

IDCAMS to SAS - The Parser Two-Step

Andrew T. Kuligowski - Nielsen Media Research

ABSTRACT

On occasion, one is given a task which appears partly suited towards the SAS® System - and partly unworkable under it. This paper describes an example of such a problem and the two-step technique chosen to overcome it. The first step uses IDCAMS as a tool to produce an output listing, which is subsequently parsed and processed by a second SAS step.

THE PROBLEM

Our analysts will request an MVS job to process data for a selected date, which could be anywhere between 1 and 60 days ago. The job requires a reference file that corresponds to the requested date. However, there are many versions of that reference file. The only indication as to which version of the reference file to use is the final suffix node of the file's MVS dataset name. In most cases, that node contains a Julian date corresponding to the Last Effective Date of the file. (The "First Effective Date" is not provided - it must be uncovered by determining the Last Effective Date of the previous file!) To further complicate the issue, there is one version of the reference file that does not have a suffix node - this is the most current version of the reference file.

Our task is to identify the proper reference file to use, and build a valid line of JCL containing the name of that reference file.

SOLUTION STEP 1 - IDCAMS

Our first task is to obtain a list of all versions of the reference file. We selected IDCAMS as our tool of choice. (By definition, IDCAMS is program interface to the Access Method Services facility for VSAM files. However, it also can be used for non-VSAM files.) The following command was issued from an IDCAMS MVS step:

```
LISTCAT LEVEL(ACCOUNT.LOOKUP.FILE)
```

The SYSPRINT was captured in a sequential file, and passed to the next step of our job.

SOLUTION STEP 2 (Part A) - SAS PARSER

The trivial part of the task has been addressed. We must now take the output from the previous step and process it. In order to do that, we will use the SAS system to parse our output - valid dataset names will be kept, but all other extraneous information from the IDCAMS listing will be dropped.

The key to any successful parser is to identify and utilize patterns in the material being parsed. (Anyone who has used the Find function of their full-screen text editor to locate the word "Cary" in their SAS output listing has, perhaps unknowingly, already utilized this premise.) In the case of the IDCAMS listing, we identified one such pattern - every line of our listing containing a valid dataset name precedes that name with the word "NONVSAM" and a series of dashes. Therefore, we look for the word "NONVSAM" in position 2 of each line of output - position 1 being reserved for carriage control characters. If a line passes this test, we read in the next two "words" - a series of dashes and a valid reference dataset name. (All lines failing this test are ignored.) We have now succeeded in passing our list of dataset names to SAS.

We also perform a little editing and preparatory processing within the same DATA step. The last 5 characters of each dataset parsed in

are checked. If they are numeric, we assume them to be a valid Julian date, and convert them to a SAS date. If they consist of the word ".FILE", we know that this represents the most current reference file, and assign an arbitrary end date of 31 December 3000. If neither condition is true, we have discovered an erroneously-named file which must be dealt with. (Please note that additional error checking is performed on the supposedly "valid" values discussed earlier in this paragraph. It has been omitted from this text due to space limitations.)

SOLUTION STEP 2 (Part B) - SAS PROCESSING

At this point, we have a list of all reference datasets and their last effective date. We now need the selected date to be processed; it will be passed in as a macro variable in SAS date format.

The SAS dataset containing the list of reference datasets is sorted by descending Last Effective Date. This sorted dataset is now processed by one final SAS data step. The date on each reference dataset is compared to the date to be processed. If our process date is earlier than the Last Effective Date of the reference file, we know that this might be the proper reference file. However, without a First Effective Date, we cannot be sure. (Remember, the First Effective Date of this reference file is the day after the Last Effective Date of its predecessor.) Therefore, we store the name of the reference file in a work variable, and move on to check the Last Effective Date of the next reference file in our parsed dataset. If the date on the new reference dataset falls before the date to be processed, then we know the previous reference file was the one to be used. We take the dataset name from the work variable and stop processing. (The date on the most recent reference file, as stated above, will be set so far in the future that our date to be processed will never exceed it. Therefore, we will always check the effective dates of at least two reference files.)

In the unlikely event that we reach the end of the list of reference datasets without having found the proper one, we assume that the last reference dataset is the appropriate one for our selected date. However, we warn the user to verify this before continuing with both a printed message and anon-zero condition code.

CONCLUSION

The procedure described above is one example of how the output of a utility can be parsed and utilized by a subsequent SAS step. We have written and used others - one of which even parsed the SASLOG from a previous SAS step! The specific example described in this paper may or may not be of use to the reader. However, the concept of parsing a listing for further processing under the SAS system can be applied under any operating system that the SAS system can be licensed.

REFERENCES / ACKNOWLEDGMENTS

SAS is a registered trademark of SAS Institute, Inc. All rights reserved. * indicates USA registration. IDCAMS and VSAM are registered trademarks of International Business Machines Corporation.

Special thanks to Luan Ardinger, Andy Parks, and Joe Ruberte for their contributions to the successful completion of this paper.

POSTSCRIPT

This paper was originally planned to discuss two variations on the IDCAMS / PARSER methodology theme. The first, parsing a list of datasets to find the one that was effective on a given date, is described above. The second, parsing a list of datasets to determine the existence of a particular dataset, was removed from the paper.

As part of the preparation for writing this paper, I referred to other papers presented in the Coders Corner section of other conference proceedings. During this process, I stumbled across "Undocumented Features to Test for Existence of Data Sets", by John Beach of BlueCrossBlueShield of Connecticut Inc. in the 1993 NESUG Proceedings. John's paper describes an undocumented feature that will allow SAS to verify or deny the existence of an MVS dataset. John's paper provided the tools to perform the function I needed - except I could now accomplish it in approximately 20 lines of SAS code rather than in two separate JCL steps!

I had an important lesson reinforced during the preparation of this paper. Look over your Conference Proceedings before undertaking a coding challenge. Someone may have already encountered and solved your problem for you.

APPENDIX - The SAS Parser

```

/*-----*
 * SOLUTION STEP 2 (Part A) - SAS PARSER *
 *-----*/
DATA ALL_REF (KEEP=REFNAME REFSUFFIX REF_END);
  LENGTH DASHES $ 8;
  REFNAME $ 44;
  REFSUFFIX $ 5;
  INFILE REFLIST;
  INPUT @ 2 NONVSAM $CHAR7. @;
  IF NONVSAM ^= 'NONVSAM' THEN RETURN;
  /* Else Do (Implied) */
  INPUT DASHES
        REFNAME;
  REFSUFFIX = SUBSTR(REFNAME,LENGTH(REFNAME)-4);
  IF ^VERIFY(REFSUFFIX,'0123456789') THEN
    REF_END = DATEJUL(REFSUFFIX - 1);
  ELSE IF REFSUFFIX = '.FILE' THEN
    REF_END = '31DEC3000'D;
  ELSE DO;
    /* Various error tracking & reporting */
    END;
  OUTPUT;
RUN;

/*-----*
 * SOLUTION STEP 2 (Part B) - SAS PROCESSING *
 *-----*/
PROC SORT DATA=ALL_REF;
  BY DESCENDING REF_END;
RUN;

DATA NULL;
  LENGTH PREV_REF $ 44;
  RETAIN CURDATE PREV_REF;
  IF _N_ = 1 THEN
    CURDATE = DATEJUL(&PRCSST);
  SET ALL_REF END=LASTREC;
  IF REF_END < CURDATE THEN DO;
    LINK WRITEREC;
    STOP;
  END;
  ELSE PREV_REF = REFNAME;
  IF LASTREC THEN DO;
    LINK WRITEREC;
    FILE LOG;
    PUT 'WARNING: '
        '(Appropriate end of file message)';
    ABORT RETURN 01;
  END;
  RETURN;
WRITEREC:
  FILE REFSSET;
  Q_POINT = LENGTH(PREV_REF) + 23;
  PUT @ 1 '//SET0001 SET REFDN='
      @ 23 PREV_REF
      @ Q_POINT ' ';
  PUT 'NOTE: (Appropriate normal '
      'termination message)';
  ABORT RETURN 01;
  RETURN;
RUN;

```