Web Enabled Government Billing Using SAS/IntrNet™ Software

Tom Ivons, Computer Sciences Corporation, Long Beach, CA

ABSTRACT

Our client's Defense Finance and Accounting Service (DFAS) Public Voucher Electronic Data Interchange (EDI) billing process could not support all contracts due to limitations inherent in the enterprise accounts receivable process. Prior to the Web implementation, only those contracts not requiring detailed accounting information could be sent electronically. Since there is a substantial cycle time reduction in the payment process for electronic invoices, it was imperative that all invoices be sent electronically. We were asked to create a Web interface that would allow accounting information to be added to invoices after they had been generated by the legacy billing system. We chose SAS/IntrNet as our implementation tool because it runs on all our enterprise platforms and allowed us to wrap a Web front end around our existing SAS® software based electronic invoicing process. This paper addresses the challenges we faced during development and implementation. These include porting a complex SAS application from the OS/390 mainframe to the Windows NT server, generating HTML forms on the fly using SAS, maintaining state in a web based transaction processing environment, and implementing system and application security using SAS.

INTRODUCTION

This paper provides an overview of our experience creating an interactive Web application using SAS/IntrNet middleware. It also comments on issues that arose as we ported a complex SAS application from the IBM OS/390™ mainframe to a Windows NT™ Web server. It is less about building invoices and more about the process of creating and integrating a Web application using SAS software tools. This paper will be most useful to people who are familiar with SAS/Base, HTML, Internet Utilities, Windows Utilities and the OS/390 mainframe. However, it is also intended to provide encouragement to everyone that this kind of application can be implemented using SAS/IntrNet software along with commonly available tools.

PROJECT OBJECTIVES

Our objective was to reuse as much of the existing electronic invoicing process as possible. This included the SAS code for applying DFAS business rules to invoices as well as SAS code that prepares invoices for EDI processing. Reuse was critical because a substantial investment had been made to implement the current process and more importantly our current process was DFAS certified.

SAS worked well for us in achieving these goals because we were able to port our invoice analysis code from the OS/390 mainframe to a Windows NT server with minimal changes. The code had to be modified slightly as described below but these modifications were propagated back to the mainframe so our analysis software remains a single image across platforms.

THE INVOICING PROCESS

Accounts receivable employees create invoices using the enterprise accounts receivable application (ARS). An option within the ARS allows invoices to be sent to the DFAS Web application. Employees use this option when government contracts require detailed accounting information that is not captured by ARS.

Employees log into the Web server and are presented with the DFAS invoicing home page. They get the option to create manual invoices or get invoices from ARS. During invoice creation or update, employees are given the option to add accounting information to the invoice.

Input is verified using JavaScript to the extent possible. Global edits are performed at the end of the process. Global errors are presented to the user in the same format as provided by the batch process. The current working set of invoices is stored in SAS datasets and can be viewed by the user in the form of on-line SAS reports passed through the SAS/IntrNet software to the browser.

Only error-free invoices can be sent to DFAS. When all errors are corrected using the update pages, the user has the option of sending the invoice.

When the send button is selected, SAS programs are executed which build the actual FTP commands and the various files that are required to complete the process. Other SAS programs review the FTP log to insure successful completion. At the end of the process, the user receives a confirmation of success or a message to cut a trouble ticket. New invoices cannot be processed until any FTP problems are resolved.

Sent invoices are passed to the OS/390 mainframe for EDI translation and immediate transmission to the value added network that interconnects with the US Government FACNET.

ARCHITECTURE

The scope of this application includes the legacy accounts receivable application; the new Web input screens and associated back-end processing, the existing invoice analysis process, and the existing EDI process. The application spans three platforms that need to be tied together in near real-time. Our interfaces are simple; we pass flat files using FTP. We also use FTP for inter-platform communication. Invoices are sent via FTP to a dataset on the OS/390 mainframe. The same FTP process sends a job to the OS/390 that tells the production job scheduler to run the EDI translate and send process. The FTP messages are written to a file that is processed by another SAS program to determine if the FTP was successful. This results in an automated process that executes in near real-time.

SAS code is used to generate all the HTML with the exception of static message pages. Users have individual directories on the web server and all their work is saved in SAS datasets as they move from page to page. This provides separation between users, allows the preservation of user's work between sessions, and maintains state during the session as the user moves from page to page.

SECURITY

This application runs on a corporate intranet behind a firewall so internet security was not an issue. However, invoicing is a sensitive process and needs to be restricted to authorized employees only. To implement this level of control, we used a combination of server and application security. Server security was implemented using Netscape web server controls. Application security was implemented using SAS.

Once the user is authenticated to the web server, our SAS programs have access to the userid through the SAS/IntrNet

middle-ware. Our application validates the userid using a SAS data set containing information about users including the invoice types to which they have access, and the state in which they last left the application. In this way we know if their last session was interrupted, or if they have an outstanding ftp problem that must be resolved before they can continue.

We also use the SAS ability to execute system commands to create an LDAP password reset function which we describe in more detail below.

CODE EXAMPLES

A few code examples will illustrate our methods. SAS/IntrNet uses the filename _webout to designate code you want sent back to the user's browser. This allows you to create complex HTML pages using PUT statements and other SAS programming constructs.

GENERATING SIMPLE HTML

This is code to produce a form element, which is loaded initially with data passed from the legacy accounts receivable process.

```
put '<code><input type="TEXT" name="cname"
size="35" maxlength="35"
value="'cname +(-1)'"></code>';
```

This is standard HTML with a PUT statement wrapped around it. The value cname is a sas data element name that was read in from SAS datasets in the user's directory. It is also the HTML form data element name. The +(-1) gets rid of an ugly looking space that would otherwise appear on the form after the data.

CREATING PULL-DOWN LISTS

You can use SAS programming structures to produce pull down lists. In this example we are creating a pull-down list of valid CAGE codes.

First get the data and load it into macro variables.

```
data _null_;
     infile cage;
     input @01 cage $05.;
call symput
('cage'||left(put(_n_,3.)),put(cage,$05.));
call symput('cnum', put(_n_,3.));

run;
```

Then create a macro that will be executed to create the pull-down list. Notice that we select the first CAGE code by default.

```
%macro cage;
      %do i=1 %to &cnum;
     %if i=1 %then
            put "<option value=""&&cage&i""
selected>&&cage&i";
     %else
            put "<option
       value=""&&cage&i"">&&cage&i";
     %end;
%mend cage;
```

The pull-down list is presented whenever CAGE code is blank.

```
/*************************************/
/*if cage code is not passed present valid list*/
/*************************************/

if cage ne '' then do;
put '<td> ';
put '<code><input type="TEXT" name="cage"
size="5" maxlength="5"
value="' cage +(-1)'"></code>';
put '</td>';
end;
else do;
put '<td><select name="cage">';
%cage;
put '</select></td>';
end;
```

WORKING WITH FTP

FTP is a powerful utility and has the advantage of being available at little or no cost on all the platforms we had to integrate. We used SAS code to build FTP statements and batch files that allow us to move data from the NT server to the OS/390 mainframe and monitor the process as it happens.

This code builds the FTP commands from a parameter file that is maintained for each accounts receivable process (ARS) that is supported by the Web application. Each ARS has its own production EDI job stream on the mainframe to make it easier to charge-back and monitor the process.

The NT FTP statement 'Literal' lets you pass commands like 'Site' that are not valid on NT but are meaningful to the FTP daemon under OS/390.

The second FTP put command sends a file called JCL.txt to the OS/390 job entry system (JES) reader for direct submission.

&_rmtuser is a macro variable containing the userid. It is created and passed to your program by SAS/IntrNet.

```
/*************************************/
/*         setup ftp commands        */
/*************************************/

data _null_;
set arslib.info;
      file ftpin;

put @01 ars_user;
put @01 ars_pw;

put 'literal';
put 'site LRECL=250 BLKSIZE=27750 TRACKS';

put "put
c:\dfas\voucher\users\&_rmtuser\adf.dat"
' ''EDI2.'filename+(-1)'.WEBIN(+1)''';

put 'literal';
put "site FILEtype=JES";

put "put
c:\dfas\voucher\users\&_rmtuser\jcl.txt";

put 'quit';
run;
```

This is how you build the jcl.txt file using the ARS parameters. This one step job triggers the production job scheduler.

```
/*************************************/
/*     setup jcl commands            */
/*************************************/
data _null_;
set arslib.info;
```

```
file jcl;

put '//'jobname' JOB (2,E04,'pdpa+(-
1)',00,42,VAC6178),''M104,IVONS'',';

put '//     CLASS=A,MSGCLASS=X,MSGLEVEL=(2,0)';

put '/*ROUTE PRINT U10';

put '//STEP01    EXEC
U7SVC,PARM=''D=EDI1.A56419GB.TRIGGER.G0000V00''';

put '/*';
RUN;
```

You pass the actual FTP command to NT using the SAS 'SYSTEM' command. Host is the DNS name of the OS/390 mainframe loaded from the info SAS dataset.

```
/*************************************/
/*    SETUP FTP SYSTEM COMMAND LINE    */
/*************************************/
data _null_;
     set arslib.info;
file ftpbat;

put "ftp -s:c:\dfas\voucher\users\&_rmtuser\
ftpin.txt " host
">c:\dfas\voucher\users\&_rmtuser\
ftpmsg.txt";

put 'exit';
run;

data _null_;
CALL
SYSTEM("C:\dfas\voucher\users\&_rmtuser\FTPBAT.BA
T");
run;
```

We check the FTP message file ftpmsg.txt to see if both put's processed successfully.

```
/************************************/
/* ftp return codes 3 250's means success*/
/************************************/
data _null_;
infile ftpmsg end=last;
retain count 0;
input ret_code $3.;
if ret_code = '250' then count=count+1;
if last then
call symput('count',count);
run;
```

DISPLAYING SAS REPORTS

SAS/IntrNet has a formatter that converts SAS output to HTML. It is quite useful for formatting simple data presentations. You run the out2htm macro which captures your output and writes it to a file for later display.

The following code tells SAS/IntrNet to start data capture. You have to pass the browser the content type so it does not get confused.

```
DATA _NULL_;

     FILE _webout;

     put 'Content-type: text/html';
     put ;

%out2htm(capture=on,
          window=output,
          runmode=b);
```

Once you have produced all the output you want to display you terminate capture and indicate an output filename.

```
%out2htm(htmlfile=c:\dfas\voucher\users\&_
rmtuser\viewit.htm,
          capture=OFF,
          window=output,
          openmode=replace,
          runmode=b,
          tcolor=red,
          fcolor=green,
          hcolor=blue);
```

This code spools the reports to the browser.

```
filename viewit
"c:\dfas\voucher\users\&_rmtuser\viewit.ht
m";

DATA _NULL_;
     FILE _webout;
     infile viewit end=last;
     input;
     put _infile_;
run;
```

WEB SERVER PASSWORD RESET

The following process is the server-side part of the password reset. &pw1 is the CGI variable entered into the password reset form.

Allocate the files you need to support password change. Notice each user has an individual directory on the application server. &_rmtuser is the macro variable to which SAS/Intrnet assigns the authenticated userid.

```
filename ldapcmds
"d:\dfas\voucher\users\&_rmtuser\ldap.txt"
;
filename changepw
"d:\dfas\voucher\users\&_rmtuser\changepw.
bat";
filename erasepw
"d:\dfas\voucher\users\&_rmtuser\erasepw.b
at";
```

You need to store a command file containing the LDAP commands you will use to actually change the password.

```
/************************************/
/*                                  */
/*          SETUP LDAP COMMANDS     */
/*                                  */
/************************************/

data _null_;

      file ldapcmds;

       put "dn: uid=&_rmtuser";
      put "changetype: modify";
      put "replace: userpassword";
      put "userpassword: &pw1";

run;
```

Now you have to create the batch file that contains the LDAP system command.

```
/*************************************/
/*                                   */
/*     SETUP LDAP SYSTEM COMMAND LINE  */
/*                                   */
/*************************************/


data _null_;

file changepw;

put
"d:\netscape\suitespot\userdb\ldap\tools\ldapmodi
fy -C
d:\netscape\suitespot\userdb\ldap\config\lcache.c
onf -f
d:\dfas\voucher\users\&_rmtuser\ldap.txt";
put 'exit';

run
```

This process will leave the ldap.txt file on the server with the user's password. To avoid compromising the password, we delete the ldap.txt file. The code below creates the batch file that accomplishes this.

```
/*****************************************/
/*                                       */
/*    SETUP ERASE COMMAND TO DELETE PASSWORD FROM
USER'S DIRECTORY    */
/*                                       */
/*****************************************/

data _null_;

file erasepw;

put "erase
d:\dfas\voucher\users\&_rmtuser\ldap.txt";
put 'exit';


run;
```

This last bit of code uses the SAS SYSTEM command to actually execute the batch files we just created. Since the batch files end with the EXIT command control is returned to SAS after the batch files execute.

```
/*****************************************/
/*                                       */
/*    EXECUTE PASSWORD CHANGE AND DELETE   */
/*                                       */
/*****************************************/

data _null_;

CALL
SYSTEM("d:\dfas\voucher\users\&_rmtuser\changepw.
bat");
CALL
SYSTEM("d:\dfas\voucher\users\&_rmtuser\erasepw.b
at");

run;
```

IMPLEMENTATION NOTES

Our original architecture was to build standard HTML form pages and pass CGI variables to a SAS back-end through the application broker. We learned early on to build our pages dynamically using PUT statements. This allowed us to use the power of SAS to produce context sensitive pages and dynamic pull-down lists with ease.

We needed to interface with the legacy accounts receivable system and did so through the same flat file structure we use for the existing electronic invoicing process. This minimized changes to our legacy process. Flat file interfaces are simple and are supported on all platforms.

We learned that there are important differences between the IBM mainframe world and the variable length record world of our servers. When we first started reading our interface files on the server, SAS was running over to the next record whenever we had spaces in optional fields at the end of a record. This caused havoc and made us think that we had a major compatibility problem. Our only problem turned out to be the line feed which we moved to its proper position by hard coding an asterisk at the end of the record.

We were reminded about the differences between ASCII and EBCDIC sort sequences when our invoice analysis process failed. Always use numbers for internal control and you can't go wrong across platforms.

We save all the user's work in process in SAS datasets that are kept in directories identified with the person's userid. This preserves state and will survive a bad network stability day.

We resolved the problem of serializing multiple invoice flows from the Web server to the mainframe by setting up the FTP receiving file as an OS/390 generation data group (GDG). The file acts as an input stack. New files are inserted at the bottom. The EDI job processes the files last in first out. As a file in the stack is processed it is archived and deleted from the stack. At any given time, there are as many EDI jobs waiting as there are generations of the FTP file. The job scheduler always triggers the same job name which causes the OS/390 job entry system to single thread the process until all the EDI jobs in the queue have been run.

CONCLUSION

SAS/IntrNet works as advertised. It allowed us to quickly turn around a complex mission critical application. Because we implemented using SAS software, we can move this application from its current Windows NT home all the way to our OS/390 mainframe without code changes. Our client is happy, we are pleased with our achievement, and we look forward to implementing this process throughout our aerospace client's divisions.

CONTACT INFORMATION

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

Tom Ivons
Computer Sciences Corporation
LB/A01/C457
1520 Hughes Way
Long Beach, CA 90810
Work Phone: 310.513.5749
Fax: 310.816.6095
Email: civons@csc.com

SAS and SAS/IntrNet are registered trademarks or trademarks of SAS Institute Inc. in the USA and other

countries. IBM and OS/390 are registered trademarks of trademarks of International Business Machines Corporation. ® indicates USA registration.

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