

Let Them Build Their Own Reports

**Allan M. Kominek
Centerior Energy
Perry, Ohio**

Abstract

This paper will discuss an approach to a reporting facility for an activity/document tracking system. The reporting facility grew out of the myriad requests for reports from the tracking system. Rather than having the support group constantly creating new reports or modifying existing ones, the users can build their own reports. The reports are created through a series of interactive screens. The user selects the level of detail required in the report and fills in a number of blanks for subsetting the information. A batch job is created and submitted behind the scenes. The paper will discuss the strengths and shortcomings of the existing reporting facility.

Introduction

An activity tracking system was instituted to assign the responsibility of certain tasks to individuals and to hold these individuals responsible for the completion of these tasks. The increased emphasis on accountability caused an increase in the requests for ad hoc reports from the tracking system. The people who were being held accountable were interested in what activities were due in the next four weeks and their bosses were interested in what was overdue. The reports were always very similar in nature, but there was always the request to add information to the report or subset the data a different way. The support

staff found themselves copying the same old program and modifying one portion of the code and running the report. One report seemed to lead to another. Soon there were so many versions of the original program that it was becoming very difficult to distinguish one version from the other. The first suggestion to break the cycle was to turn the reports over to the user community. That would be the ideal situation, but a great deal of effort would be necessary to implement it. The majority of users requesting these reports would need a great deal of education to come up to speed with SAS®. That's why the support group inherited the task in the first place. The next suggestion was to let the system build the reports for the users. This paper will describe the tracking system and the approach taken by the support group.

The Problem

The activity tracking system that was installed consisted of a group of indexed files. These files were all indexed on an activity or document number. Most people who use the system refer to this number as the document number and that is how it will be referred to throughout the rest of this paper. The document number consists of a document type and an identifying number. Some of the files also had alternate indices. The importance of these alternate indices will become apparent as the approach to the reporting system is discussed. Each activity

that is tracked always has a main document. This main document could also have one or more minor tasks or steps. Each of these steps could also be broken down even further into one or more substeps. These then are the three main files: the main document, the step, and substep. In addition to these files, there are also a number of files that contain supporting information. These include the text file, which allows the user to enter up to 99 pages of information, a page being 13 lines (70 characters) of text; the work order file, which lists work orders that are related to the document; the equipment file, which lists equipment numbers associated to the document; and the related document file, which lists other documents that are somehow connected to the original document. The information contained in these files was entered through a series of CICS screens. These screens did a greater deal of data validation. The validation values were maintained in another file. The validation file is indexed on a field that can best be described as table type. For example, there is table of valid document types, so that users cannot create their own document types without first having them added to the table. The CICS system did not have any reporting capabilities associated with it. Reports were initially generated on an ad hoc basis. These ad hoc reports were particularly challenging in that they could request information from any or all of the files mentioned above. Copying an old report could involve deleting or adding multiple lines of code in order to meet the latest request. What just got deleted this morning could be pasted back in this afternoon. Add to this the fact that everyone seemed to use the text pages differently, e.g., some people entered the responsible engineer's name on line 11 of page 2, whereas most other people used the text pages for additional comments, and each

request became a true adventure. To further confuse the support staff, there were a number of routes that could be taken to get at the data. A user could start with the main document or they could request all of the documents for a particular work order, equipment number, or related document number.

Inspiration

In a moment of frustration someone from the support staff made the comment that wouldn't it be great if we could hand the users a checklist, they could check off what they wanted, and feed it into the computer and out would come their report. The answer was why not, let them build their own reports! The checklist could be a SAS/AF® screen and the reports could be built using Screen Control Language (SCL). The undertaking was a little more difficult than merely taking a list and feeding it into the machine.

In the Beginning

The key to the success of this undertaking was first understanding the users' needs and then planning very carefully before any coding took place. From the users perspective, the system had to be able to produce reports that contained information from any or all of the files that have just been discussed. It also had to have the capability to accept a document number, a work order, an equipment number, or a related document number as a starting point and gather the needed information from whatever files were necessary. Finally, the system had to be easy to use as the vast majority of people who would be using the system were not programmers. From the support staff came some additional requirements. The reports would have to

run in a batch environment as some of the files contained hundreds of thousands of records and previous attempts at running reports on-line had caused terminals to be tied up and accounts to time out before reports were generated. Due to the size of the files and constraints on the CPU, the programs to run the reports would have to be as efficient as possible. The information requested by the user would also have to be validated as quickly as possible, as a great deal of CPU time could be wasted reading through files to generate a blank report.

The Leg Work

The first major hurdle was reading the files. Given a document number, reading any of the above files would be no problem, they are all keyed on the document number. Any language worth it's salt should be able to curtail the I/O if you give it the key. No problem, except that the system should also have the ability to start with a work order number, an equipment number, or a related document number. Yes, the related document number file was keyed on the document number, not the related document number.

Fortunately, these files had alternate indices that consisted of these other fields. This would make life so much simpler. For example, given a work order number, the work order file could be read using the alternate index. From this read, the document numbers that are related to this work order can be obtained. These document numbers can then be used as the keys for the other files. It was also discovered that the key for the text file consisted of the document number followed by the page number, so that the programs could zero in on a particular page or range of pages. Reading the files efficiently from any of the possible starting points has been

taken care of. Score one for the good guys! The next consideration was running in a batch environment. The word batch is synonymous with JCL in the IBM world, and this application was being developed on an IBM mainframe. Originally, the reports that spurred this entire process were run with the same JCL. The lucky soul who was using it merely commented out the files that weren't being used, changed the name of the program to the latest version, and submitted the JCL. Usually the name of the latest program was something like AMK.SAS.PGM(PGM107). PGM106 was yesterday's version! This tinkering with the JCL and creating the latest version of the program is definitely what is meant by build the report, but how can this be put into a package that could easily be used by a group of users who were selfprofessed nonprogrammers?

The Solution

There are some capabilities within SAS/AF that allow you to do just that, create customized JCL and programs that can be submitted to run in a batch mode. The feature that makes this happen is the PREVIEW buffer. Any coding, be it SAS or JCL, that is contained in a SUBMIT block within an SCL program is saved to the PREVIEW buffer. Code from multiple SUBMIT blocks is appended to whatever is already in the buffer. In this manner, JCL and SAS program code can be constructed based upon a series of SCL IF conditions. For example, if the user wishes to include information from the equipment file, there is an SCL variable that is checked by an IF statement and if the condition is true, the JCL to allocate the equipment file is included in the code saved to the PREVIEW buffer. The JOB card is constructed using the automatic macro variable &SYSJOBID,

which contains the userid that invoked the current SAS session. The actual SAS program code can also be included after the JCL. This code is also customized depending on which information the user is requesting. This includes reading files based upon the primary or alternate index, constructing the keys for additional files that will be needed for the report, and also constructing subsetting IF statements to limit the scope of the data. Once the entire program is constructed, it can be saved to an external file by using the PREVIEW function with the FILE option.

```
rc=PREVIEW('FILE','AF.SUBMIT');
```

The contents of the external file can then be submitted using the SYSTEM function which issues TSO commands.

```
rc=SYSTEM('SUBMIT AF.SUBMIT');
```

The PREVIEW buffer is then cleared to allow another batch job to be constructed.

```
rc=PREVIEW('CLEAR');
```

The program is being submitted in a batch mode and is utilizing the keys to read the data efficiently. So far, so good. What about validating the information requested before the batch job is submitted, so all that efficiency isn't used to read records that will be wiped out by a subsetting IF statement? There is a validation file that was mentioned earlier. An autoexec program reads the validation file and stores all the values in a series of SCL lists and temporary data sets that are used to validate user entries. The report can not be submitted unless valid values have been entered. The users and the support group have added to the validation tables to enhance them. Such information as department designations and printer

destinations have been added. The printers were added to prevent that nasty old JCL error: Invalid Destination. The data from the temporary data sets are used to create selection lists. If the user is unsure of what the acceptable values for a particular field are, a question mark (?) can be entered to pull up the selection list.

Putting It All Together

The support group designed a series of screens that walk the user through the process of building the report. The first menu asks them to choose their means of approach: Document Number, Work Order, Equipment Number, or Related Document Number. The next screen asks for a partial or full key value, e.g., the Equipment Number. The screen also asks which additional information is desired. If the Equipment Number is entered, the report will automatically include the main document information, but information from the step, substep, the text file, the work order file, and the related document file could also be included. The screen following this requests information about subsetting the data. Finally the user is prompted to enter the printer destination and the requester's name. The requester's name appears in a PROC EXPLODE at the front of the report to help identify the report in the event that two or more users submit similar reports to the same printer.

Making It Better

The current production system was arrived at through a joint effort between the users and the support group who took a pilot version of this system and began to make enhancements. In addition to the requester's name, a page listing the key value and the subsetting criteria is also printed out on the

first page of the report. This was in the event that the user would be submitting several variations of the same request in succession. The users also requested the ability to enter multiple values for various fields. This was accomplished by creating an extended table screen that allows the user to enter virtually an unlimited number of values. Only one extended table program was written. The values entered are stored in a temporary data set along with the field name that they relate to. They are later used to create the key values or subsetting IF statements. The user signals the system that multiple values are to be entered by entering an asterisk (*) in the field rather than a value. Originally, the subsetting information used AND's exclusively.

IF DEPT='ENG' AND UNIT='MECH';

There were certain instances where an OR would have been more appropriate. For example, some users were interested in reporting on all the documents initiated or closed during a certain time period. This could be done by comparing the initiation date and the close date against the desired date range. However, this is an either or situation. A screen was designed that

displayed the selection criteria and permitted the user to overtype the AND's connecting the different expressions with OR's. Another suggestion has been to create a window where the user can change the sort sequence. Currently, the reports are sorted in document number sequence.

Shortcomings

This system is extremely flexible, but in order to accomplish that flexibility, the code is very complex and there is an awful lot of it. It took a great deal of time and effort to get it functioning properly. It is going to take an equally long time to document the system to the point where someone outside of those people who were actually involved in the development will be able to maintain it.

Conclusion

Using SAS/AF and SCL, a system has been put together that answers the needs of the user community in an efficient and timely manner. The system has also provided the opportunity for the user community and the support group to work as a team.

SAS and SAS/AF are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.