Paper CC-02

Format Follows Function: User-Written Formats and User-Written Functions that talk to the SAS Metadata Server

Philip James Busby, SAS Institute

ABSTRACT

The content shows how to create a user-written function in SAS that can be used in a data-step; not a macro, but an actual user-written function. And then I take it a step further and call the function from a user-written format. By doing this, formats can do things that they couldn't have possibly done before, and I show a format that translates a field containing a metadata ID into the actual name of the object on the metadata server.

INTRODUCTION

This paper explores two capabilities in SAS, and then combines them in an unexpected way to further extend the capabilities of SAS. The first section explores how to create a custom user-written format, which is often used for doing lookups on coded datasets when displayed. The second section explores how to create a custom user-written function with PROC FCMP. This is a relatively new feature that isn't commonly used as it has a lot of overlap in functionality with calling user macros. There are still some things it can do that a macro can't; for example a user-written format can't call a user macro, but it *can* call a user-written function.

CREATING A CUSTOM FORMAT

Custom formats have been around since the early 1990s, and is fairly well-understood capability of the SAS language, and fairly pervasive in use by the SAS programming community. SAS comes with a lot of formats, but it's often useful or necessary to create industry-specific formats. Say for example we have the following table of students and their respective number of credit hours completed

```
data students;
  length student_id student_name $12
         credit_hours 8.;
  input  student_id $ student_name $
         credit_hours;
  datalines;
200800001 Alice  115
200800002 Bob    119
200900001 Carol  91
200900002 Dave   80
200900003 Erin   83
201000001 Frank  42
201100001 George 15
;
run;
```

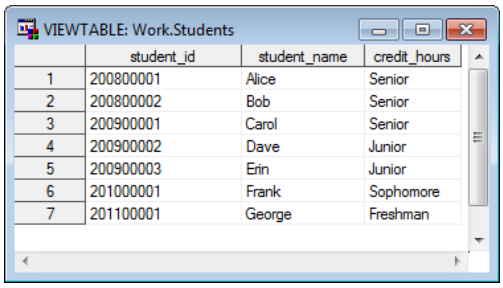
Student ID	Student Name	Credit Hours Completed
200800001	Alice	115
200800002	Bob	119
200900001	Carol	91
200900002	Dave	80
200900003	Erin	83
201000001	Frank	42
201100001	George	15

At this school, class is determined by credit hours completed. Seniors have at least 90 hours. Juniors have at least 60, Sophomores have at least 30, and any students under that are Freshmen. The easiest way to put this on the dataset would just be to add it by calculating it from the credit_hours, but then it would have to be recalculated when the credit_hours changes. Derived data can be complicated to maintain, and when speed isn't an issue it's better not to store it. Instead, we can just have the class determined from a format.

```
proc format;
  value class(multilabel)
    0-29 = 'Freshman'
    30-59 = 'Sophomore'
    60-89 = 'Junior'
    90-high = 'Senior'
    other = '**ERROR**'
    ;
run;
data students;
```

```
  set students;
  format credit_hours rank.;
run;
```

This makes the viewer show the rank, instead of the hours, and computes it when it displays it.

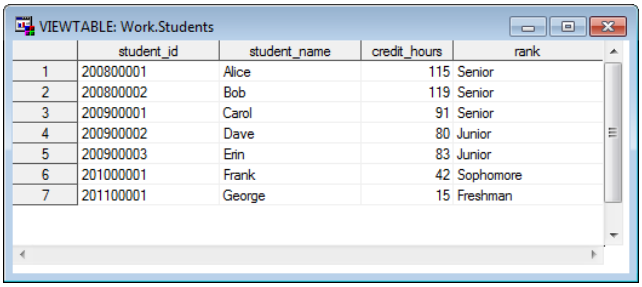


	student_id	student_name	credit_hours
1	200800001	Alice	Senior
2	200800002	Bob	Senior
3	200900001	Carol	Senior
4	200900002	Dave	Junior
5	200900003	Erin	Junior
6	201000001	Frank	Sophomore
7	201100001	George	Freshman

CREATING A CUSTOM FUNCTION

Originally developed as a programming language for SAS/STAT, SAS/ETS, and SAS/OR procedures, a new but less commonly used feature of SAS 9 is the ability to create a custom function, which can be used from a DATA Step, as well as PROC SQL blocks. The language is not exactly the same as DATA Step, it is very similar.

```
options cmplib = work.funcs;
proc fcmp outlib=work.funcs.helper;
  function classrank(hours) $12;
    select;
      when(hours >= 90) return('Senior');
      when(hours >= 60) return('Junior');
      when(hours >= 30) return('Sophomore');
      when(hours >= 0) return('Freshman');
      otherwise return('**ERROR**');
    end;
  endsub;
run;
data students;
  set students;
  format credit_hours 8.;
  rank = classrank(credit_hours);
run;
```



	student_id	student_name	credit_hours	rank
1	200800001	Alice	115	Senior
2	200800002	Bob	119	Senior
3	200900001	Carol	91	Senior
4	200900002	Dave	80	Junior
5	200900003	Erin	83	Junior
6	201000001	Frank	42	Sophomore
7	201100001	George	15	Freshman

CUSTOM FUNCTIONS THAT CALL SAS FUNCTIONS

So far this is nothing too special, just functions that can be repeatedly called to save us time and keystrokes. Custom functions become very useful when they're used to call other functions.

USING A CUSTOM FUNCTION IN A CUSTOM

To start out with, lets create this dataset with a list of Metadata URIs to our SAS Metadata Server:

```
 data uris;
  uri = 'omsobj:PhysicalTable:/A5BEUBXV.BF000002'; output;
  uri = 'omsobj:PhysicalTable:/A5BEUBXV.BF000003'; output;
  uri = 'omsobj:PhysicalTable:/A5BEUBXV.BF000004'; output;
run;
```

URI
omsobj:PhysicalTable:/A5BEUBXV.BF000002
omsobj:PhysicalTable:/A5BEUBXV.BF000003
omsobj:PhysicalTable:/A5BEUBXV.BF000004

Doesn't look too pleasant, but the metadata_getattr function can be used to get the "name" attribute from the server by doing the following:

```
data uris;
  set uris;
  length name $200 rc 8.;
  rc = metadata_getattr(uri,'Name',name);
run;
```

But those metadata functions can be tricky to use, and are often strung together in a series of loops in order to traverse the metadata structure (for example, to get the Library to which a table belongs). Calling them can be simplified by creating a function to do so.

```
proc fcmp outlib=work.funcs.helper;
  function metaname(uri $) $;
    length value $200;
    rc = metadata_getattr(uri,'Name',value);
    return(value);
  endsub;
quit;
```

Now lookups can be done easily,

```
data uris;
  set uris;
  name = metaname(uri);
run;
```

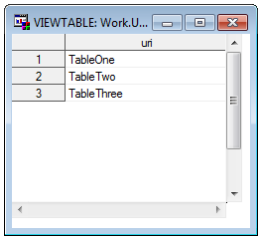
CALLING A CUSTOM FUNCTION FROM A CUSTOM FORMAT

These custom functions can also be called from a custom format.

```
proc format;
  value $metaname other=[metaname()];
run;
```

And attaching this format to our uris dataset causes them to display the name, looked up from the SAS Metadata Server when it's displayed.

```
data uris;
  set uris;
  format uri $metaname.;
run;
```



CONCLUSION

PROC FORMAT allows users to create their own formats, which is immediately useful for data compression and lookup tables. PROC FCMP allows users to create their own functions, which may seem to overlap functionality with macros, however these functions can be called from user-written formats, and allows user-written formats to do things otherwise impossible. This paper has shown how to combine these two features of the SAS language to dynamically call a SAS metadata server on display of a SAS dataset to resolve a metadata ID into a name.

RECOMMENDED READING

- Watts, P. 2006. “Using Database Principles to Optimize SAS Format Construction.” *Proceedings of the SAS Global Forum 2008 Conference*.
- Wang S, Zhang, J. “Developing User-Defined Functions in SAS: A Summary and Comparison.” *Proceedings of the SAS Global Forum 2011 Conference.*

CONTACT INFORMATION <HEADING 1>

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

Philip James Busby
SAS Institute
32767 SAS Campus Drive
Cary, NC 27612
+1(919)522-2763
philip.busby@sas.com
http://philihp.com