Paper 021-31

Batch Job to Create and Send Mass Excel Reports Using REXX, SAS, HTML(XML) and Email System

Stanislaw Furdal, Bristol Myers Squibb Co., Plainsboro, NJ

ABSTRACT

The paper presents a computing architecture and related problems of a batch job to create and send mass Excel reports to a lot of users using REXX, SAS Base, HTML, MSO XML and Email system. The paper addresses an application of a SAS parameterized program invoked multiple times from REXX to create Excel reports and send them by Email. It reviews, based on a real business example, a proposed computing architecture, programmable components written in REXX, SAS, HTML(XML), stresses their effectiveness, presents optimization problems and solutions to save CPU time and time of HTML to Excel conversion.

INTRODUCTION

Sometimes we want to send one kind of a report to a lot of users. For example a sales department wants to send emails with a territorial daily sales report to territorial business managers or a customer services department wants to send emails with an attached order status report to all its customers. We deal here with similar kind (form) of reports; only data sent to different email recipients are different. Such a task can be done automatically by invoking multiple times the same reporting SAS program during execution of one computer batch job. Of course the series of multiple SAS executions raises some optimization problems because it has to be done in some reasonable, available time period. There are already at least two known approaches to do that, one is to use a wrapper SAS program (Andrews, Cogswell 2005) which can invoke multiple times the reporting SAS program and the other approach is to use some script program (higher level program) to invoke the reporting SAS program multiple times.

This paper shows how we can create the Excel reports and send mass emails using a top program written in REXX (Fosdick 2005). The REXX program will invoke multiple times the SAS program to create an Excel (HTML) report for each user and Email system will send the emails with the attached reports. All will be presented based on a particular business task to send daily sales reports to company's territorial business managers. The paper discuses also optimization problems and solutions.

BUSINESS TASK EXAMPLE

A company has hundreds of territorial sales representatives and every day a specific territorial sales report showing product sales at a customer level has to be sent to each sales representative. Each report has to be sent via Email in an attached Excel formatted file.

Sales quantities, territorial IDs and email addresses are stored in a huge invoice database together with other data. To avoid filtering multiple times that huge database, the sales, IDs and addresses should be extracted first and kept in two separate entities. So the sales for all territories will be extracted first and kept in a SAS data library, the territorial IDs and email addresses will be extracted and kept in a separate external file.

PROPOSED COMPUTING ARCHITECTURE

1. Operation System with installed REXX interpreter
2. REXX program, to invoke a reporting SAS program and Email system to send the emails with the attached reports
3. SAS program, to create an Excel report in HTML(MSO XML) format
4. Email system
5. Excel

Here is the sequence of steps to create and send reports using the proposed mechanism applied to the above business task example.

The REXX program reads a first record from the file with a particular territorial ID and a respective email address, and then invokes the SAS reporting program with the territorial ID as a parameter. The SAS reporting program extracts the sales quantities for this particular territorial ID and creates the report in HTML (MSO XML) format. The REXX program checks the SAS return code and based on that invokes the Email system to send an email with the attached report (emails could be sent also from SAS program).

Then the REXX program reads the next record with the next territorial ID and a respective email address, and the executions of the SAS program and Email system repeat.

All goes on until the last ID and address are read and the last executions are finished. After that all emails with the attached reports are sent and the task is completed.

REXX

REXX is a high level (script) language, it means can invoke other programs written in other languages. Each statement produces more executable code. It is interpreted (doesn't require compilation, very good for testing) and it is called often a glue language because can join different components like OS commands, functions, routines or even GUI objects.

REXX was first published in 1979 by Michael Cowlishaw from IBM. Since that time dozen of other versions have been created for different platforms and with different, extended features. The best known are: REGINA REXX (by Anders Christensen, multiplatform, 1992), REGINALD (by Jeff Glatt, for Windows, 2001) and others like NetREXX, REXX/imc, BREXX, roo.

HOW TO INVOKE SAS FROM REXX

The following REXX sample statement allows invoking a SAS program under mainframe MVS:

address LINKMVS SASXALO "**SYSPARM=**TerrID";.

The following REGINA REXX sample statement allows invoking a MySASpgm from Widows XP:

address SYSTEM sasexe **'-sysin** C:\MySASpgm.sas **–icon –nosplash'**;
where sasexe="'C:\ProgramFiles\SAS Institute\SAS\V8\sas.exe'";.

HOW TO CODE HTML

One of the solutions to write a report in HTML format is to use the Output Delivery System. The other option is to use the SAS Base PUT statements and write HTML code directly into a report file. Of course ODS saves tedious coding of PUT statements but ODS is a "broker" in HTML coding and sometimes it can create too much metadata which then can take more time to display the report under Excel after clicking an email attachment.

HOW TO SEND EMAILS

The emails can by sent from a SAS program using a **filename email** statement. To send emails from mainframe we can invoke XMITIP command from REXX.

OPTIMIZATION PROBLEMS

Optimization efforts in such automation process are focused first of all on reduction of execution time (CPU). The SAS reporting program, which is going to be invoked multiple times, should run as fast as possible and also should produce as less as possible log messages. The HTML code used to format the Excel spreadsheet should be very precise. It means the HTML code should be short to reduce the SAS program execution and also to reduce time to display a report under Excel after clicking an Email attachment.

OPTIMIZATION SOLUTIONS (HINTS)

- Store input data for the reporting SAS program in a SAS library. Create an index if you predict that it could be used by the SAS program.
- Avoid multiple file allocations in REXX.
- Eliminate unnecessary data passes in the SAS program.
- Apply optimal SAS statements like choosing WHERE or IF, IF-ELSE-IF instead of parallel IFs, OR instead of IN.
- Avoid invoking extensively SAS functions.
- Eliminate unnecessary output messages in SAS log (NOSOURCE).
- Apply correctly SAS options (SORTEDBY, KEEP, DROP).
- Use <STYLE> HTML tag to avoid repeating attributes for <TR> and <TD>.
- Apply MSO XML attributes to format more precisely a report.
- Using PUT SAS statements to write HTML code can be sometimes better than using ODS. ODS can create too much metadata which can consume time in HTML to Excel conversion.

EXAMPLE OF REXX CODE

The following REXX EXECIO DISKR statement reads all records from the external file into the array called MAILREC. Each record contains a territorial ID and email address. The **do** REXX statement is parsing each record to get a territorial ID and an email address, then the **address** REXX statement invokes a SAS program with the territorial ID as a parameter. Finally the XMITIP command is invoked to send the email with the attached report file.

```
EXECIO * DISKR MAILFILE (STEM MAILREC.
do IDX1=1 to NrLines by 1;
  parse var MAILREC.IDX1 1 TerrID 9   10  Address 49;
  address LINKMVS SASXAL0  "SYSPARM=TerrID";
  .........
  XMITIP Address From Subject Msgds Html File FileN Format(xls);
end;
```

EXAMPLE OF SAS CODE

The following SAS code shows how, using PROC DATESETS, the SALES input file was stored in the SALESLIB SAS library prior to the main batch job. Also at the same time the index SALEINDX was created.

```
PROC DATASETS LIB=SALESLIB;
  MODIFY SALES;
```

```
INDEX CREATE SALEINDX=(SA TERRID);
CONTENTS DATA=SALES;
```

The below SAS code shows usage of the NOSOURCE and sortedby options as well WHERE–IN and IF subsetting statements.

```
Options NOSOURCE;
……
PROC SORT DATA=SALES OUT=SALESLIB.SALES; BY SA TERR;
  where SA in ('008','009','010');
RUN;
……
DATA   SETUPER SETMIDL SETDOWN;
  SET   SALESLIB.SALES (sortedby=SA);
  ……
  IF SA = '009';
```

The following SAS code presents the small SAS macro called header1 used to code HTML to format the Excel spreadsheet column headings.

```
%macro header1;
put    '<tr>'                    @;
  . . . . .
put  "<th>&MTH3S</th>"       @;
put   "<th>&MTH2S</th>"      @;
put   "<th>&MTH1S</th>"      @;
  . . . . .
put    '</tr>';
%mend header1;
```

The last example of the SAS code shows how to avoid invoking extensively the SAS INTNX function. The SAS IF statement checks an invoice date. The CURWEEK – 14 and CURRWEEK – 7 expressions were used instead of INTNX('WEEK',today(),-2) and INTNX('WEEK',today(),-1) functions.

```
CURWEEK=INTNX('WEEK',today(),0);
IF INVDATE >= CURWEEK - 14 AND
   INVDATE <   CURWEEK - 7   THEN
     DO;
     ……
     END;
```

CONCLUSIONS

- REXX-SAS-HTML-Excel combination to create and send mass reports via Email system is very effective and can be executed on different platforms.
- Optimization of a reporting SAS program is very important to reduce CPU time and amount of log messages.
- REXX programs are very reliable, don't fail. REXX language allows building controllable processes.
- Using MSO XML attributes to format precisely an Excel report reduces time to display the report under Excel.
- Couple of jobs, using the above mechanism on mainframe, work very well and are very effective. For the presented business example, creation of 173 sales

reports (17 columns, average 200 rows) and sending respective 173 emails takes about 4.5 minutes of CPU.

REFERENCES

Andrews Rick, Dixon Sherry, (2005, SUGI 30): “Performance Monitoring for SAS Programs on Windows XP”

Cogswell Denis L. (2005, SUGI 30): “More than Batch – A Production SAS Framework”

Fosdick Howard (2005): “REXX Programmer’s Reference”, Wiley Publishing Inc., 720 pp.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Please feel free to contact the author at:

Stanislaw Furdal
Bristol Myers Squibb Co.
777 Scudders Mill Road
Plainsboro, NJ 08543
E-mail: stan.furdal@bms.com