How to Build a Data Dictionary – In One Easy Lesson
Gary E. Schlegelmilch, Enterprise Information Services (EIS), Inc.

ABSTRACT

In the wonderful world of programming, the Child Left Behind is usually documentation. The requirements may be thoroughly analyzed (usually on a combination of phoned-in notes, e-mails, draft documents, and the occasional cocktail napkin). Design is often on the fly, due to various restraints, deadlines, and in-process modifications. And doing documentation after the fact, once the program is running, is, well, a great idea – but it often doesn’t happen.

Some software tools allow you to build flow diagrams and descriptions from existing code and/or comments embedded in the program. But in a recent situation, there was a lament that a system that had been running in the field for quite a while had no Data Dictionary – and one would be really handy for data standardization and data flow. SAS® to the rescue!

INTRODUCTION

The idea behind this paper is to show how easily a fairly detailed Data Dictionary can be built with any SAS library of data as input. The example shown is actually data from Oracle tables, which shows you its flexibility.

The idea was to build a process that would create a Data Dictionary dataset that could be updated regularly. Eventually, it could them be exported to Excel for common use, and then periodically run against the library (or libraries) to get a fresh picture of the structure, then apply any external updates so no information is lost. Once you see how easy it is to get the base information, you’ll also see how flexible you can make it.

KEYWORDS

Data dictionary, PROC TRANSPOSE, PROC SQL

HOW IT’S DONE

I’ve included the full source code as an Appendix, so you don’t have to make yourself crazy cutting and pasting. But this will take you through the process, step by step. The things you will have to put in according to your local processes are in <>.

First, define the library of datasets. I named it ORALIB for this paper.

```sas
libname ORALIB oracle path= '<location of the data library>'
   schema=<schema name>
   user=<user id>
   password='<password>';```

Use PROC SQL to read in the data. We use PROC SQL here because the DICTIONARY data is only available from PROC SQL. We sort it inline to prepare the data for the PROC TRANSPOSE step. In this case, we are not only looking for data that is listed in more than one dataset, but we are also looking for the potential of problems by having data items with the same name defined differently in different datasets, which could explain things like truncated data or error for using numeric data in a string process.

```sas
proc sql;
create table WORK.DD2
   (DROP=npos varnum idxusage sortedby xtype notnull precision scale
    transcode)
AS
   select * from DICTIONARY.COLUMNS
   where libname='ORALIB'
   order by name, type, length, format, informat, memname;
quit;
```
This results in this initial dataset:

```
/* get all the dataset fields into a single observation, but */
/* break them out by type, length, format and informat - in */
/* case of different utilizations of the same fieldname, */
/* and/or aid in data standardization. */

proc transpose data=work.dd2
   out= work.dd3 (drop=_name_ _label_);  
   var memname;
   by libname name type length format informat label;
run;

proc datasets lib=work;
   delete dd2;
quit;
```
This will produce this:

```
/* Build the data dictionary dataset. */
data work.dd4;

/* The LABEL statement serves also to put */
/* the Data Dictionary fields in a logical */
/* reading order. */
LABEL  libname = "Library Name"
       name = "Field Name"
       libraries = "Contained in Dataset"
       label = "Description"
       type = "Data Type"
       length = "Field Length"
       format = "Field Format"
       informat = "Field Informat";
length LIBRARIES $2100.;
set work.dd3;
LIBRARIES=catx(' ', of col:); /* use the OF and ',' wildcard to */
/* concatenate ALL COLnnn fields, with */
/* ' ' as delimiter for readability */
drop col:; /* use wildcard ':' to remove all COLnnn fields */
run;
```
And now, you're the proud owner of your Data Dictionary!

```
proc datasets lib=work;
  delete dd3;
quit;
```
CONCLUSION
In just about 60 lines of executable code (and in this case, less than 12 seconds clock time), you have a good tool to always keep an eye on your data.

REFERENCES


ACKNOWLEDGMENTS
Much thanks to Kirk Paul Lafler for writing such a great book on PROC SQL; and the hard-working chairs of the 2014 SESUG conference, Abbas Tavakoli and Darryl Putnam.

RECOMMENDED READING
- *Professional SAS Programmer’s Pocket Reference*, Rick Aster

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

Name: Gary E. Schlegelmilch  
Enterprise: Enterprise Information Services (EIS), Inc.  
Address: USCG ISD Bldg 63 ALC  
City, State ZIP: Elizabeth City, NC 27909  
Work Phone: (252) 384-7215  
E-mail: Gary.E.Schlegelmilch@uscg.mil  
Web: www.goeis.com

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.

Appendix. Source Code

/**********************************************************************
* Job:           create_data_dictionary.sas                          *
*                                                                    *
* Description:   Small SAS program to create a working Data          *
*                Dictionary from any SAS library                     *
*                                                                    *
* Source Tables: Any SAS data library, DICTIONARY.COLUMNS            *
*                                                                    *
* Target Table:  WORK.DD4, output Data Dictionary dataset            *
*                                                                    *
* Generated on:  Wednesday, February 5th, 2014                       *
*                                                                    *
* Generated by: Gary E. Schlegelmilch, EIS                           *
*                                                                    *
* Version:       1.0                                                 *
**********************************************************************/
libname ORALIB oracle path= '<location of the data library>'
    schema= <schema name>
    user= <user id>
    password="<password>";

/* First, read the COLUMNS dataset and get individual field attributes. */
proc sql;
  create table WORK.DD2
    (DROP=npos varnum idxusage sortedby xtype notnull precision scale
     transcode)
    AS
      select * from DICTIONARY.COLUMNS
      where libname='ORALIB'
      order by name, type, length, format, informat, memname;
quit;

/* get all the dataset fields into a single observation, but */
/* break them out by type, length, format and informat - in */
/* case of different utilizations of the same fieldname, */
/* and/or aid in data standardization. */
proc transpose data=work.dd2
    out= work.dd3 (drop=_name_ _label_);
    var memname;
    by libname name type length format informat label;
run;

proc datasets lib=work;
  delete dd2;
quit;

/* Build the data dictionary dataset. */

data work.dd4;
  /* The LABEL statement serves also to put */
/* the Data Dictionary fields in a logical */
/* reading order. */

LABEL
  libname  = "Library Name"
  name     = "Field Name"
  libraries= "Contained in Dataset"
  label    = "Description"
  type     = "Data Type"
  length   = "Field Length"
  format   = "Field Format"
  informat = "Field Informat"

length LIBRARIES $2100.;
set work.dd3;
LIBRARIES=catx(', ', of col:); /* use the OF and ':' wildcard to */
/* concatenate ALL COLnnn fields, with */
/* ', ' as delimiter for readability */
drop col;; /* use wildcard ':' to remove all COLnnn fields */
run;

proc datasets lib=work;
delete dd3;
quit;