# Formats to the Rescue

**Gary McQuown**
**Data and Analytic Solutions, Inc. Fairfax, VA**

**ABSTRACT**
Formats are seldom utilized to their full potential. In addition to making a date comprehendible, they can resolve a number of data related issues and improve efficiency. This is especially true as the increasing quantity of data makes sorting, indexing, joining and merging more challenging.

This presentation focuses on **when, where, how and why** formats may be used to expand and improve upon typical and atypical ETL processes. Two examples will be provided. Anyone familiar with the data step may use these techniques in a variety of ways.

**INTRODUCTION**

Informats and formats associate one value with another. Informats determine how a value will be read and formats determine how a format will be written. While this sounds and is very simple, it can be used in very complex situations with great effect and efficiency. Using formats and informats may seem complicated because so many come with the software, they are frequently added to and even more can be user defined, but this just makes them more versatile.

Under what might be considered "normal" circumstances, data is fairly simple (single key field), manageable in size (a few million observations or less), static (not updated frequently) and the task is fairly simple. So under "normal" conditions we can use the data step merge, proc SQL and possibly proc Datasets append to combine data. But as the data becomes more complex (multiple key fields), larger (millions or billions of observations), more dynamic (updated frequently) and the tasks become more complicated, alternative solutions such as formats should be considered. This paper was inspired by two such instances.

The Megadata task involved linking multiple variables from frequently updated data sets with billions of records to categorical values in dozens of smaller data sets. The Reversal task required select observations to be conditionally flagged according to characteristics in another data set.

**WHEN**

**Formats are a great tool to pass information from a relatively small data set to a larger one, or to conditionally match many fields to one or one field to many.**

The Megadata task was to link pharmaceutical prescription drug data with multiple reference files in an efficient manner for the production of numerous reports, queries and cubes. As the data sets contained a number of fields that were to be queried on and contained billions of observations that would be updated frequently, the use of merges, joins and SQL queries were problematic. A further complication was that the most important key field, the medical providers ID would need to be matched to one of many alternate tables. Hashing was a viable option, but was not considered as versatile as the use of user defined formats.

The Reversal task was a matter of properly cleaning some data. Millions of transactional records were pulled from SAP into a primary SAS® data set for analysis. But some observations in the primary data set represented transactions that were later amended or cancelled. This information was contained in a separate reversal file. Both files contained five variables that together identified a series of unique transactions. If the transactions for a certain key combination appeared in the reversal file, the three earliest observations in the primary file would need to be flagged for deletion. A previous version of this task read the primary data set once for each observation in the reversal file, resulting in a thirty-hour process. Modifying the code to use formats so that the primary file was only read once reduced processing time to less than five minutes.

**WHERE**

**Formats can be used in many places.**

The most obvious use of formats is probably converting the numeric value of a SAS date to something more comprehendible. This can take place within a data set as an attribute, within a data step as illustrated by FOO1 below, or within a number of procs. Formats can also be used to combine data, functioning as a relatively efficient look up table. The two capabilities, combined with the ***WHERE*** data set option can solve many ETL related problems.

A recent SAS-L post by Sigurd Hermansen explains why: *"A WHERE clause would subset each yearly dataset as it is being streamed into an input buffer. This subsetting operation does not require an index. In SAS it generally works very fast. If you are selecting more than 20% of each yearly dataset, SAS likely won't use an index anyway.*

*Scanning a dataset and filtering works faster.*

*Since the WHERE clause filters out unneeded rows as they are being read, it can greatly reduce the volume of data being written by a Data step or query. Look for efficiency there first."*

```
title Example 1;

data foo2;
  set f001
    (where=(put(key,myfmt.)='1'));
     sasdate=17064;
     worddate=put(sasdate, worddate.);
     date9 = put(sasdate, date9.);
     put _all_;
run;

sasdate=17064
worddate=September 20, 2006 date9=20SEP2006

proc print data=f002;
  var sasdate;
   format sasdate mmddyy6.;
run;


092006
```

In the fist example, the `(where=(put (key , myfmt. )=1))` portion selects observations from F001 only when the value of KEY resolves to "1" . Therefore if the primary data set is large and the format is relatively small, the process can be extremely efficient. `The worddate = put(sasdate, worddate.)` portion converts the SAS date value of 17064 to the formatted and more easily comprehendible values of `September 20, 2006` and `20SEP2006.` Procs such as Print, Means, Summary, Tabulate, Report, Graph, and others allow the use to formatted values to reduce the necessity to manipulate the data prior to running the proc.

**HOW**

**Proc Format with the CNTLIN option allows users to create their own formats.**

Once the format has been created, it can be used to conditionally select or process observations and to create new variable as needed.

Both the Megadata and Reversal tasks applied user defined formats. For the reversal task, the format consisted of the results of a Proc Freq run on a single calculate variable which consist of a underscore delimited concatenation of the five key variables (Example 2). The resulting N value represented the number of observations in the primary data set for that unique combination of key

variables that needed to be flagged for deletion. The remainder of the task was to:

1) Identify the key combinations where a reversal is to take place.

2) Obtain the number of observations to flag for deletion.

3) Flag the correct number of observations by decreasing the "count" to be flagged as each observation is processed.

You begin the process by creating a data set that contains at least the following information:

**FMTNAME**= The name of the format. This must have a length of less than eight.

**TYPE**= Character ("C") or Numeric ("N")

**START**=Contains the values that you with to be converted.

**LABEL**=Contains the values that START will be converted into.

**HLO**= Dictates how values not contained in START will be treated. Includes Hi ("H"), Low ("L"), and Other ("O").

There are a number of additional control variables that typically resolve to default settings that may also be set by the user (see Appendix 1). Although not normally necessary, they allow greater control and versatility. The HLO variable is not technically required and may be omitted, but not including a default "catch all" category has significant risks. If the value to which the format is applied does not match a value or range within the START field, the format will not change the ORIGINAL value. This may have unexpected and undesired results.

The following example shows one method using the concatenated five digit key field (CAT_KEY) and the N value (COUNT) resulting from a proc freq. If only a temporary format is desired, the libname and library option should be omitted.

```
Title Example 2;
Libname library "C:\NESUG";

data reversal
  (keep =     fmtname
              hlo
              label
              start
              type);
   retain fmtname "REV"
          type "N";
   set Freq_output
      end = lastrec;
      start = CAT_KEY;
      label = COUNT;
      output;
  if lastrec then
  do;
      hlo = "O";
      label = "0";
      output;
  end;
run;

proc format cntlin = reversal library=library;

run;
```

The formats is then conditionally applied as in the Example 3.

```
Title Example 3;
Data Reversal;
  Retain count;
  Set Primary;
    By cat_key;
  If first.cat_key and Delete_Flag= . then count = put(cat_key, rev.);

  if count > 0 then
  do;
     Delete_Flag="&sysdate"d;
     Count =  count -1;
  end;
run;
```

Formats for the Megadata task were created in the same manner. But the task was more complex due to the nature of the medical providers ID (`prescriberid`) in the primary data sets. This field could contain the medical providers Name, their Social Security Number, one of their DEA numbers, or one of their EIN (Tax ID) numbers. Therefore a format had to be created for each possibility so that a single unique value (MP_ID) would be "merged" with all observations pertaining to that medial provider in the following manner. Please note that the most reliable matching criteria takes place first and that all matches are flagged by how they match was performed.

```
/** match by DEA number ? **/
  if put(prescriberid, $deanbr.)
   ^= "MISSING" then
   do;
     MP_ID =
       put(prescriberid, $deanbr.);
       match  = "DEANBR";
   end; else

/** match by EIN - TAX ID **/
  if put(prescriberid, $taxid.)
   ^= "MISSING" then
   do;
     MP_ID =
       put(prescriberid, $taxid.);
       match  = "TAXID";
   end; else

/** by SSN **/
  if put(prescriberid, $SSN.)
   ^= "MISSING" then
   do;
       MP_ID =
       put(prescriberid, $ssn.) ;
       match  = "SSN";
   end; else

/** by name, city and state **/
  if put(prescriberid, $name.)
   ^= "MISSING" and
     put(prescriberid, $CITY.)
    = CITY and
      put(prescriberid, $STATE.)
    = STATE
 then
   do;
     MP_ID
      = put(prescriberid, $name.);
     match  = "NAME_LOC";
```

```
   end; else

/** no match **/
    do;
       MP_ID = " ";
       match  = "NO MATCH";
    end;

/** append additional info **/
 FULL_NAME=
     put(prescriberid, $FNAME.);
 PRIMARY_CITY=
     put(prescriberid, $CITY.);
 PRIMARY_ST=
     put(prescriberid, $STATE.);

run;
```

**WHY / CONCLUSION**

**Formats are versatile, efficient and easy to use.**

The ability to use formats in the data set WHERE option, the IF statement, with functions such as PUT and INPUT and with various procs makes them extremely versatile. The same qualities and not having to sort or index the primary table makes them very efficient. And given the simplicity of the Proc Format CNTLIN option, user defined format are easy to create and use. All of these reasons make them an invaluable tool for SAS programming, especially when the data or complexity of the task make merges / joins and indexes problematic.

**REFERENCES:**

Jonas V. Bilenas “The Power of PROC FORMAT“

Jack Shoemaker “Ten Things You Should Know About PROC FORMAT”

The SAS Institute, “SAS Online-DOC”

**ACKNOWLEDGMENTS**



**CONTACT INFORMATION**

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

Gary McQuown
Data and Analytic Solutions Inc.
3127 Flintlock Road
Fairfax, VA 22030
Tel: 703.628.5681 Fax: 703-991-8182
mcquown@DASconsultants.com
www.DASconsultants.com

Publications by the author and others can be found at http://www.dasconsultants.com/pubs.html

## APPENDIX:

**Variables in the output control data set:**

The SAS Online Doc lists the variables in the output control data set as:

DEFAULT a numeric variable that indicates the default length for format or informat

END a character variable that gives the range's ending value

EEXCL a character variable that indicates whether the range's ending value is excluded. Values are

Y the range's ending value is excluded

N the range's ending value is not excluded

FILL for picture formats, a numeric variable whose value is the value of the FILL= option

FMTNAME a character variable whose value is the format or informat name

FUZZ a numeric variable whose value is the value of the FUZZ= option

HLO a character variable that contains range information about the format or informat in the form of eight different letters that can appear in any combination. Values are

F standard SAS format or informat used for formatted value or informatted value

H range's ending value is HIGH

I numeric informat range (informat defined with unquoted numeric range)

L range's starting value is LOW

N format or informat has no ranges, including no OTHER= range

O range is OTHER

M MULTILABEL option is in effect

R ROUND option is in effect

S NOTSORTED option is in effect

LABEL a character variable whose value is the informatted or formatted value or the name of an existing informat or format

LENGTH a numeric variable whose value is the value of the LENGTH= option

MAX a numeric variable whose value is the value of the MAX= option

MIN a numeric variable whose value is the value of the MIN= option

MULT a numeric variable whose value is the value of the MULT= option

NOEDIT for picture formats, a numeric variable whose value indicates whether the NOEDIT option is in effect. Values are

1 NOEDIT option is in effect

0 NOEDIT option is not in effect

PREFIX for picture formats, a character variable whose value is the value of the PREFIX= option

SEXCL a character variable that indicates whether the range's starting value is excluded. Values are

Y the range's starting value is excluded

N the range's starting value is not excluded

START a character variable that gives the range's starting value

TYPE a character variable that indicates the type of format. Possible values are

C character format

I numeric informat

J character informat

N numeric format (excluding pictures)

P picture format