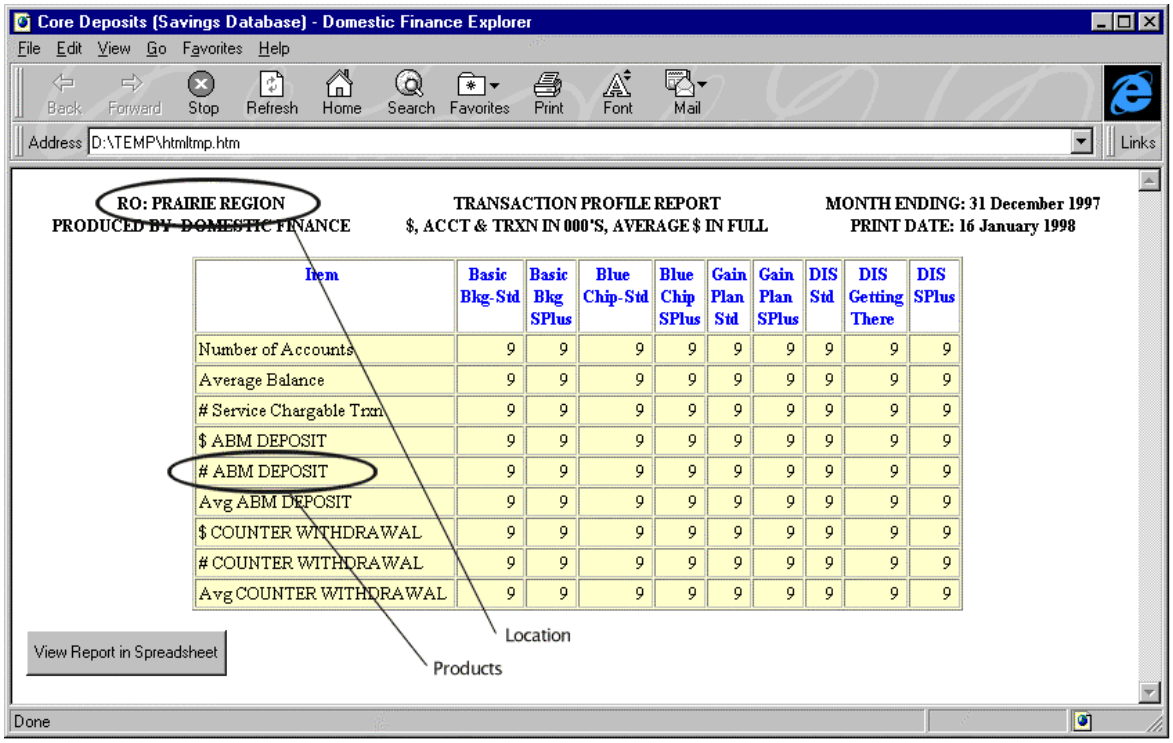

Figure 1 **Products within a Location**

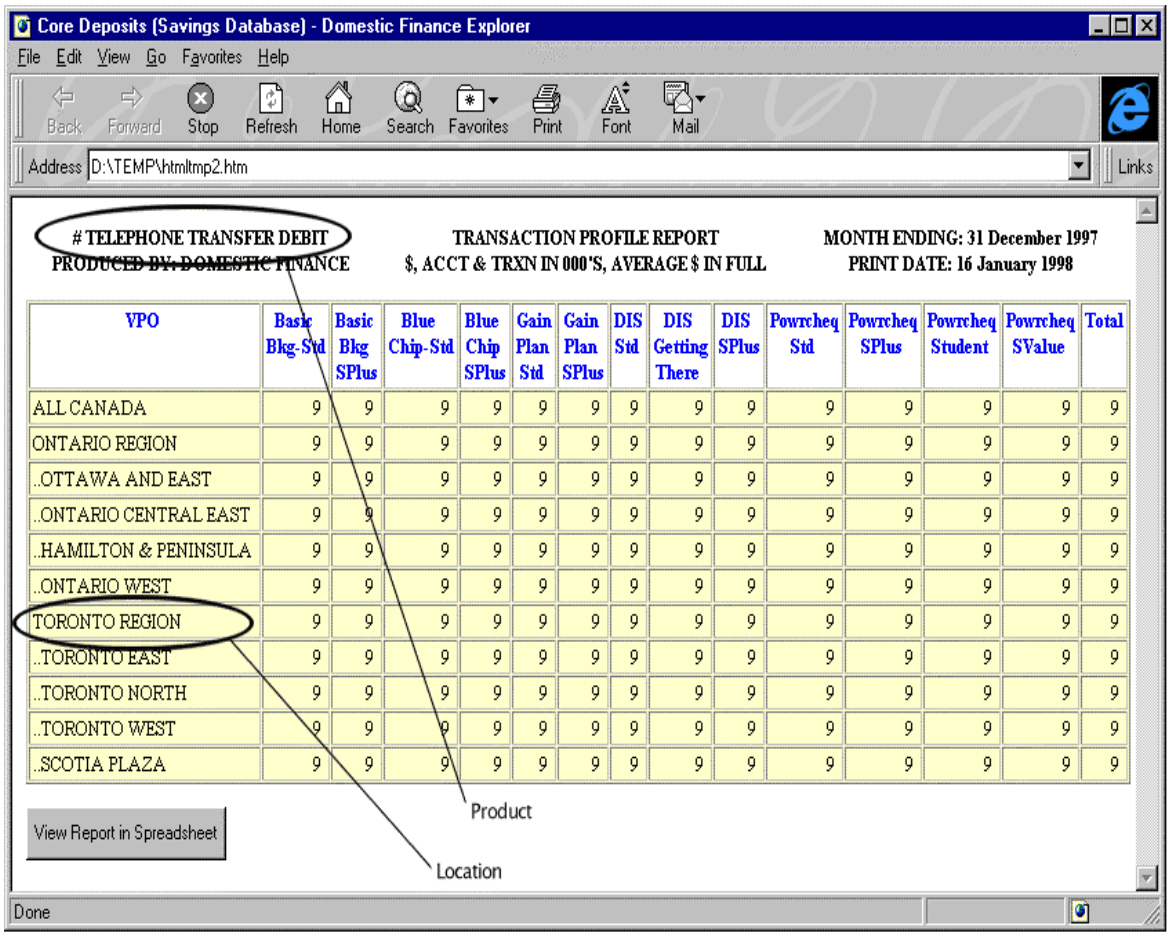

Figure 2 **Locations within a Product**

bridge the gap to full HTML control. For example, character “Alt-0160”, in most Windows true type fonts, appears to the user as a space. However, to the computer, it is a character and can be used to prevent the browser from wrapping the two words directly surrounding it. Similarly, in HTML code, “ ” also creates a space, which can be useful in some programming situations. Example 3 illustrates a typical usage of the DS2HTM macro to create a simple table, with variables as columns of the table.

Example 2

```
/*Create ActiveX link as file component*/

%OUT2HTM(CAPTURE=ON,
         WINDOW=OUTPUT,
         RUNMODE=B);
DATA _NULL_;
   FILE PRINT NOTITLES LS=174;
   PUT '<SCRIPT LANGUAGE="JavaScript">';
   PUT '<!--//SCRIPT STARTS HERE';
   PUT 'VAR TEMP;';
   PUT 'DOCUMENT.WRITE(''<P><OBJECT
        ID="HTMLSpread"'');';
   PUT 'DOCUMENT.WRITE(''classid="clsid:E072405B-
        E8C3-11D0-B2ED-00A0241CE01D"'');';
   PUT 'TEMP=
       "codebase=\"http://www.domfin.scotiabank.ca/
        ActiveX/STCHTMLToSpread";';
   PUT 'TEMP = TEMP +
       "/HTMLToSpread.CAB#version=0,1,0,1\"
       align=\"baseline\"";';
   PUT 'DOCUMENT.WRITE(TEMP);';
   PUT 'DOCUMENT.WRITE(''BORDER="0"
        WIDTH="145" HEIGHT="33">'');';
   PUT '//SCRIPT ENDS HERE -->';
   PUT '</SCRIPT>';
   RUN;
%OUT2HTM(HTMLFILE= MYHTML.HTM,
         CAPTURE=OFF,
         OPENMODE=APPEND,
         ENCODE=N,
         WINDOW=OUTPUT,
         RUNMODE=B,
         CTEXT=BLACK);
```

Example 3

```
/* Typical use of the ds2htm macro */

%ds2htm(htmlfile= MYHTML.HTM,
        openmode=append,
        data=ki,
        where=,
        obsnum=,
        labels=y,
        formats=y,
        round=0,
        id=,
        var=OS PPORACT QTRVAR QTRVARPT
            CATEGORY YTDGR YTDVAR
            YTDVARPT WHYRPLAN WHYRPT
            YROVYR YRCHG,
        sum=,
        encode=y,
        runmode=b,
        bgtype=,
        bg=,
        brtitle=QV Report,
        center=n,
        ctext=);
```

TAB2HTM

The TAB2HTM macro is closely related to the TABULATE procedure in SAS. By replacing or modifying the TAB2HTM macro, and using the same organizational conditions, an advanced table can be created in HTML. Unfortunately none of the reports in our situation used the TABULATE procedure, and thus presented no opportunity to uncover the finer points of this last macro.

Discussion

Most of the reports used a large sorted data set to create the printed reports. However, using this data set to create tabulated data through the DS2HTM macro would simply output the whole data set to an HTML file, making it large and unmanageable. DS2HTM does not have any capability to modify or subset the data set. To pass the records for one report page to the macro, a data set containing the data for that page must be created. To do this efficiently without repetitious code, macros must be employed to dynamically create code that will produce the web pages.

By creating a data set containing the filtering criteria for each report, and in conjunction with macros, individual data sets can be created for report processing. By passing parameters to macro variables, output information can be dynamically generated.

Consistent appearance was vital to the acceptance of the reports by users. A great deal of effort was put into creating identically styled reports. The colours used in displaying tables and text, fonts, relative (minimum) size, titles and labels for the tables, and formats of different types of data are identical in each report.

Titles

For consistency, identically styled titles for the reports were produced through ‘title’ data sets. Alternatively, titles from the SAS data sets can be output as part of the data with the DS2HTM macro (see DS2HTM documentation for details on the formatting of titles). Titles were stored in

a separate data set and output as a table with three columns. Example 4 below shows partial code for producing the titles for the reports.

Example 4

```
/**loop through all locations**/
%do I=1 %to &nregn;

   /**select obs from one location only**/
   data reptpage;
      set rept&reptnum;
      where region eq "&&regn&I";

      %if &reptnum=109 or
          &reptnum=110 or
          &reptnum=112 or
          &reptnum=113 %then %do;
          label ytdgr = 'YTD ACTUAL';
      %end;

      unit=put(&reptnum, curunit.);
      call symput('unit', unit);
      reptfor=put(region, $rgn.);
      call symput('reptfor', reptfor);
      reptname=put(&reptnum, rname.);
      call symput('reptname', reptname);
   run;

   data title;
      length column1 column2 $ 55 column3 $ 55;
      tday=put(date(),worddate20.);
      mday=put(&asof,worddate20.);
      column1 = "REPORTED FOR:  &reptfor";
      column2 = "&reptname";
      column3 = "MONTH ENDING:  "||mday;
      output;
      column1 = "PRODUCED BY:  DOMESTIC FINANCE";
      column2 = "&unit";
      column3 = "PRINT DATE: "||tday;
      output;
      label
         column1 = ' '
         column2 = "MONTHLY BRANCH REPORTING SYSTEM"
         column3 = ' ';
      run;
      title1;
%end;
.
.
.

/*format data set title into html table*/
%ds2htm(htmlfile = %if &member='REGION'
    %then %do;
       "d:\html\&reptnum\mbr&reptnum&&regn&i...htm",
    %end;
    %else %if &member='PRODUCT' %then %do;
       "d:\html\&reptnum\mbr&reptnum&&prdt&i...htm",
    %end;
    openmode = replace,   /* create new .htm file */
    data = title,
    where = ,
    obsnum = ,
    labels = y,
    formats = y,
    round = 0,
    encode = y,
    runmode = b,
    center = y,
    border = n,
    cpad = 0,
    bgtype = color,
    bg = white,

    /* this is the title in the browser window */
    brtitle = Monthly Branch Report,

    septype = none,
    twidth=800,
    twunits=pixels,
    id = column1,
    ihalign = center,
    iwrap = no,
    isize = 2,
    itag = bold,
    var = column2 column3,
    clsize = 2,
    clwrap = n,
    vhalign = center,
    vsize = 2,
    vwrap = n,
    vtag = bold,
    ctext=);
```

Data Tables

The data tables were relatively easy to produce using the DS2HTM macro. Similar to the title data set, the data tables contain labels, colours, backgrounds and borders.

The data table is appended to the file already containing the title table. The subset data is passed to the DS2HTM macro that matches the table to its title. Example 5 shows the code for the data table portion of the report.

Example 5

```
/*format data set reptpage into html table*/
%ds2htm(htmlfile = %if &member='REGION'
    %then %do;
       "d:\html\&reptnum\mbr&reptnum&&regn&i...htm",
    %end;
    %else %if &member='PRODUCT'
    %then %do;
       "d:\html\&reptnum\mbr&reptnum&&prdt&i...htm",
    %end;
    openmode = append,  /*add to existing .htm file*/
    data = reptpage,
    where = ,
    obsnum = ,
    labels = y,
    formats = y,
    round = 0,
    var = %if &member='REGION' %then %do;
             octbase prevmo movar_2 lastmo movar_1
             thismo movar mo1 desc ytdgr ytdvar
             whyrplan yrchg yrovyr,
          %end;
          %else %IF &member='PRODUCT' %then %do;
             octbase prevmo movar_2 lastmo movar_1
             thismo movar mo1 region ytdgr ytdvar
             whyrplan yrchg yrovyr,
          %end;
    encode=y,
    runmode=b,
```

```
    center=y,
    septype=none,
    border=y,
    cpad=2,
    tbbgcolr=#ffffc0,
    twidth=800,
    twunits=PIXELS,
    clbgcolr=white,
    clcolor=blue,
    clsize=2,
    clwrap=,
    clvalign=TOP,
    vbgcolr=#ffffc0,
    vcolor=,
    vhalign=center,
    vvalign=middle,
    vsize=2,
    vwrap=n
    ctext=);
```

Export to Spreadsheet

Once titles and data tables for the reports have been output, the last remaining piece of the report is the ActiveX component. Briefly, the ActiveX scripting allows a user who is viewing the reports with a browser, to pull the data off the web report page and into an Excel spreadsheet. The only requirement for this function to work is that Microsoft Excel ® be installed on the client's PC. This capability saves users from having to re-enter the data, which often leads to errors. Time spent entering and correcting data is now spent analysing and making active decisions.

General Report Presentation

Report Layout

The original programs were written with a paper report in mind. It was visually appealing, with columns lined up, separator lines and spaces and other such niceties. These were eliminated in the HTML format, since applying them to tabulated data in HTML would detract from the visual layout. Early in the development of the web reports, the OUT2HTM macro was used in conjunction with the customized page layouts from the original programs. Unfortunately, it was difficult to produce the same level of visual quality as the paper reports for the browser. Output on the screen has different requirements than that in paper form. Size, spacing and alignment all proved to be difficult to maintain. For these reasons, DS2HTM was chosen over OUT2HTM as a means of displaying the data.

Column Labels

Column labels are an insufferable problem. It is probably easiest to use the default column labels. However, for cosmetic purposes, the default is not always best. Depending on the report, labels may need to be changed, or in the case of display columns, assigned. A good way to do this is through the label statement. However, only one of these statements can be issued during a data step. This is where macros are at their strongest. Using macro conditional logic, it is possible to issue more than one label statement per data step. Example 6 shows the code for conditionally assigning data set labels.

Example 6

```
/* conditionally select labels for tables */
%if &rept =109 %then %do;
    label OS='ACTUAL $'
          PPORACT='PLAN $'
          QTRVAR='VARIANCE $'
          QTRVARPT='TO PLAN %'
          YTDGR='ACTUAL $'
          YTDVAR='VARIANCE $'
          YTDVARPT='TO PLAN %'
          WHYRPLAN='ACHIEVED/ EXPENDED $'
          WHYRPT='W/Y PLAN %'
          YROVYR='YR/YR $'
          YRCHG='YR/YR %';
%end;
%else %do;
    label OS='PORTFOLIO $'
          PPORACT='GROWTH $'
          QTRVAR='VARIANCE $'
          QTRVARPT='TO PLAN %'
          YTDGR='GROWTH $'
          YTDVAR='VARIANCE $'
          YTDVARPT='TO PLAN %'
          WHYRPLAN='W/Y PLAN $'
          WHYRPT='ACHIEVED %'
          YROVYR='YR/YR $'
          YRCHG='YR/YR %';
       %end;
```

Index Pages

An index page with hyperlinks to all the freshly generated HTML reports is created before the program ends. Since the web pages and frames on the Intranet web site were customized for consistency and aesthetic purposes, manual editing is required before the index page is presentable. Allowing the program to generate index pages reduces the amount of manual intervention required with the index pages, and saves time for the maintenance programmer. The current program brackets each link with

<DD> HTML tags so that the page of links is properly indented. Example 7 illustrates part of the index page for locations for the Monthly Branch Web Reports.

Example 7

```
/**texts and hyperlinks for index page**/
data index;
   set region;
   length text $ 40  link $ 14;
   text = put(region, $rgn.);
   link = "mbr&reptnum."||region||".htm";
   output;
run;

/**output file for index page**/
%out2htm(capture=on,
         window=output,
         runmode=b);

data _null_;
   set index;
   file print notitles ls=174;
   put '<dd><a href="'
       link
       '">'
       text
       '</a>'
   ;
run;
%out2htm(htmlfile="d:\html\&reptnum\
         Regndx.htm",
   capture=off,
   window=output,
   encode=n,
   openmode=replace,
   runmode=b,
   bgtype=color,
   bg=white,
   brtitle="MBR Region Index",
   septype=none);
```

The Algorithm for Automatic HTML Report Generation

Since there are several different groups of reports to be generated from any one data set, the data is read through many times until all data is exhausted. The original algorithm presented a problem since the double nested structure of the program drained an incredible amount of memory, thus crashing the active SAS session as well.

The final solution was to split the data into several smaller data sets, grouped by the type of report. The large complete data set is deleted. Each subset of data is sent to a macro where an additional subset data set is chosen to produce a specific report. Although this process is cumbersome, a macro was written to perform this as a generic process and it was the only foreseeable way to prevent system crashes.

Memory usage is still a problem with this algorithm. However, since the input data set is divided into smaller ones before the macro is invoked, multiple SAS sessions on several Pentium PCs are used to deal with several reports at one time.

Naming Files

Correlation between HTML file names and the contents is suggested for maintenance purposes, although it is not relevant what naming convention is used for the HTML reports. Loop counters, numeric table descriptions and report numbers are used to produce file names. By creating a format table, destinations of files can be associated with specific report traits and variables and, by using macro processing, reports can be saved in specific locations. Less human intervention during maintenance is always appreciated.

Data accessibility

One of the primary concerns of HTML data delivery is the accessibility of the data. Tabulated data in HTML format cannot be manipulated directly from the browser. This lack of control creates the need for a decision on an appropriate tool for managing the data. Listed below are a few options currently provided to the user and a few offered as future enhancements.

Viewing Only

There is the option of doing nothing. The user can simply view the documents and take notes on the available data as it is presented on the screen. This single type of data delivery should be used for messages to users only. Reports and other documents should offer more accessibility.

Users are able to cut and paste the data directly from the browser to a spreadsheet program, and then parse the data into a usable form. This is an extremely tedious task and since data copied from an HTML table gets pasted as one long string into one cell of a spreadsheet, it is not very useful. Users could save or export the HTML file as a text file, and then import it into a spreadsheet, but this still requires the user to do further editing.

Printing

Most web browsers are presentation tools, and only perform at a mediocre level when faced with other tasks, like printing. Having a large table that extends past the boundaries of the destination paper, and trying to print it, brings undesirable results. By adjusting font sizes and resolutions, a table can be made to fit nicely. Unfortunately, HTML scripts do not have the ability to change client browser settings and configurations and so it is left to the user to find the best fit for the printed object.

Importing

Another option is to automatically import the HTML file into a spreadsheet program, so that the data can be handled at whim. All that is required is an ActiveX component or a Java Script to execute the spreadsheet program while passing the current HTML location. Excel lives up to its name in terms of importing an HTML file and keeping all the graphical attributes. In testing and development, Lotus 1-2-3 Suite® Version 7.0 was not able to import HTML pages easily with all the applicable table attributes attached.

Transferring Files

A future option for the users could be a button or link which will automatically download a spreadsheet file to the user's computer, and allow the manipulation of the file by the user. Specific spreadsheet files are a lot smaller than their HTML cousins are. The drawback to this method is that two copies of every report must exist on the server. SAS is also capable of producing the spreadsheets through the DBLOAD procedure and thus would be easy to implement into our existing processes.

Delivery Channel

The Intranet resides on an IBM 350, Pentium 200MHz, with 164 MB of RAM, with approximately 7.5 Gigabytes of hard drive space. In addition to the production server, there is a UAT (user acceptance testing) server as well. The UAT is also an IBM 350, Pentium 133MHz, but only has 32 MB of RAM, with approximately 5 Gigabytes of hard drive space. The extra memory is needed on the production server to assist in running the telecommunication software for the modem users dialing into the server. The web server itself is the Internet Information Server from Microsoft, Version 3.0. IIS software was chosen over the Netscape server primarily due to the cost of the license. The next step in terms of development of the Intranet is to migrate to a true Windows NT® server, (as opposed to the Windows NT® workstation), and then on to a Unix platform, within the next two years.

Both the UAT and production servers are accessible to internal users through a local area network. Users across the country dial in over modems. Security issues will not be discussed here, but it is noted that the Intranet is a closed domain and poses little security risk to the bank. Reports undergo an approval process on the UAT server. Once sign-off is obtained, the reports are then moved to the production server for full viewing.

The PC SAS programs were developed on an IBM 365, Pentium 75MHz with 16 MB of RAM running SAS 6.12 under Windows95. Mainframe programs to create original data were run in MVS/TSO. Several programmers with varying levels of expertise in SAS programming developed the programs over a six-month period.

Conclusion

Currently there are four main types of reports created by the OUT2HTM and DS2HTM macros. The process produces 600 – 800 HTML files. Each contains an ActiveX component for spreadsheet data access. Their respective programs create product and location indexes.

The functionality gained from the implementation of the SAS web tools far outweighs the effort applied. The rewards are numerous and include mass availability, reduced printing and reduced ad hoc requests. This project was well received by management and end users.

Future Considerations

It is recommended that this project be taken to the next level by extending this pull technology to include user interaction and dynamic generation of HTML reports. There are many ways in which to achieve this, and many products to explore. It is hoped that dynamic push technology can eventually be used to send the user his/her applicable reports. New state-of the-art push technology is in its infancy and

would definitely save search time for our internal users. In-depth analysis will determine the best course of action to take.

Acknowledgements

Candy Wong and **Larry Leung**, Co-op students from the University of Waterloo, both of whom spent many grueling hours on the PC developing and perfecting the programs to the user specifications and did a wonderful job.

Genia Vanderkruk, Senior Programmer Analyst, Domestic Finance, Scotiabank. Her technical advice and report editing talents are much appreciated.

John Gilbert, Nomina Communications Inc., SAS Consultant. We wish to express our thanks for his support and suggestions.

Colleen Johnston, Vice President and Domestic Comptroller, Domestic Finance, Scotiabank. We wish to extend our thanks and appreciation for her vision and ongoing support of our technical team.

References

SAS® Institute Web Tools documentation (delivered with the web macros off the Internet).

"SAS® Guide to Macro Processing", Version 6, Second Edition, 1990, SAS Institute Inc., Cary NC, USA.

M. Afergan, R. Darnell, B. Farrar, R. Jacobs, D, Medinets, R. Mullen, and M. Foghlu, "The Web Programming Desktop Reference", 1996, Que® Corporation, ISBN 0-7897-1028-5.

Contacts

For comments, suggestions or copies of code, please contact:

Arline Loh
Senior Technical Manager
dloh@sprint.ca
or
Sheila Hobeck
Programmer Analyst
hobeck@interlog.com

Domestic Finance
Scotiabank
100 Yonge St., Suite 505
Toronto, Ontario
M5C 2W1