Set Yourself Free  
–Use ODS Report Writing Technology in SAS Enterprise Guide  
Instead of Dynamic Data Exchange in PC SAS

Robert Richard Springborn,  
Healthcare Outcomes Center, Office of Statewide Health Planning & Development,  
Sacramento, California

ABSTRACT
The ability to prepare complex reports and convey your message in a clear and concise manner is an absolute imperative in today’s sophisticated business environment. Clearly designed reports can enhance your organization’s credibility and reputation. The traditional use of Dynamic Data Exchange (DDE) to produce custom designed reports is the result of widespread and popular use of Microsoft Excel. However with most business organizations transitioning to SAS Enterprise Business Intelligence (EBI), where DDE is not compatible, ODS Report Writing technology is a powerful alternative to create custom designed reports in Adobe as a secured “pdf” file. The impetus for this paper was the need to create hospital specific data discrepancy reports which compare California CABG Outcome Reporting Program (CCORP) clinical data to OSHPD’s hospital administrative data to verify risk factors used in a risk-adjusted operative mortality model featured in a OSHPD public report which publishes hospital performance scores for coronary artery bypass graft surgery (CABG) in California. This paper also describes the importance of functional groups to organize SAS® code during report development. Finally a “peer-review” step-by-step process is described to obtain management and staff input to guide report development with minimal stress to all involved.

INTRODUCTION
TELL ME ABOUT THE PROBLEM
The usual scenario is that a manager wants new data to be featured in an Microsoft Excel report that was created using Dynamic Data Exchange (DDE). But why use DDE to create an Microsoft Excel report? The answer is comfort. Most people have a greater comfort level when they receive a report in Microsoft Excel because this format has been available for a very long time and most of us have used Microsoft Excel to view data, plot trends, and perform analyses. So once we see a report in Microsoft Excel we feel like a kid again with “instant playtime.” This comfort factor also carries into the SAS world where many years ago, when SAS graphics software was still in its infancy, most SAS users would export their data directly into Microsoft Excel for easy viewing, graphing trends, and simple data analysis. Then the user would return the data to SAS to use more sophisticated statistical tools to complete the analysis. Today this is no longer the case because SAS has many excellent tools to display and analyze your data.

The second reason this topic is important is that many organizations are transitioning from PC SAS to SAS Enterprise Business Intelligence (SAS EBI) (which contains SAS Enterprise Guide) where DDE is not compatible. OSHPD is the leader in collecting data and disseminating information about California’s healthcare infrastructure, promoting an equitably distributed healthcare workforce, and publishing valuable information about healthcare outcomes. Data which is reported to OSHPD is submitted in different formats and, once received, siloed in various repositories including SQL servers, SAS Integration Technologies server, local databases, and an Oracle enterprise data warehouse. Microsoft Office Suite, SAP Business Objects (BO), and SAS are used as query/reporting tools to access the data. For some time, OSHPD has needed a more streamlined and powerful system for data storage and utilization. The new approach had to support more complex queries, integrate data sources, eliminate duplication, provide up-to-date reporting via web tools, consolidate and maximize resources, leverage current investments, and integrate IT solutions with considerable flexibility. To this end, OSHPD decided to upgrade the existing data environment with SAS EBI. SAS EBI is a comprehensive, easy-to-use business intelligence software solution that integrates the power of SAS Analytics and SAS Data Integration to provide insights that power better decisions.

The focus of this paper was to build a custom designed report using a SAS programming language compatible with SAS EBI. The solution was to use ODS Report Writing technology which provides a programming language more powerful than DDE to easily position custom tables, headers, footers and note descriptions on the printed page. Before I show how this technology was used to create my custom designed report let’s take a brief tour of Dynamic Data Exchange and compare its features to ODS Report Writing technology.
**DYNAMIC DATA EXCHANGE**

Let’s learn the basics of Dynamic Data Exchange and compare its features to ODS Report Writing Technology.

Dynamic data exchange (DDE) is a powerful tool within SAS that programmers can use to send data stored in SAS data sets directly into formatted Microsoft Excel tables. The following overview is from Beal (2004); Kilburn (2009); Smith (2010); Vyverman (2001); and Vyverman (2002). Additional papers can be found by searching [http://www.sas.com](http://www.sas.com) or [http://www.lexjansen.com](http://www.lexjansen.com). Dynamic Data Exchange both reduces human error because the data are not hand entered into Excel where numerical precision can be lost, and saves time since the user need not format the table manually. DDE includes SAS code and macros for advanced SAS users that can be used to open Excel from SAS; center table titles across specified cells; bold, superscript, underline or italicize specific characters within individual cells; present numerical data in multiple formats such as scientific notation, commas or rounded to a specific precision; insert “page x of y” and date into the footer of the table; put borders around any column or row; dynamically assign rows and columns to insert data headings into the body of the table; load both numeric and character data into the same Excel column; specify font size and type; specify page setup options and margins; autofit specific columns; and create multiple worksheets within a single Excel file in a specified order with sheet names specified by the user.

DDE is a communication protocol available on Windows and OS/2 that enables some applications on these platforms (e.g., Microsoft Access, Excel, Word, FoxPro and others) to communicate with each other in a client/server fashion. The client application initiates a conversation with the server application and asks the server to perform a specific task. SAS (version 6.08 and later) can act as a DDE-client but not as a server. Most things that can be done manually in Excel can be automated from within a base SAS program using DDE. DDE enables a SAS session to take control of the Excel application and tell it precisely what to do. More information about the general idea of using DDE between SAS and DDE-compliant applications is found in the SAS Institute’s Technical Support document TS325 (SAS Institute, 1999). You can use DDE with the DATA step, the SAS macro facility, SAS/AF applications, or any other portion of SAS that requests and generates data. DDE has many potential uses, one of which is to acquire data from a Windows spreadsheet or database application. A search of the Microsoft web page for “macrofun” will reward you with the Microsoft-Windows “HLP” file “macrofun.hlp” that will provide all the Excel macro commands that you can use within the DDE statements.

There are several limitations to Dynamic Data Exchange. First DDE requires a template which defines the desired presentation or “style” for the report. This template contains a series of instructions to apply format properties, such as font size, font type, color, line width, etc., to each item in the report such as a data value, text label, table, header, or footer. The DDE program code reads the SAS data set, applies the template instructions, and places the result at a specific worksheet cell or other defined region. However it can be very time consuming to write DDE code. The effort to calculate the specific position or worksheet cell of each item in the report is very time consuming and laborious. The reward for your labor is that you get the desired look or “style” you wanted for the report. However these instructions are not easily edited when changes are needed. The position of each item on the page is relative and related to all other items on the page. Deleting a row or column, or moving one of the report items will alter the position of remaining items on that page. Also constructing a table consisting of a specific number of rows and columns limits the size of surrounding tables to the same dimension. ODS Report Writing technology is a much better alternative which does not require a template and gives you the ability to position custom designed tables, headers and footers of various sizes on the same page with ease.

Second, when you use DDE to create a report containing multiple worksheets, it was necessary to activate the next worksheet tab to view and print the next page of the report. Therefore printing all pages of the report requires manually changing the size of the printed region on each “worksheet” containing the next page of the report. ODS Report Writing technology creates a single “pdf” file from which all pages of the report can be printed without altering print region size.

Third, each page of a DDE report can only be viewed by activating each worksheet tab that contains the next page of the report. ODS Report Writing technology creates bookmarks which the user can use to quickly access each page of the report.

Finally it may not be desirable to distribute a report which is not protected from the user. Reports may contain confidential or sensitive information which should not be changed. ODS Report Writing technology will create a write protected PDF file which will secure the information. In this way the manager and users will view the report, and only the SAS programmer will alter report contents.

The next section gives more details about ODS Report Writing technology.

**ODS REPORT WRITING TECHNOLOGY**

ODS Report Writing technology is a programming language more powerful and easier to use than DDE. Recommended papers include O’Connor 2003; O’Connor 2008; O’Connor 2009a; O’Connor 2009b; Dorinski 2008; Herbison 2010; and Li 2010. Highlights from these papers are presented below. Others may be found...
Set Yourself Free – Use ODS Report Writing Technology In SAS Enterprise Guide, continued

at http://www.sas.com or http://www.lexjansen.com by searching on this topic. O’Connor (2009b) contains many examples which were the basis of a step-by-step process to build the custom designed report featured in this paper. This step-by-step process is presented in a later section below.

The ability to prepare complex reports and convey your message in a clear and concise manner is an absolute imperative in today’s sophisticated business environment. Clearly designed reports can enhance your organization’s credibility and reputation. DATA _NULL_ report writing has long been an integral part of the custom report writing offered by SAS®, but with this newly updated ODS Report Writing technology in SAS® 9.2, you have the ability to produce reports that you have only dreamed about. These new features will allow you to build custom data-centric reports in an easy-to-use object-oriented manner that is fully integrated with the ODS System. This technology is perfectly suited for creating custom invoices, inserting narrative descriptions in a table or document, creating form letters and non-rectangular reports, inserting custom subtotals, and it will address a variety of custom reporting needs.

The DATA step is an essential concept in SAS. The DATA step serves many purposes like creating SAS data sets, custom report writing, file management, and information retrieval. The ODS Report Writing Interface will exclusively focus on the custom report writing capabilities. The ODS Report Writing Interface is intended to fully embrace ODS features such as proportional fonts, traffic lighting, using colors, images, Unicode characters, while at the same time providing pixel perfect placement capabilities. This interface is not only fully integrated with all the capabilities of the ODS System, but also takes advantage of the rich programming features that the DATA step offers such as conditional logic, formatting capabilities, by-group processing, arrays, and a wealth of other features. The ODS Report Writing Interface is an object-oriented language that provides you with flexibility and control so that even the most rigid reporting requirements can be met with ease.

Before presenting the details of how I used ODS Report Writing technology to build my custom designed report we need to take a tour of the data problem that gave rise to the report.

METHODS

DATA SOURCE FOR THE REPORT

The California Coronary Artery Bypass Graft (CABG) Outcome Reporting Program (CCORP) is the largest public reporting program on CABG surgery outcomes in the United States. Each year OSHPD releases the *California Report on Coronary Artery Bypass Graft Surgery* which presents findings from analyses of data collected from California-licensed hospitals where surgeons performed adult isolated CABG surgery. This report presents risk-adjusted operative mortality to help evaluate hospital and surgeon performance. *The California Report on Coronary Artery Bypass Graft Surgery, 2007-2008 Hospital and Surgeon Data* can be found at [http://oshpd.ca.gov/HID/Products/Clinical_Data/CABG/index.html](http://oshpd.ca.gov/HID/Products/Clinical_Data/CABG/index.html).

CCORP reviews data submitted by each hospital for completeness and errors. The data discrepancy report compares the CCORP clinical data to OSHPD’s hospital administrative data source, the Patient Discharge Data (PDD). Hospitals are asked to review and account for discrepancies between the two data sources via patient medical chart review to verify that coding for the following ten data variables (risk factors) are consistent. Data variables include: 1) all CABG surgeries including status of isolated or non-isolated surgery; 2) Status of Isolated and Non-Isolated CABG surgery; 3) Resuscitation Prior to CABG Procedure; 4) CABG surgery deaths; and 6 postoperative complications including Stroke, Renal Dialysis, Prolonged Ventilation, Bleed/Tamponade, Infection, and Graft Occlusion.

It is important to verify that both data sources reported the same risk factor value because these risk factors will determine hospital and surgeon performance published in our public report. To make fair comparisons of care delivered by different healthcare providers, it is necessary to adjust for differences in severity of illness (case mix) of patients across providers. CCORP “levels the playing field” by considering the pre-operative condition of each patient. Providers that handle more complex cases receive a larger risk-adjustment weight in the risk model, and providers that handle less complex cases receive a smaller weight. Thus, hospitals and surgeons treating sicker patients are not at a disadvantage when their performance is compared with other surgeons and hospitals.

This paper will create a data discrepancy report containing data variable discrepancies at 122 California-licensed hospitals.

ORGANIZE YOUR SAS CODE

Before we start building SAS code we need to be organized. A project of this complexity requires designing a series of successive steps or functional groups. Each functional group is a collection of SAS program statements that create a product or series of products. A product is usually a SAS data set but also can be compiled SAS macros, parts of a report such as text notes, tables or other information. These steps are linear and sequential, so changes to any functional group which creates a new product will have consequences for subsequent functional groups that use that product. Given this complexity we must carefully design each functional group so that the final product or data
discrepancy report meets the needs of our organization. This is where peer-review is important but more importantly “orderly” and “low stress” peer-review. Management and staff will want to provide input and direction so we must include in report development “bus stops” or places where we can stop, review what we have done, make any changes, and then proceed onward.

It is always easier to build a project correctly the first time instead of taking everything apart somewhere in the middle, and doing it all over again. I learned this important lesson several years ago. I wanted to do a favor for a neighbor by building a complicated dresser with glass doors above, and a cabinet below using one of those kits where all of the wooden pieces are precut and they give you a bag of special wood screws. The challenge was the wood screws could only be used one time. So there was no going back and changing any part you had already assembled. When the task was done I discovered I reversed the backboard so the pretty finish was directed out instead of in toward the glass doors on top and toward the wooden cabinet doors below. There was no way to start over. Fortunately the neighbor and I were still friends for many years.

**Use Functional Groups**

First we identify all tasks that are needed to create the data discrepancy report. Second we group similar tasks into functional groups. For this report four functional groups were defined: a) Create the SAS data set containing sufficient information that will be used to count data variable discrepancies; b) Create notes and descriptive text used in the report; c) Create index variables which count data variable discrepancies; and d) Create the report.

Functional groups facilitate documenting program changes, and simplify troubleshooting efforts. When tasks are revised comments should be used to note changes to SAS program code. As a result troubleshooting efforts are simplified because we can track a task to a specific functional group where we instantly know why a specific change was made. Also if a task does not work correctly we know instantly where to find the SAS code that supports that task. Let’s look at the four functional groups in the data discrepancy report.

Functional group #1 creates a linked SAS data set containing CCORP clinical data and Patient Discharge Data (PDD) merged together on the basis of patient identifier information; and uses patient diagnostic, and procedure information to identify 10 data variables or risk factors featured in the data discrepancy report.

Functional group #2 creates text information used in various sections of the report including hospital help instructions, definition of data discrepancy variables, actions required by hospitals, headers, and footers.

Functional group #3 reads the SAS data set created in the first group and creates a series of index variables which count data variable discrepancies for each hospital.

Functional group #4 uses products created by the first three groups to generate the report.

Each year we begin with different versions of the CCORP clinical data and Patient Discharge Data (PDD). Data variable names in both files may change over time to incorporate new or updated information. Therefore we need to verify that the SAS program code is altered accordingly. Otherwise data variables in subsequent functional groups will be missing critical information.

Again, using functional groups simplifies troubleshooting efforts. If patient identification information is missing or incorrect we can quickly go to the first functional group. If notes or text is incorrect we go directly to the second group. If data discrepancy variables are missing or incorrect or the wrong hospital records displayed in the report we can go directly to the third group. If pages of the report are missing we can go directly to the fourth group.

Ok, now we need more detail about the peer-review process in the next section.

**Obtain Peer-Review During Project Development**

The main objective is to develop a work plan for including all report features required by the organization. In order to ensure report development is fun for all involved we need to create an orderly process whereby management provides input at designed “bus stops” where we can review the report, discuss possible enhancements, make any additional changes, and complete the report. Management wants to be involved in report development so we create “bus stops” for their input and direction.

Report features can be divided into functional and presentation. Functionality refers to collecting patient identifier or data discrepancy information required for the report. Presentation refers to word content; format properties of text such as font size, point size; location of tables or printed text, and use of colors for traffic lighting.

Here are the steps I used.
Step #1 Assemble the building blocks of the “prototype” report

This step includes collecting and assembling example code from multiple sources to create each and every feature in the report from tables, text, instructions to hospitals, headers, footers etc. As a result there will be multiple “sample” data sets used to create the report.

Step #2 Verify functionality and presentation of “prototype” report for a single source of “test” data

In this step we modify SAS program code to consolidate all “sample” data sets into one large “test” data set. Two items are being accomplished here. First we move each programming task from multiple sources of “sample” data to a single source of “test” data to verify how each report building task operates. Second we verify that each data variable and narrative text is displayed the same way. For example each date has the same SAS date format, each number has the same precision, and each text has the same font size.

Step #3 Verify functionality of “prototype” report using “live data” from previous report

Here we replace the “test” data set with a “live” data set from a previous report to verify the functionality of the report. If the report created with the old method and last year’s data has the same statistics as the report “prototype” also using last year’s data, we can be confident the report is functioning correctly and displaying the correct hospital records that show the correct data variable discrepancies.

Step #4 Verify presentation of “prototype” report in using “live data” from previous report

We know that the report functions correctly and displays the correct data discrepancies but we need to get peer review of report presentation. The report must display sufficient information to guide hospitals to review and edit each data variable discrepancy. This requires input from managers and others who have experience working with hospital staff. For example office staff wanted the report to list 24 diagnoses and 20 procedures instead of the 5 diagnoses and 5 procedures originally included in the report because this additional information will help hospitals compare CCORP clinical data to OSHPD’s hospital administrative data source, the Patient Discharge Data (PDD).

Step #5 Replace the “live data” from a previous report with “live data” for the current report

Our manager is already happy with the functionality and presentation of the report in Steps #3 and #4. In this step we create the report using current year’s data.

If data variable names in the CCORP clinical data set and the Patient Discharge Data (PDD) have not changed, the functionality and presentation of the report with the current year’s data should be perfect. However data names do change for many reasons so it is important to verify this fact by comparing PROC CONTENTS of the previous year’s data to current year’s data. If SAS cannot find a variable name it will continue to process code because it assumes that the programmer is always right and this variable is no longer needed. This misguided trust could have bad consequences. Also we need to verify the SAS log contains no errors, warnings, or notes which indicate errors in our program code that needs to be solved. The best way to do this is to compare each SAS Job Log for each functional group with the new SAS Job Log for that same group. Finally when I develop new SAS program code such as creating index variables to count data discrepancies I always verify that the program code is functioning correctly. Never assume your code is perfect but verify your SAS program code.

Remember as time marches forward the organization will want to change the report by adding data variable discrepancies, and changing functionality or presentation. If we maintain our functional groups and remember the above 6 steps the task will be much easier.

Now let’s look at an example of a hospital level report that was created using this approach.

RESULTS

EXAMPLE DATA DISCREPANCY REPORT

Let’s take a look at our Data Discrepancy Report for one of the hospitals and compare the old method of Dynamic Data Exchange and the new method of ODS Report Writing Technology. The first page of the report, which contains a list of instructions for the hospitals, is given for the old method (Display 1) and the new method (Display 2). Note the use of Microsoft Excel worksheet tabs in the former and Adobe bookmarks in the latter.

Our report in Microsoft Excel begins with a worksheet tab containing a list of instructions to the hospitals followed by ten worksheets for each data variable discrepancy section. Each data variable discrepancy section will contain multiple hospital records. The objective was to create a new section only if hospital records existed for that section. However, the DDE method was prone to errors and would exclude sections when they should have been included. This was most likely a programmer error by one of my predecessors. Instead of replacing the tire (fixing one problem) we decided to buy a new car (update the entire methodology)!
Display 1. Data Discrepancy Report in Microsoft Excel Showing Help Instructions.

The ODS Report Writing Technology version of the report also begins with a list of instructions to the hospitals followed by ten sections for each of the data variable discrepancies. Note the new method includes two bookmarks for each section. The first noted “Definitions” links to help text and presents the number of hospital records displayed in that section. The second bookmark noted “Hospital Level Data” links to the hospital records.

Using the new method the first bookmark “Definitions” is always displayed for each section. The second bookmark only appears if hospital records are displayed in that section. Sections IV, and X only have a single bookmark labeled “Definitions” because there are no hospital records to display.

A hospital record is displayed using the old method (Display 3). The first 8 rows notes the number of hospital records displayed in that section. Displays 4 and 5 show the two bookmarks “Definitions” and “Hospital Level Data” generated using the new method.

Remember that “possible under-reporting” notes risk factors present in the Patient Discharge Data (PDD) but not present in the CCORP clinical data. The reverse is true for “possible over-reporting” which notes risk factors present in the CCORP clinical data but not in the Patient Discharge Data (PDD).
Display 3. Data Discrepancy Report in Microsoft Excel Showing Hospital Record.

The presentation of patient identification information, diagnoses, and procedures using the old method is confusing and difficult to follow. You have to look carefully for several minutes before you can isolate the information you are looking for. Whereas new method provides a simpler presentation with the use of colors so you can quickly isolate the information you need.
Display 4. Data Discrepancy Report (New Method) Showing First Bookmark to Section Title Page.

PROGRAM CODE FOR DISCREPANCY REPORT

The data discrepancy report was created by 2,400 lines of SAS program code organized into the four functional groups described above. Unfortunately it is not possible to give a thorough review of this information in the present paper. My hope is to provide this review as topic for a WUSS paper in 2013. If you would like a copy of the full SAS program code please contact me. In addition I suggest you begin with the examples described by Daniel O’Connor (O’Connor 2009b). These examples will give you a “primer” and make it easier to understand the structure of the SAS programming code used in my paper. Let me know if you have any questions.
CONCLUSION

In today’s world where the ability to prepare complex reports and convey your message in a clear and concise manner is an absolute imperative in today’s sophisticated business environment, ODS Report Writing technology is a powerful alternative to Dynamic Data Exchange (DDE). In addition when many organizations are transitioning to SAS Enterprise Guide for their data collection, analytics, and data reporting needs ODS Report Writing technology can replace the use of DDE which is not compatible with SAS Enterprise Guide.

TRADEMARK CITATION

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.

ACKNOWLEDGEMENTS

The author would like to thank Daniel O’Connor at SAS Institute for sharing the paper O’Connor (2009b) and responding to my questions. In addition I could not have completed this project without my mentor Art Carpenter who was kind enough to share a “glimmer” of his great knowledge of SAS macros in his two workshops “Introduction to SAS Macros,” and “Advanced SAS Macros.” Completion of this project would not have been possible without the use of nested SAS macros to “bundle” the ODS Report Writing technology. Also for sharing with me a clever “snip-it” of SAS code to create a list of hospital license numbers and execute the report for each hospital on that list.

REFERENCES


CONTACT INFORMATION

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

Robert Richard Springborn, Ph.D.
Healthcare Outcomes Center
Office of Statewide Health Planning and Development
400 R Street, Suite 250
Sacramento, CA 95811-6213
Office (916) 326-3874

ROBERT.SPRINGBORN@OSHPD.CA.GOV