

VIEWS

International SAS Programmer Community



News, Hints, Tips and Information for SAS® Users

Issue 43, 3rd Quarter 2008

There will be a number of big SAS-related events coming up in October this year. The 2008 PhUSE Conference will be held in Manchester and the SAS Forum UK in Telford, where I will be very happy to meet any VIEWS members there. I have more articles from familiar sources again in this issue of *VIEWS News*: including some advice on using double ampersands in macros from Nina Werner, a tool from LeRoy Bessler to run a SAS process from Enterprise Guide in a disconnected batch mode that can be monitored, and some more tips on using regular expressions from the Amadeus Software Consultancy Team. Along with our regular sections, including some new and old SAS Formats, Options, and Functions, I hope you will find them all interesting and helpful.

If you would like to contribute an article, or discuss the possibility of doing so, please send an email to me at newsletter@views-uk.org. A list of possible subjects for your articles can be found on the *VIEWS* web site, and anyone is very welcome to add to that list by sending emails with your own questions to newsletter@views-uk.org.

Philip R Holland (Newsletter Editor)

Membership of *VIEWS* is free and you can register for notification of all future *VIEWS* events by emailing us at membership@views-uk.org, making certain you include your name and postal address. If you would like to receive email notification whenever a new issue of the free quarterly *VIEWS News* is published on the web site, please remember to also include your email address.

Ask The Consultant

This part of VIEWS News is where you can get your technical questions answered. Send your questions to the Editor.

Double Ampersand [&&] SAS Macro Variables

Q: How do you use macro variables with double ampersands?

A: [SAS 9] Compound-named macro variables can be created from SAS data set variables and resolved within a %DO...%END loop in a macro program.

Example 1. CALL SYMPUTX(var,value) in open SAS code, not in a macro:

```
DATA _NULL_;
  SET sashelp.class END = nd;
  CALL SYMPUTX('name' ||
               STRIP(PUT(_N_,2.)),name);
  IF nd THEN CALL SYMPUTX('_tot',PUT(_N_,2.));
RUN;
```

In **Example 1**, we populate macro variables based on SAS data set. **CALL SYMPUTX()** macro function can create macro variables from data set values in a DATA step. It has two parameters separated by a comma inside its parentheses. The first, a constant enclosed in quote marks, will be the name of the macro variable being created. The second is the value being assigned to that macro variable. The value of any macro variable will be a character string. This value will not be available until after the execution of this **DATA _NULL_** step, but is then available for the rest of our SAS session, since this macro variable resides in the **_GLOBAL_** symbol table. Ordinary conditional logic may be used in the DATA step to define when to set the macro variable, so **&_tot** is set only once by the second **CALL SYMPUTX()** when the DATA step reaches the end of **sashelp.class**, **_N_=19**.

Let's examine the first **CALL SYMPUTX()** statement in more detail. The first parameter, the macro name, is a compound construction consisting of the constant 'name' concatenated with a character representation of the automatic variable **_N_**, thus **namel** through **namel9**, since there are nineteen students in **sashelp.class**. The value assigned to each macro variable is the character string from the SAS data set variable 'name' in **sashelp.class**.

Example 2. Macro program using %DO loop for macro variables:

```
%MACRO _all_class;
%DO i = 1 %TO &_tot.;
  TITLE "&name&i.";
  PROC PRINT DATA=sashelp.class
    (WHERE=(name="&name&i."));
  VAR age height weight; RUN;
%END;
%MEND _all_class;

%_all_class;
```

In **Example 2**, we use the macro variables we created in **Example 1**. We want to perform the same program steps for each of the macro variables we created, so we write a macro program containing a %DO...%END loop which will iterate **&_tot** times. Remember that **&_tot** is equal to the number of students from our SAS dataset. The same way we concatenated the constant 'name' and **_N_** to create multiple macro variables, we now concatenate **&name** and our loop index **&i** inside a macro program. This only works inside a macro program because the macro pre-processor will run through the loop and write multiple iterations of SAS code to be executed later. As we increase **&i** from 1 to 19 in the loop, the double ampersand [&&] triggers the macro pre-processor to concatenate macro variable name components together (**&name**) with (**&i**), so that **&i** is first resolved to 1, then **&name1** is resolved to its value, and the name of student 1 = Alfred. We can run this code with **OPTIONS SYMBOLGEN** to see how the substitution is performed, or with **OPTIONS MPRINT** to see the lines of SAS code generated.

Partial output:

Alfred			
Obs	Age	Height	Weight
1	14	69	112.5
...			
William			
Obs	Age	Height	Weight
19	15	66.5	112

This process is very useful, for example, in writing reports for each of the seventy-two counties in the state of Wisconsin without having to submit seventy-two times, or remember how to spell all their names. If you have comments or questions, please don't hesitate to contact me.

Nina Werner, Madison, WI, USA

Taken from a presentation by Nina Werner at the 2008 Wisconsin Illinois SAS Users Conference.

Did You Know?

If you have a useful hint or tip, please send it to the Editor, and share it with the VIEWS membership.

SAS Batch from Enterprise Guide: Monitorability and Freedom

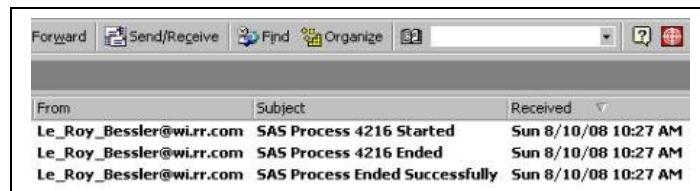
[SAS 9] The most common client interface to a SAS BI Server or SAS Enterprise BI Server is Enterprise Guide (EG). Enterprise Guide creates a packaged audit trail of code blocks that have been submitted, supplemented with the logs, and with links to any permanent data sets involved. This can be an advantage. However, you have no access to the SAS log to get an idea of what a long-running program might be doing, i.e. whether it is progressing, rather than being hung or looping. Also, if you want to run a program overnight or on the weekend, but must physically secure your laptop, you need a locked office.

The **SAS_EG_Batch** macro presented here enables you to use email and NotePad to monitor your program's progress, and lets you shut down your PC or laptop after launching the program, if desired. The macro, in effect, writes your SAS log to disk, writes your SAS report (a.k.a. "print") to disk, sends you a Started email, an Ended email, and a Completion Status email. The Started and Ended messages include the Windows process ID for your batch job, useful if a System Administrator needs to kill a runaway or hung SAS process. The Completion Status message provides the SAS task return code and its interpretation, as well as a link to the SAS log and, if one was created, a link to your SAS report. The SAS report is the default text listing. Both the SAS log and the SAS report are created with file extension *.txt, rather than *.log and *.lst, so that there is no automatic launch of SAS when you open them. The SAS log can be opened for read even when the program is still running, disconnected from EG.

Separate messages are created for Ended and Completion Status because the process ID is available only to your running program, but the task return code is available only to the EG macro-generated code that launched your program. The Started and Ended messages are sent by code that is concatenated before and after your actual program code. This three-part concatenation is submitted and run in batch as one program by **SAS_EG_Batch**.

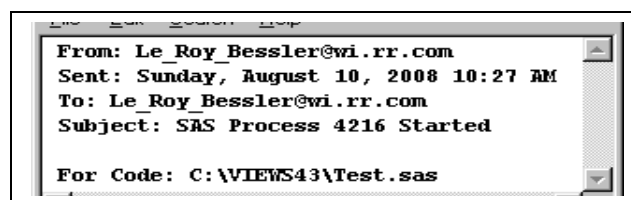
The SAS log and report filenames are based on the filename of your SAS program (CodeFile). All output (except WORK) is written to your SASUSER folder. All temporary files (also written to SASUSER) created and used by **SAS_EG_Batch** are automatically deleted when your program has completed. The log and report filenames can optionally be created with a datetime suffix. Your WORK data library is the same folder where a sub-folder is always created for your EG sessions. Your disconnected batch job does not use the sub-folder of the EG session that launched it.

Here is a picture of an email InBox:



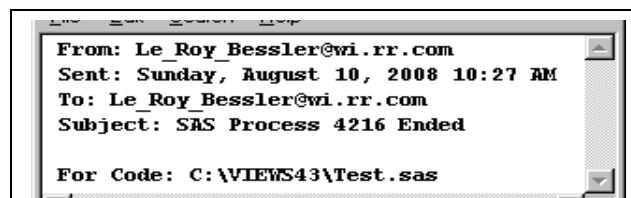
From	Subject	Received
Le_Roy_Bessler@wi.rr.com	SAS Process 4216 Started	Sun 8/10/08 10:27 AM
Le_Roy_Bessler@wi.rr.com	SAS Process 4216 Ended	Sun 8/10/08 10:27 AM
Le_Roy_Bessler@wi.rr.com	SAS Process Ended Successfully	Sun 8/10/08 10:27 AM

Here are pictures of the messages:



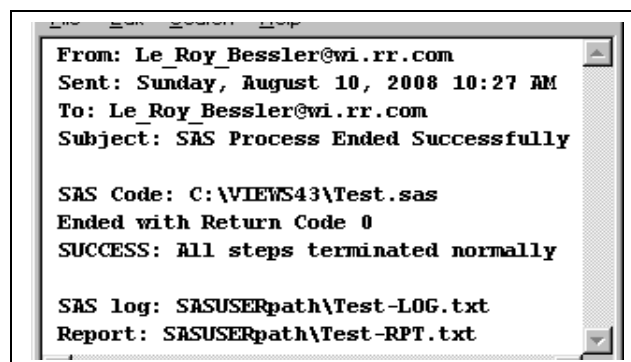
From: Le_Roy_Bessler@wi.rr.com
Sent: Sunday, August 10, 2008 10:27 AM
To: Le_Roy_Bessler@wi.rr.com
Subject: SAS Process 4216 Started

For Code: C:\VIEWS43\Test.sas



From: Le_Roy_Bessler@wi.rr.com
Sent: Sunday, August 10, 2008 10:27 AM
To: Le_Roy_Bessler@wi.rr.com
Subject: SAS Process 4216 Ended

For Code: C:\VIEWS43\Test.sas



From: Le_Roy_Bessler@wi.rr.com
Sent: Sunday, August 10, 2008 10:27 AM
To: Le_Roy_Bessler@wi.rr.com
Subject: SAS Process Ended Successfully

SAS Code: C:\VIEWS43\Test.sas
Ended with Return Code 0
SUCCESS: All steps terminated normally

SAS log: SASUSERpath\Test-LOG.txt
Report: SASUSERpath\Test-RPT.txt

In the message above, SASUSERpath would be whatever was retrieved by **SAS_EG_Batch** as your server's path for your SASUSER folder. It is shown here as above for generality.

Below are the **SAS_EG_Batch** macro and a sample invocation, and the **JobMsg** macro used by **SAS_EG_Batch**:

```
%MACRO JobMsg(StartOrEnd=) ;
DATA _NULL ;
LENGTH line $256;
FILE "&Path.\Job&StartOrEnd..sas" LRECL=256;
line = 'FILENAME SASMAIL EMAIL' ||
' TO="' || "&Notify" ||
' FROM="' || "&Notify" ||
' SENDER="' || "&Notify" ||
' SUBJECT="SAS Process ' ||
' &sysjobid' ||
' &StartOrEnd.ed" || ' ';;
```

```

PUT line $256.;
line = 'DATA _NULL_';
PUT line $256.;
line = 'file SASMAIL';
PUT line $256.;
line = "PUT 'For Code: " || "&Code.'";
PUT line $256.;
line = 'run;';
PUT line $256.;
RUN;
%MEND JobMsg;

%MACRO SAS_EG_Batch(Folder=,CodeFile=,
    Notify=,DateTime=NO);

%LET Code = &Folder.\&CodeFile..sas;

%LET DateTime = %UPCASE(&DateTime);

%LET Path = %SYSFUNC(PATHNAME(sasuser));

%LET Work =
    %SUBSTR(%SYSFUNC(PATHNAME(work)),1,
    %EVAL(%INDEX(%SYSFUNC(PATHNAME(work)),_
        - 2)));

PROC FORMAT LIB=work;
VALUE $TaskRc
'0'='SUCCESS: All steps terminated normally'
'1'='WARNING: SAS System issued warning(s)'
'2'='ERROR: SAS issued error(s)'
'3'='INFORMATIONAL: ABORT statement'
'4'='FATAL: ABORT RETURN statement'
'5'='FATAL: ABORT ABEND statement';
RUN;

DATA _NULL_;
LENGTH DateTime $16 Time $6;
Time = TRIM(LEFT(COMPRESS(PUT(TIME(),
    TIME8.),':')));

IF LENGTH(Time) EQ 5
    THEN Time = '0' || LEFT(Time);
DateTime = 'D' ||
    TRIM(LEFT(PUT(TODAY(),YYMMDDN8.))) ||
    'T' || Time;
CALL SYMPUT('dt',DateTime);
RUN;

%IF &DateTime EQ YES
    %THEN %LET FileSuffix = -&dt;
    %ELSE %LET FileSuffix = %STR();

%LET Bat =
    &Path.\&CodeFile.-BAT&FileSuffix..bat;
%LET Run =
    &Path.\&CodeFile.-RUN&FileSuffix..sas;
%LET Log =
    &Path.\&CodeFile.-LOG&FileSuffix..txt;
%LET Rpt =
    &Path.\&CodeFile.-RPT&FileSuffix..txt;

%JobMsg(StartOrEnd=Start);
%JobMsg(StartOrEnd=End);

FILENAME concat2 ("&Path.\JobStart.sas"
    "&Code"
    "&Path.\JobEnd.sas");

DATA _NULL_;
INFILE concat2;
INPUT;
FILE "&Run";
PUT _INFILE_;
RUN;

```

```

X "del &Path.\JobStart.sas" NOXWAIT NOXSYNC;
X "del &Path.\JobEnd.sas" NOXWAIT NOXSYNC;

DATA _NULL_;
LENGTH line $ 1024;
FILE "&Bat" LRECL=1024;
line =
    '"C:\Program Files\SAS\SAS 9.1\sas.exe"' ||
    ' -sysin "' || "&Run" || '"' ||
    ' -work "' || "&Work" || '"' ||
    ' -log "' || "&Log" || '"' ||
    ' -print "' || "&Rpt" || '"';
PUT line $1024.;
line = 'EXIT';
PUT line $1024.;
RUN;

SYSTASK COMMAND "&Bat" WAIT STATUS=TaskRC;

FILENAME SASemail EMAIL TO="&Notify"
    FROM="&Notify" SENDER="&Notify" SUBJECT=
%IF &TaskRC EQ 0 %THEN %DO;
    "SAS Process Ended Successfully";
%END;
%ELSE %DO;
    "SAS Process Ended with Return Code &TaskRC";
%END;

DATA _NULL_;
LENGTH TaskRC $3 HowEnded $50;
FILE SASemail;
PUT "SAS Code: &Code";
PUT "Ended with Return Code &TaskRC";
TaskRC = "&TaskRC";
IF TaskRC LE '5'
    THEN HowEnded = PUT(TaskRC,$TaskRC.);
    ELSE HowEnded = 'FATAL: ABORT ABEND ' ||
        TaskRC || ' statement';
PUT HowEnded;
PUT ' ';
PUT "SAS log: &Log";
IF FILEEXIST("&Rpt") EQ 1
    THEN PUT "Report: &Rpt";
RUN;

DATA _NULL_;
x = SLEEP(1);
RUN;

X "del &Run" NOXWAIT NOXSYNC;
X "del &Bat" NOXWAIT NOXSYNC;

%MEND SAS_EG_Batch;

%SAS_EG_Batch(Folder=C:\VIEWS43, CodeFile=Test,
    Notify=Le_Roy_Bessler@wi.rr.com);

```

Usability Note: If the EMAILHOST that your SAS server uses does not block the destination, SAS_EG_Batch can route the messages to **any** email address, so that remote job monitoring is possible.

It would be possible to enhance the macro to specify a report filename other than using the CodeFile filename and/or to provide the option of writing the report to an ODS destination.

However, a power user of ODS would probably prefer to build that packaging inside the program itself to gain custom control. An enhanced macro could also allow specification of a preferred target folder for the log and default text report. Besides supporting a Completion Status message to the person executing SAS_EG_Batch, either the macro could be enhanced to support report distribution, or that could be embedded in the program.

SAS Tip: Though not yet documented by SAS, the **FILEEXIST()** function can be used to test for the existence of a Windows folder, not only for the existence of a file.

In SAS 8.2, the EMAIL engine does not support **SENDER=**. I believe the rest of the **SAS_EG_Batch** macro to be compatible.

It might be possible to create a variation of **SAS_EG_Batch** to use in Display Manager, but I have not had a reason to do so. If anyone develops such code, I would be interested to see it. If interested in a more complex edition of **SAS_EG_Batch**, please request the code via email.

For other information about SAS batch, not in the Enterprise Guide context, please see “More Than Batch – A Production SAS Framework” by Denis Cogswell at:

www2.sas.com/proceedings/sugi30/021-39.pdf.

LeRoy Bessler, Assurant Health

PRX 3: Repeat counts, and locating matches

[SAS 9] Repeat counts are one of the most important components of PRX regular expressions. The list of repeat specifiers looks like this (although it is not quite complete, there are some very useful undocumented features in this area, which will be covered in a later article).

- * means “any number of times”. Warning: “any number” includes zero!
- + means “one or more times”.
- ? means “zero or one times”.
- {n} means “exactly n times” (n being a positive integer).
- {n,} means “n or more times”.
- {m,n} means “between m and n times”.

The repeat count always follows the character (or character class, or sub-expression) it refers to.

For example, an expression that would match all standard UK postcodes (while not validating them very strictly) is:

```
"/[A-Z]{1,2}\d{1,2}[A-Z]?\s+\d[A-Z]{2}/"
```

meaning “either one or two letters, then either one or two digits, then optionally another letter, then one or more white space characters followed by a digit and exactly two letters”.

In the absence of any anchors (such as “^” and “\$” which tie our expression to the beginning and/or end of the input string), **PRXMATCH()** scans the entire input string looking for a match for its regular expression. The rules are:

- It will return the earliest match it can find.
- Of matches beginning equally early in the string, it will return the longest.

In our previous examples we have been using the **PRXMATCH()** function, which returns the position of the match found (or 0 if none). This time we will use instead the **CALL PRXSUBSTR()** routine, which does exactly the same as **PRXMATCH()** except that it also returns the length of the matching sub-string.

```
DATA test;
  LENGTH instr $40;
  INPUT instr $ 1-40;
  rx = PRXPARSE("/((C.*T)|(D.*G))/");
  CALL PRXSUBSTR(rx, UPCASE(instr), p, ln);
  IF p > 0 THEN
    PUT "Match of length " ln
      " found at position " p " in " instr;
```

```
ELSE PUT "No match found in " instr;
  DATALINES;
Raining cats and dogs.
Cat on a hot tin roof.
There's a catch here, dang it!
Hodge was Dr Johnson's cat.
RUN;
```

The output is:

```
Match of length 3 found at position 9 in
Raining cats and dogs.
Match of length 14 found at 1 in Cat on a hot
tin roof.
Match of length 19 found at position 11 in
There's a catch here, dang it!
Match of length 2 found at position 3 in
Hodge was Dr Johnson's cat.
```

Recall that “[” means “or”, so we are looking here for a string that *either* begins in C and ends in T, *or* begins in D and ends in G. In the first example the “cat” comes before the “dog”, so it is the match returned. In the second example, the longest of the matches beginning with the C of “cat” is that one that ends with the T of “tin”. Something similar happens with the third example. In the fourth example, our matching string is the “dg” of “Hodge”, recall that the “*” means it’s OK to have *zero* characters between the D and the G. This time there is a longer match, “cat”, appearing later in the string, but this is ignored.

Amadeus Software Consultancy Team

PRX 4: Analysing matches - PRXPAREN and PRXPOSN

[SAS 9] Our “cats and dogs” example above showed how to check whether our input string matched either of two alternatives. We saw how the **CALL PRXSUBSTR()** routine could tell us exactly what sub-string of the input gave the match. But **CALL PRXSUBSTR()** did not actually tell us which of the two alternatives we had found a match for. There is another function that can do this: **PRXPAREN()**.

```
DATA test;
  LENGTH instr $60;
  INPUT instr $ 1-60;
  rx = PRXPAREN("/(bird)|(plane)|(superman)/");
  CALL PRXSUBSTR(rx, instr, p, ln);
  IF p > 0 THEN DO;
    PUT "String: " instr;
    paren = PRXPAREN(x);
    PUT "Matched parenthesis " paren;
  END;
  ELSE PUT "No match found in " instr;
  DATALINES;
When a new planet swims into his ken - Keats
I teach you the superman - Nietzsche
The dromedary is a cheerful bird - Belloc
RUN;
```

The output is:

```
String: When a new planet swims into his ken -
Keats
Matched parenthesis 2
String: I teach you the superman - Nietzsche
Matched parenthesis 3
String: The dromedary is a cheerful bird -
Belloc
```

Matched parenthesis 1

Parts of the expression that are enclosed in parentheses, like “(bird)” and “(plane)” above, are called *sub-expressions*. The **PRXPOSN()** function can tell us what sub-string of the input matched a particular sub-expression. In our example, we will use a useful piece of new syntax: “\b” indicates a “word boundary”. “\B”, of course, means “not a word boundary”.

```
DATA test;
  LENGTH instr $60 p1-p3 $20;
  INPUT instr $ 1-60;
  rx=PRXPARSE ("/ (Mrs?) ?.* (\b[A-Z] [a-z]+\b) " ||
    ".* (\b[A-Z] [a-z]+\b) /");
  IF PRXMATCH(rx, instr) > 0 THEN DO;
    PUT "String: " instr;
    p1 = PRXPOSN(rx, 1, instr);
    p2 = PRXPOSN(rx, 2, instr);
    p3 = PRXPOSN(rx, 3, instr);
    PUT p1= p2= p3=;
  END;
  ELSE PUT "No match found in " instr;
  DATALINES;
Mr John Q Public
Mrs Jane Public
James James Morrison Morrison Wetherby George Dupree
Attila the Stockbroker
Norman StJohn Stevas
RUN;
```

The output is:

```
String: Mr John Q Public
p1=Mr p2=John p3=Public
String: Mrs Jane Public
p1=Mrs p2=Jane p3=Public
String: James James Morrison Morrison Wetherby
George Dupree
p1= p2=George p3=Dupree
String: Attila the Stockbroker
p1= p2=Attila p3=Stockbroker
String: Norman StJohn Stevas
p1= p2=Norman p3=Stevas
```

Our expression contains three sub-expressions. The first (for which the matching sub-string is stored in **p1**) matches “Mr” or “Mrs”. The first “?” says that the “s” is optional; the second says that the whole sub-expression is optional. The others (**p2** and **p3**) each match a name, consisting of an upper-case letter followed by one or more lower-case letters, and both beginning and ending at a word boundary. Since the PRX functions return (by default) the longest possible matches, the names returned in **p2** and **p3** are the last two in the string. Neither “the” nor “StJohn” gives a match, the first because it is not capitalised and the second because of the capital “J” in the middle.

Amadeus Software Consultancy Team

Formats, Options, and Functions

This section is devoted to the description of useful, or unusual, SAS Formats, Options, and Functions.

[SAS 9] The **PERCENTw.d** format is well known and well used, but will only display negative percentage values in parentheses. The **PERCENTNw.d** format, displays negative percentage values with a minus sign in front.

Everyone will recognise these data set options, but, surprisingly, they have never appeared in this section before now:

<http://www.views-uk.org>

KEEP= specifies the variables to keep in an input or output data set, and **DROP=** specifies the variables to drop from an input or output data set. These options can be very important to reduce the size and processing time of data sets, especially when they have large numbers of variables and/or rows.

[SAS 9] The **STRIP()** function has the same functionality as the **TRIM(LEFT())** combination of functions, but with less typing and fewer parentheses to forget!

PhUSE News

2008 PhUSE Conference

The 2008 PhUSE Conference will be held from 12-15 October 2008, Midland Hotel, Manchester, UK

Following a very successful Paper Review Meeting, we are delighted to publish the full list of papers which will be presented at the Conference in Manchester. Across the 13 streams, a record 123 papers were submitted. The full list of presentations are by 84 authors from 47 companies across 13 countries. Visit www.phuse.eu/Conference-2008.aspx for the list of papers and for other important conference information.

As well as the vast number of papers, we will have two panel discussions at this year's conference. One will be at the end of what promises to be a very thought provoking CDISC Stream, the other will be at the end of the conference and will conclude this year's theme, "Working in a Global Environment". More information about the closing panel discussion and the key note speaker will be released before the conference.

Please complete the conference registration by visiting www.phuse.eu/registration2008.aspx. We look forward to welcoming you to what will be an excellent conference.

Olivier Leconte and Benjamin Szilagyi
2008 PhUSE Conference Chair and Co-chair

PhUSE Single Day Events in Basel & London

PhUSE held 2 “Best of the Best” single day events at Novartis in Basel on 28 May 2008 and at the Royal Statistical Society in London on 11 June 2008, for papers and presenters that had won “best in stream” prizes at the annual PhUSE conferences in Heidelberg, Dublin or Lisbon.

Amadeus Software, Magnus Mengelbier and Phil Holland presented on both days along with several other presenters on each day, where SQL was a prominent topic for the agenda items. The events were both well attended and received good feedback from the 80+ Basel and the 40+ London attendees.

Philip R Holland (Newsletter Editor)

News

2008 SAS UK Special Interest Groups

The 2008 SAS UK Special Interest Groups (SIGs) meeting was held, as usual, at SAS UK in Marlow. This year saw the biggest turnout yet, with over 200 registrations, and an estimated attendance of about 150 delegates.

The delegates came from a wide variety of industries, and this was reflected in the wide range of content of the presentations across the six SIGs. SAS presentations were focused on informing about, and demonstrating new features of, new

http://www.sascommunity.org/wiki/VIEWS_News

products. The event ended with a generous and well organised barbecue lunch provided by SAS, which was a perfect opportunity for further networking with other SAS Professionals. A detail agenda of the whole meeting can be found at the SIGs web site:

www.sas.com/uk/events/sigs/agenda.html

This year, the presentation slides will be available for download through the new UK Social Network for SAS Professionals at sasprofessionals.ning.com.

Methodios Typou, Aeolus Consultancy

In Brief

- Back issues of *VIEWES News* are available from the *VIEWES* web site, and also from the *VIEWES News* page at:
www.sasCommunity.org/wiki/VIEWES_News
- Phil Mason now has a SAS tips newsletter, which can be requested by going to www.woodstreet.org.uk and then clicking on Tips. Previous tips can also be found on the web site.
- More SAS hints and tips can be found on the SAS FAQ pages www.hollandnumerics.com/sasfaq and also on the Amadeus Software Tips pages at www.amadeus.co.uk/TipsList.aspx?ST=1.
- Looking for a SAS-related job? Try the following web pages: www.hollandnumerics.com/careers and www.amadeus.co.uk/Careers.aspx.
- You are invited to join the SAS Professionals social networking group, described as the “365 Day SIG” for SAS Users in the UK, at:
sasprofessionals.ning.com
- Phil Holland has also started a group page on Squidoo for other pages related to SAS, and would like to make it a central resource for SAS hints, tips, links and books, at:
www.squidoo.com/groups/SAS_users

Diary

*Are you organising an event that would be of interest to the *VIEWES* readership? Let us know as we are interested in all non-profit making events related to SAS.*

September 2008

11 **How-To Seminar for SAS Programmers**, Oxfordshire

See the Amadeus Software web site www.amadeus.co.uk for further details.

October 2008

12-15 **PhUSE Conference**, Manchester

The Pharmaceutical Users Software Exchange (PhUSE) conference is the premier programming event for the pharmaceutical industry in Europe. See www.phuse.eu for more details.

22 **SAS9 Business Intelligence-Delivered**, Oxfordshire

See the Amadeus Software web site www.amadeus.co.uk for further details.

21-22 **SAS Forum UK**, Telford, Shropshire

The SAS Forum UK is held each year and focuses on the latest SAS software releases. For more details please go to www.sas.com/uk/events/sasforum.

November 2008

4 **New and Enhanced Features of SAS 9.2**, Oxfordshire

See the Amadeus Software web site www.amadeus.co.uk for further details.

20 **How-To Seminar for SAS Programmers**, Oxfordshire

See the Amadeus Software web site www.amadeus.co.uk for further details.

December 2008

11 **SAS9 Business Intelligence-Delivered**, Oxfordshire

See the Amadeus Software web site www.amadeus.co.uk for further details.

March 2009

22-25 **SAS Global Forum**, National Harbor, MD, USA

The SAS Global Forum (SGF, formerly called SUGI) is held each year and focuses on the technical aspects of SAS software. See www.sasglobalforum.org for more details.

*All *VIEWES* event information will be posted and/or emailed to registered members of *VIEWES*, and will include an event application form. Please send your queries about any *VIEWES* events to event@views-uk.org, and don't forget to look at the web site for the latest news.*

Contacts

Amadeus Software	Email: info@amadeus.co.uk Tel: +44-(0)1993-878287 Web: www.amadeus.co.uk
LeRoy Bessler Assurant Health	Email: le_roy_bessler@wi.rr.com
Methodios Typou Aeolus Consultancy	Email: meth.typou@aeolusconsultancy.com
Nina Werner Madison, WI, USA	Email: nina_werner@mac.com
PhUSE	Email: office@phuse.eu Web: www.phuse.eu
SAS UK	Tel: +44-(0)1628-486933 Web: www.sas.com/uk
<u>VIEWS</u>	
Web Site	Web: www.views-uk.org
Chairman	Email: chair@views-uk.org
Newsletter Editor	Email: newsletter@views-uk.org
Membership secretary	Email: membership@views-uk.org
Events secretary	Email: event@views-uk.org

VIEWS News is published quarterly by VIEWS International SAS Programmer Community. VIEWS is a not-for-profit organisation. Our Mission Statement and Rules can be viewed on our web site together with a list of current committee members.