Fall 2009 OASUS Questions and Answers

The following answers are provided to the benefit of the OASUS Users Group and are not meant to replace SAS Technical Support. Also, an Enterprise Guide project is provided as a companion to this document. The project is available on the OASUS web site (www.oasus.ca) under the Spring 2010 meeting since this is when these answers have been presented formally.

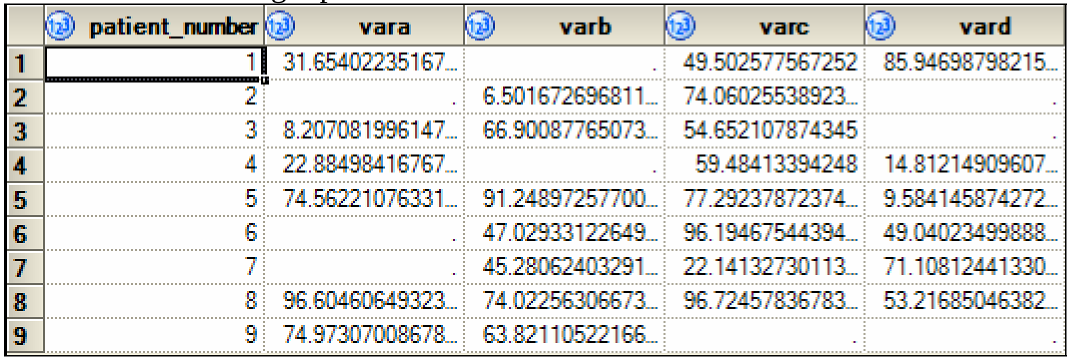
Question 1 : How to display unique list of variables with missing value per each unique record from a data set with more than 200 variables?

The intention here is to create a report that lists, for each record, the variables that contain a missing value.

The trick to solve this problem is to first transpose the data to get the variable names and the corresponding values on 2 different columns. A simple PROC PRINT report can then be run against the transposed data to get the desired report.

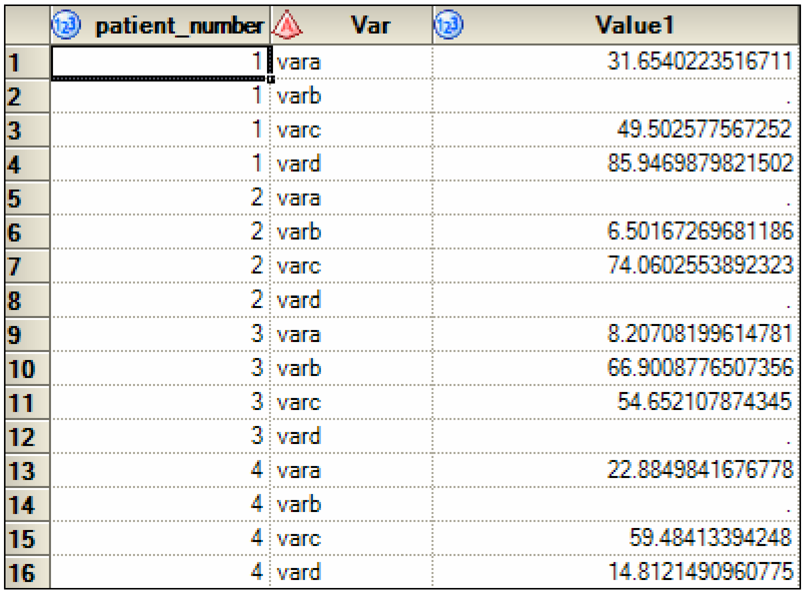
Step #1 Transpose the data

Convert the following input data:



	patient_number	vara	varb	varc	vard
1	1	31.65402235167…	.	49.502577567252	85.94698798215…
2	2	.	6.501672696811…	74.06025538923…	.
3	3	8.207081996147…	66.90087765073…	54.652107874345	.
4	4	22.88498416767…	.	59.48413394248	14.81214909607…
5	5	74.56221076331…	91.24897257700…	77.29237872374…	9.584145874272…
6	6	.	47.02933122649…	96.19467544394…	49.04023499888…
7	7	.	45.28062403291…	22.14132730113…	71.10812441330…
8	8	96.60460649323…	74.02256306673…	96.72457836783…	53.21685046382…
9	9	74.97307008678…	63.82110522166…	.	.

into the following transposed dataset with PROC TRANSPOSE:



	patient_number	Var	Value1
1	1	vara	31.6540223516711
2	1	varb	.
3	1	varc	49.502577567252
4	1	vard	85.9469879821502
5	2	vara	.
6	2	varb	6.50167269681186
7	2	varc	74.0602553892323
8	2	vard	.
9	3	vara	8.20708199614781
10	3	varb	66.9008776507356
11	3	varc	54.652107874345
12	3	vard	.
13	4	vara	22.8849841676778
14	4	varb	.
15	4	varc	59.48413394248
16	4	vard	14.8121490960775

using the following code:

```
PROC TRANSPOSE DATA=MISSING
      OUT=TransposedMISSING
      PREFIX=Value
      NAME=Var
      LABEL=Var
;
      BY patient_number;
      VAR vara varb varc vard;

RUN; QUIT;
```

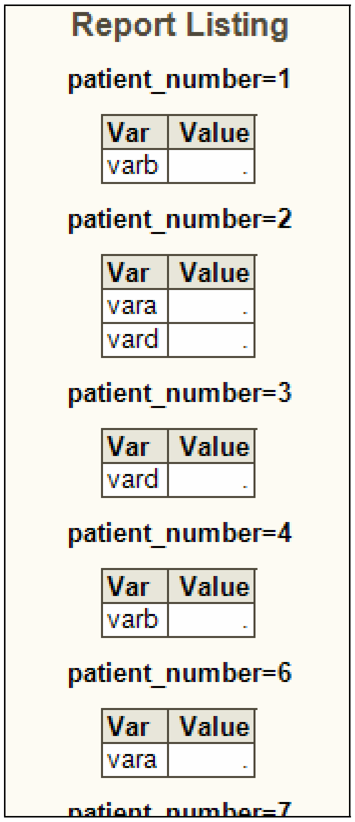
Step #2 Keep only the missing values:

```
PROC SQL;
   CREATE TABLE TransposedMISSING2 AS
   SELECT t1.patient_number,
          t1.Var LABEL='',
          t1.Value1 AS Value
      FROM TRANSPOSEDMISSING AS t1
      WHERE t1.Value1 IS MISSING;
QUIT;
```

Step #3 Generate the report

```
PROC PRINT DATA=TRANSPOSEDMISSING
      NOOBS
      ;
      VAR Var Value;
      BY patient_number;
RUN;
```

Final report:



Report Listing

patient_number=1

Var	Value
varb	.

patient_number=2

Var	Value
vara	.
vard	.

patient_number=3

Var	Value
vard	.

patient_number=4

Var	Value
varb	.

patient_number=6

Var	Value
vara	.

patient_number=7

Question 2 : Can you keep the original creation date when using data ; set; run;?

Normally, when you run a DATA step with an implicit OUTPUT statement, the output dataset gets re-created if it does already exist. Therefore, you lose the original date.

There is a way around, but it requires that you use the MODIFY statement in conjunction with the OUTPUT statement as followed:

Step #1 Remove all the observations from the original dataset using a SQL DELETE statement. This tep does not delete the dataset but simply empty it.

```
proc sql;
delete from MainTable;
run;
```

Step #2 Add observations to the original table using a DATA step with a MODIFY and an OUTPUT statement. Notice that the MODIFY statement never gets executed but is compiled. This forces the OUTPUT statement to add observations at the end of the DATA set, hence retaining the original creation date.

```
data MainTable;
   if (0) then modify MainTable;
   set OtherTable;
   output;
run;
```

It would possible to obtain similar results with either PROC APPEND or SQL INSERT statement. As usual, there are many ways to achieve a desired result using SAS.

Question 3 : Can you create a format that is based on 2 variables: a “regular” variable and a variable that represent a time interval?

This question relates to the fact that code sets may vary overtime. For example, in the case of hospital data, a code 3 for Operating Room could mean “surgery” before 2002, but “delivery” from 2002 to 2004 and “treatment” in 2005.

A SAS format can only be applied to one variable at a time. So, there is no way to indicate when creating a SAS format to use an auxiliary variable to determine what string to associate a given code with. However, it is possible to use two formats to derive a formatted value at runtime. It requires the use of the PUT and INPUTC functions:

- PUT function: using a format, this function converts a given numeric or character value into another character value using a specified format.

- INPUTC function: using an informat, this function converts a given character value into another character value at runtime.

The trick is to create a format that will determine an informat to use based on some auxiliary information (for example a date) and then derive the final formatted value into a new variable at runtime in a DATA step. Format and informat are similar in nature but they are used for different purposes:

- FORMAT: A format is an instruction that SAS uses to write data values from either a numeric variable or a character variable. The result of a format is always a character string.
- INFORMAT: An informat is an instruction that SAS uses to read data values into a variable. The result of applying an informat is either character or numeric depending if the informat is character or numeric. The input values are always character.

Let’s have a look at the whole process using an example, starting on next page.

Step #1 Create the necessary format and informats.

```
proc format ;

      invalue     $f2002h     '3'="surgery"
                              OTHER = "other";
      invalue     $f2003h     '3'="delivery"
                              OTHER = "other";
      invalue     $f2005h     '3'="treatment"
                              OTHER = "other";
      invalue     $f2006h     '3'="treatment"
                              OTHER = "other";

      value       qa          2001-2002='$f2002h'
                              2003-2004='$f2003h'
                              2005='$f2005h'
                              2006='$f2006h';
run;
```

In this example, the format is used to determine which informat to use based on a year. Each informat derives the appropriate code depending on the corresponding year.

Step #2 Create a new variable properly formatted with a DATA step.

```
data HospitalData;

      infile cards;
      length OpRoomf $100;
      input  @1 oproom $1 @4 Date mmddyy10.;
      format Date date9.;

      OpRoomf=inputc(put(OpRoom,$1.),put(Year(date),qa.));

cards;
3 01/10/2001
4 01/10/2001
3 01/10/2002
3 01/10/2003
3 01/10/2004
3 02/10/2005
;
```

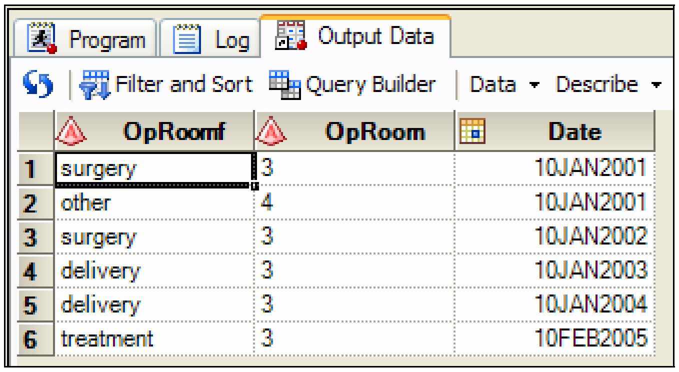
The important statement is the assignment statement that sets the OpRoomf variable which is used to store the formatted value. This statement uses the INPUTC and the PUT functions as followed:

- The PUT function uses the qa format previously created. It converts the Year variable into a string representing the informat to be used by the INPUTC function.
- The PUT is also use to convert OpRoom into a character string. This is required by the informat which can only be used to read character data.
- The INPUTC format takes the value of the OpRoom variable (converted into a character) and turned into a formatted value based on the informat identified.

As an example, if Oproom=3 and date=01/10/2002, the assignment takes place as followed (substitutions are marked in red):

```
1) OpRoomf=inputc(put(OpRoom ,$1.),put(2002,qa.));
2) OpRoomf=inputc(put(OpRoom,$1.), '$f2002h');
3) OpRoomf=inputc(put(3,$1.), '$f2002h');
4) OpRoomf=inputc('3', '$f2002h');
5) OpRoomf='surgery'
```

Here are the final results with the derived variable OpRoomf properly formatted:



	OpRoomf	OpRoom	Date
1	surgery	3	10JAN2001
2	other	4	10JAN2001
3	surgery	3	10JAN2002
4	delivery	3	10JAN2003
5	delivery	3	10JAN2004
6	treatment	3	10FEB2005

Question 4 : How to get rid of the background code in SAS EG? There are many things listed in the log before showing the code you submitted.

In releases prior to 4.2, there isn't any way to remove from the log the code that is added by Enterprise Guide.

However, in EG 4.2 many of the log entries can be eliminated by unticking the "Show generated wrapper code in SAS log" box under Tools | Options | Results | Results General.

Here's an explanation of what the remaining code does, and why it's useful. Once the user understands why it's needed, it doesn't seem as intrusive!

```
;*';*";*/;quit;run;
```

This odd looking line of code attempts to clear the SAS buffer. The first time you submit your code to a SAS server from EG, a SAS session is started which is kept active while your EG project is open (unless you choose to release the server). That's why you can use WORK datasets from one submission to another. Because of this, if you forget to finish a piece of SAS code with a *RUN;* or *QUIT;* the SAS session will try to incorporate the code in the next submission from EG as part of the previous step. This line tries to force the SAS session to finish the previous step and start a new one.

```
OPTIONS PAGENO=MIN;
```

Resets the page number to 1 for any reports created in the EG task.

```
%_eg_hidenotesandsource;
```

This SAS macro forces most of the log entries to be hidden.

```
%_eg_conditional_dropds(SASUSER.QUERY_FOR_SHOES);
```

EG uses PROC SQL to build queries and to prepare data for descriptive tasks. A quirk of the SQL standard is that if you try to create a table that already exists, instead of replacing it the SQL run will fail. The code above checks to see if the table named in the parameter exists, and if it does it deletes it.

```
PROC SQL;
 CREATE VIEW WORK.SORTTempTableSorted AS
     SELECT T.Stores
 FROM WORK.X as T
;
QUIT;
```

In some descriptive tasks, the data needs to be preprocessed before being analyzed. EG creates a view to apply the necessary processing, and then runs the descriptive task against the view. In other cases, like this one, nothing is needed but the view is created anyway. Go figure.

Your task code will usually appear next. After the task code, EG inserts code to hide most log entries, and to delete any temporary SQL tables and views that were created. Finally,

```
QUIT; RUN;
```

to try to make sure that the SAS server isn't still waiting for code in a step, and we're done!

If you're using an earlier release than 4.2, or if you don't choose to untick the "Show generated wrapper code in SAS log" box, you may also see the following entries before your task code:

```
%LET _CLIENTTASKLABEL='Query Builder';
%LET _CLIENTPROJECTPATH='C:\Doc...Desktop\Log check.egp';
%LET _CLIENTPROJECTNAME='Log check.egp';
```

Three macro variables are set; the type of task, the full path and name of the EG project, and the file name of the EG project. EG may use these variables, but you can use them in your SAS code also.

```
ODS _ALL_ CLOSE;
ODS PROCTITLE;
OPTIONS DEV=ACTIVEX;
GOPTIONS XPIXELS=0 YPIXELS=0;
FILENAME EGSR TEMP;
ODS tagsets.sasreport12(ID=EGSR) FILE=EGSR ...;
GOPTIONS ACCESSIBLE;
```

These lines set up the SAS GRAPH and ODS environment so that the SAS server knows how to format the results.

Your task code will usually appear next. After the task code, EG inserts code to restore the SAS GRAPH and ODS environments, and to reset the _CLIENT... macro variables.

Finally, another

```
;*';*";*/;quit;run;
```

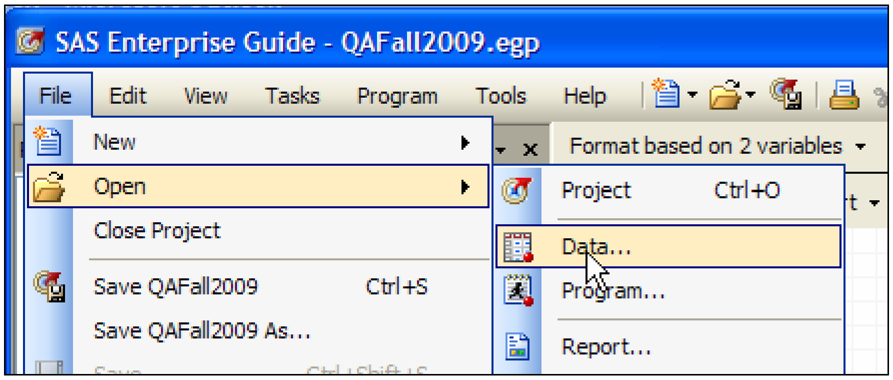
to try to make sure that the SAS server isn't still waiting for code in a step.

Question 5 : How do you read SAS data from a Unix server when you use Enterprise Guide?

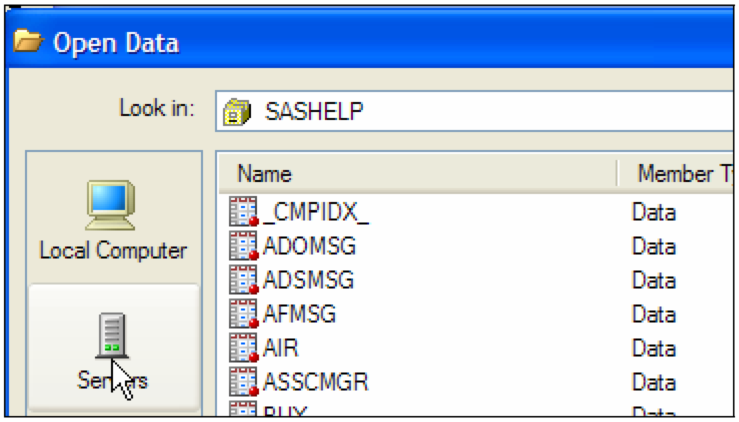
The Unix server must be defined in the profile that is active in Enterprise Guide.

Using the *Open* Dialogue

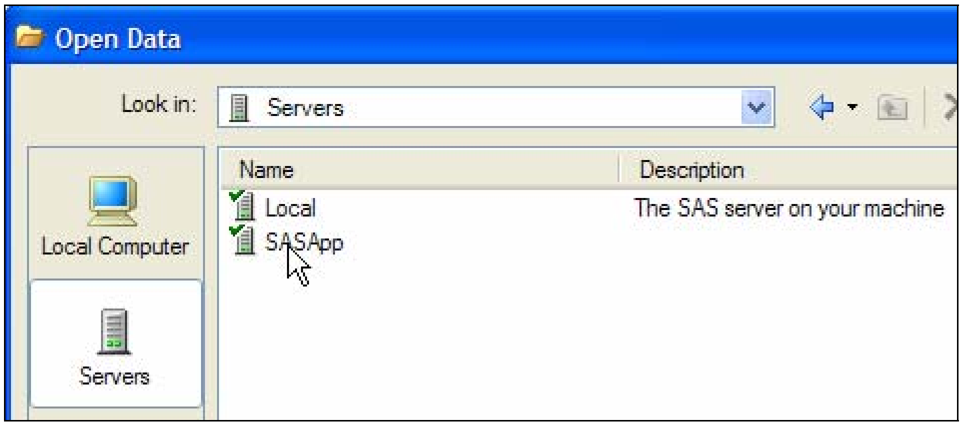
1. Select *`File`* | *`Open`* | *`Data...`*



2. Select *`Servers`*.

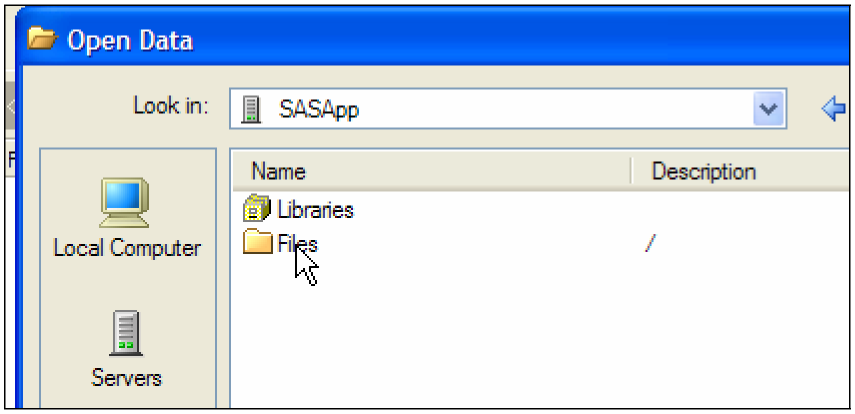


3. Double-click on your server.



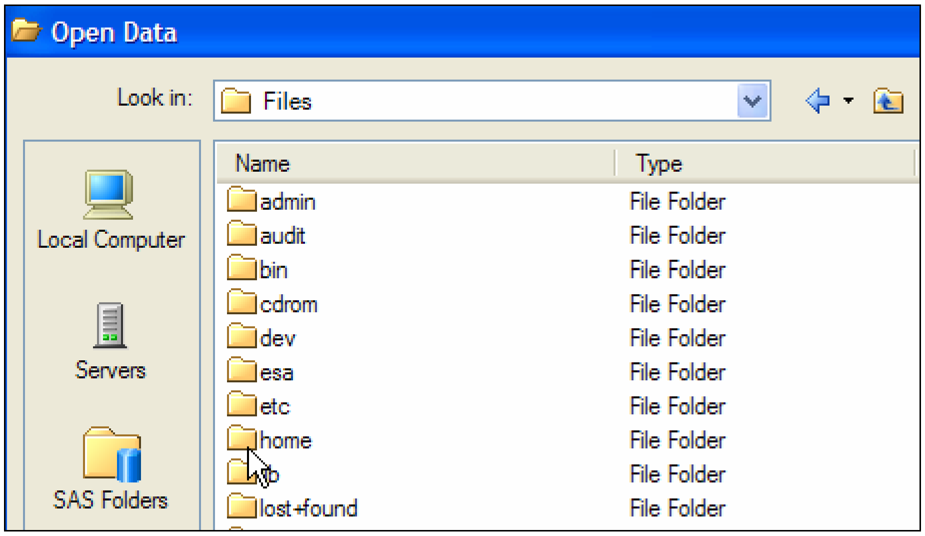
Name	Description
Local	The SAS server on your machine
SASApp	

4. Double-click on `Files`.



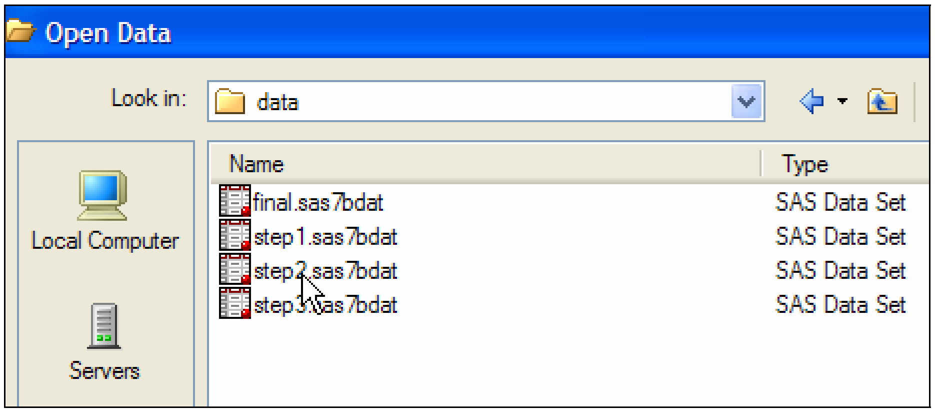
Name	Description
Libraries	
Files	/

5. Follow the directory tree to your file.



Name	Type
admin	File Folder
audit	File Folder
bin	File Folder
cdrom	File Folder
dev	File Folder
esa	File Folder
etc	File Folder
home	File Folder
lib	File Folder
lost+found	File Folder

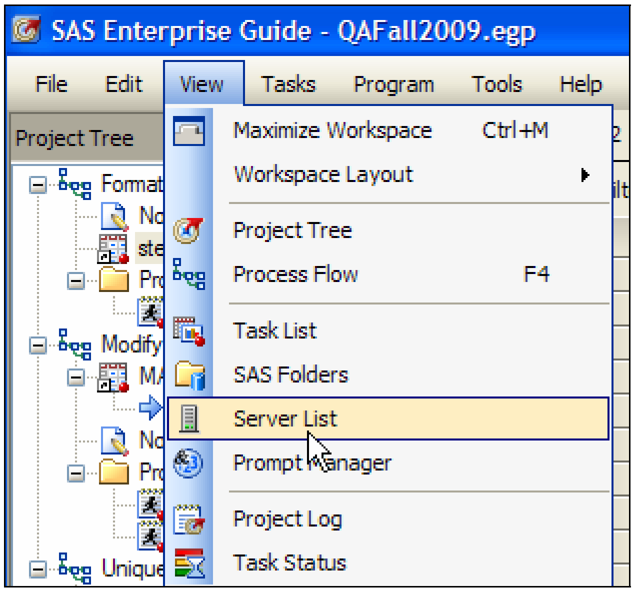
6. Double-click on the filename to open it.



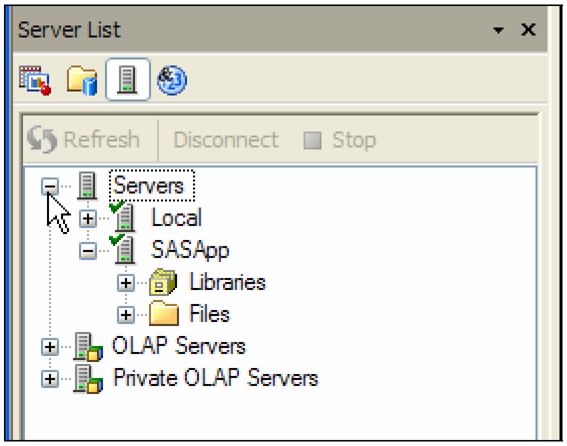
Name	Type
final.sas7bdat	SAS Data Set
step1.sas7bdat	SAS Data Set
step2.sas7bdat	SAS Data Set
step3.sas7bdat	SAS Data Set

Using the Server List

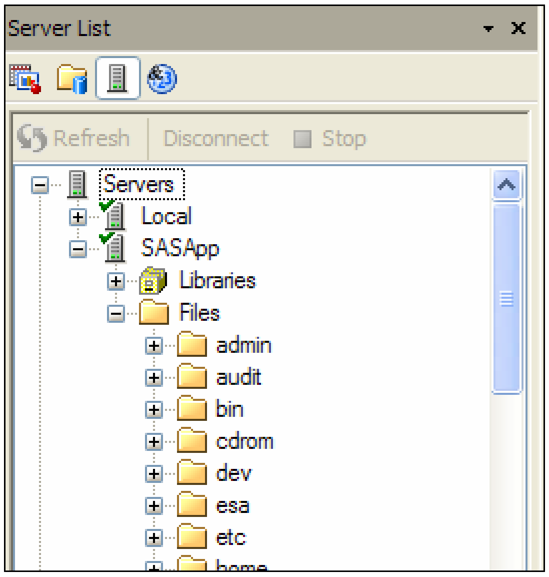
1. Select `View`| `Server List.`



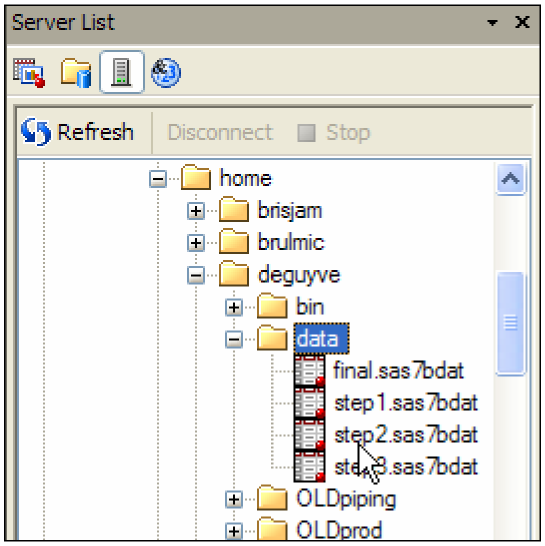
2. Expand `Servers`, expand your server, expand `Files.`



3. Follow the directory tree to your file.

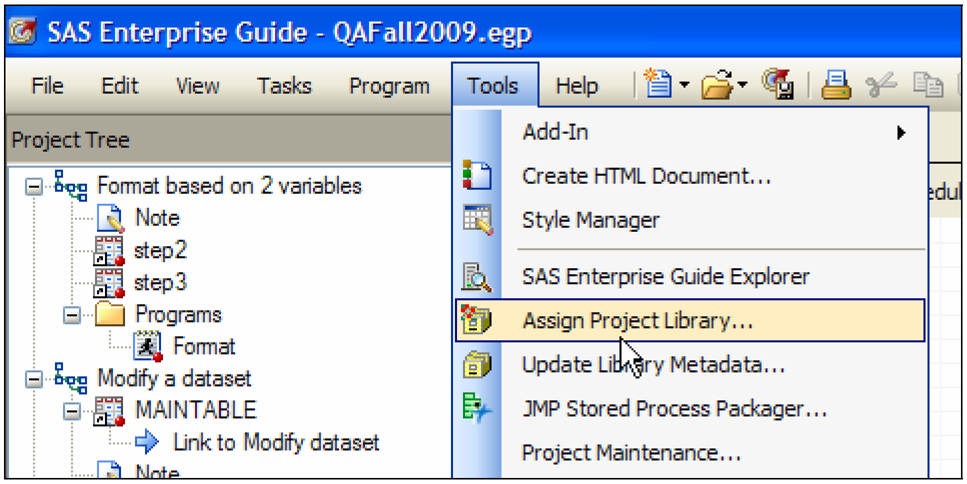


4. Double-click on the filename to open it.

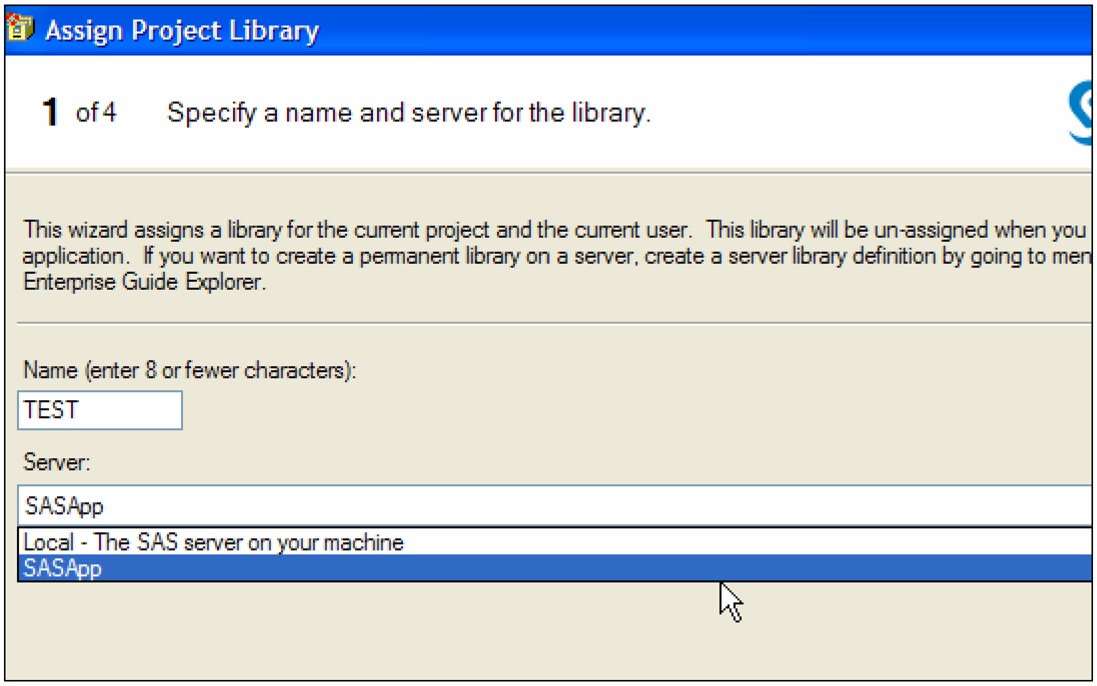


Using the *Project Library* dialogue

1. Select `Tools` | `Assign Project Library...`



2. On the first screen, enter a library name and select your server from the `Server:` pick list.



Assign Project Library

1 of 4 Specify a name and server for the library.

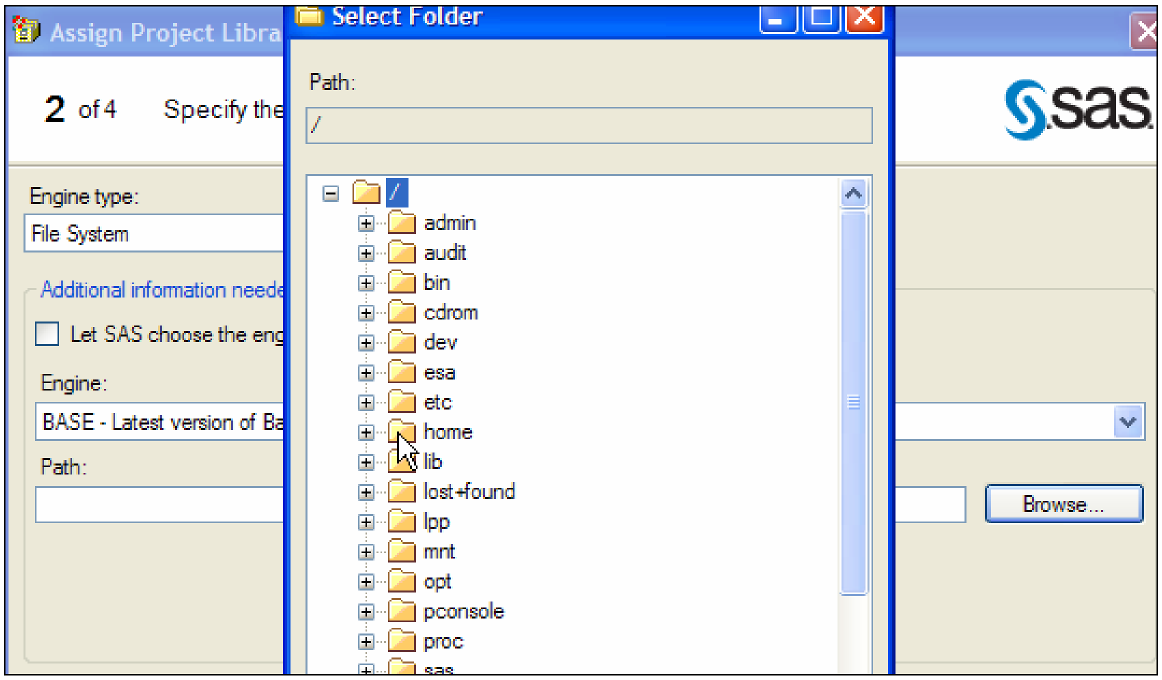
This wizard assigns a library for the current project and the current user. This library will be un-assigned when you application. If you want to create a permanent library on a server, create a server library definition by going to men Enterprise Guide Explorer.

Name (enter 8 or fewer characters): TEST

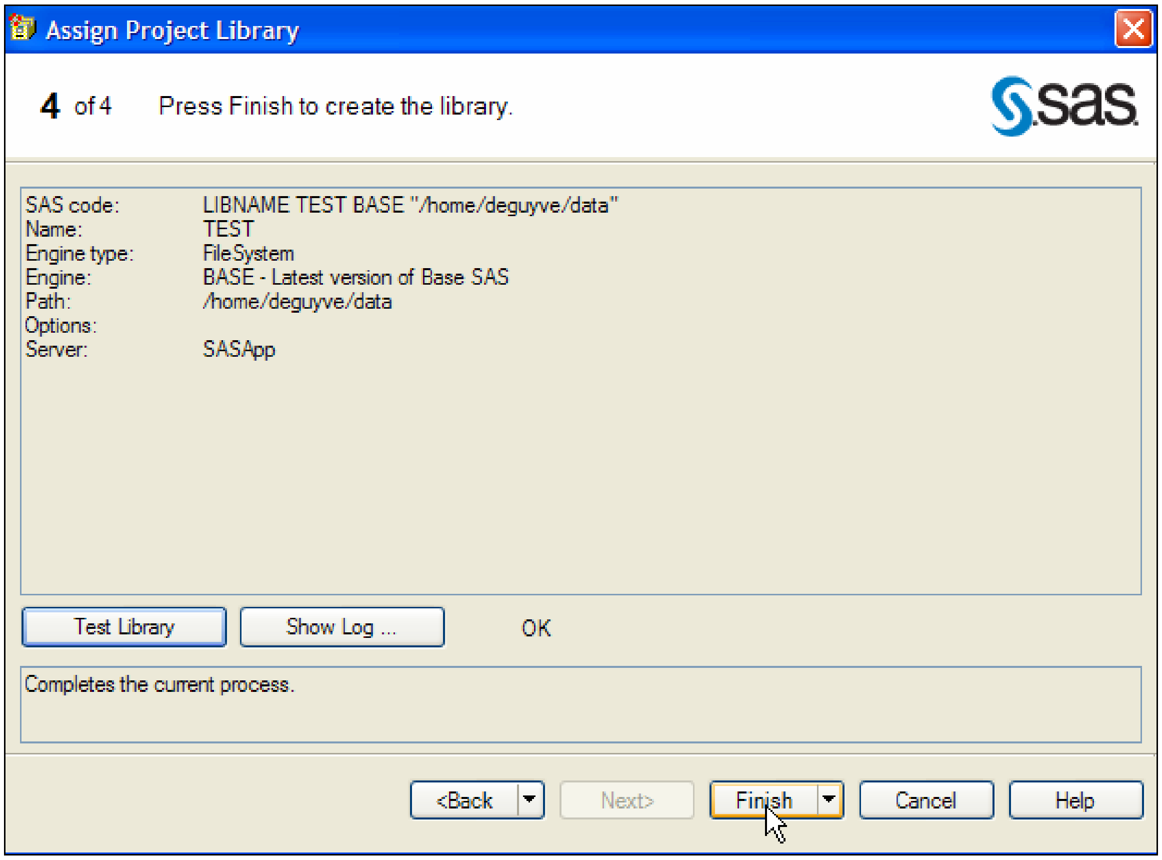
Server: SASApp

Local - The SAS server on your machine
SASApp

3. On the second screen, select *Browse. . .* beside the *Path:* box and follow the directory tree to the directory containing your file.



4. Follow the rest of the dialogue, which will assign the SAS Library.



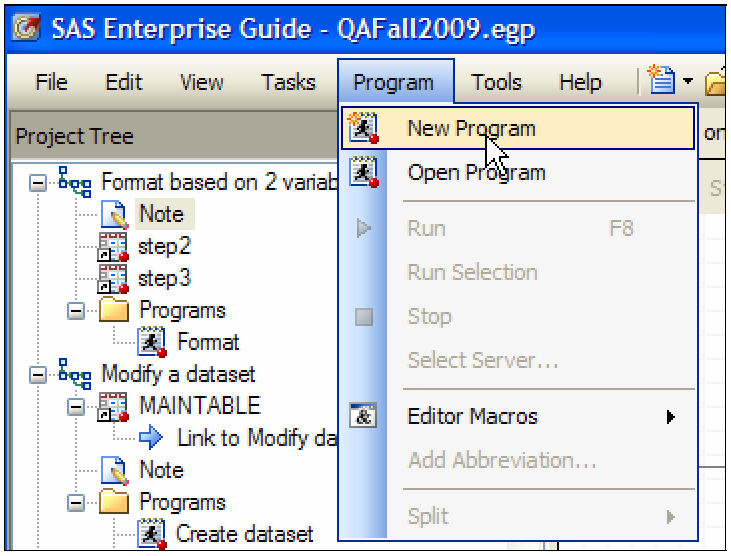
```
SAS code:      LIBNAME TEST BASE "/home/deguyve/data"
Name:          TEST
Engine type:   FileSystem
Engine:        BASE - Latest version of Base SAS
Path:          /home/deguyve/data
Options:
Server:        SASApp
```

Completes the current process.

5. The file can then be used from the library.

Using SAS Code

1. Select *Program* | *New Program*…

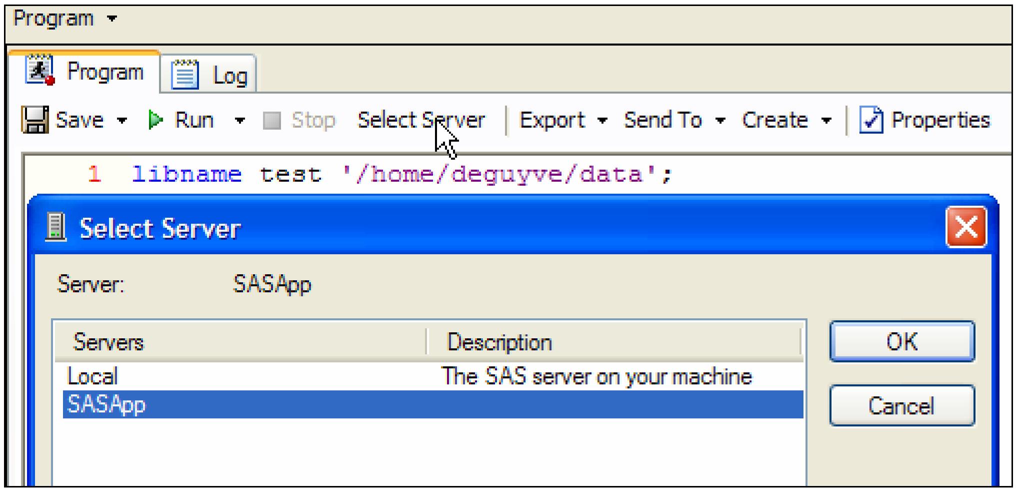


2. In the program editor, type a program like the following:

```
libname lname '/home/deguyve/data';
```

where *lname* is the name of the SAS library (8 characters maximum) and the entry in quotes is the path to the file.

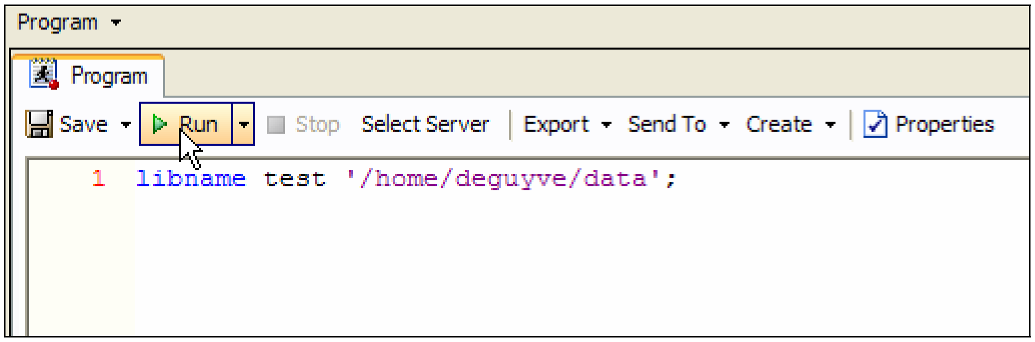
3. Run the program on your server to assign the SAS Library.



```
libname test '/home/deguyve/data';
```

Server: SASApp

Servers	Description
Local	The SAS server on your machine
SASApp	



```
libname test '/home/deguyve/data';
```

4. The file can then be used from the library .

If the Unix server isn't defined in the EG profile, the only way to access a file on that server is to use normal operating system facilities. Some options are:

- ftp the file from the server to your computer;
- Create a share on the server using Samba;
- Mount the directory to another server that is defined in EG.

These options are beyond the scope of a discussion about Enterprise Guide.

Question 6 : Without format, can’t view or open a dataset. Lots of error messages in SAS EG when format is not attached. How can we use the data without the format?

Whether or not a missing format generates an error depends on the SAS System Option FMTERR / NOFMTERR.

The following program sets FMTERR, so any variables that have missing format will cause an error and execution will stop:

```
OPTIONS FMTERR;

DATA WORK.FMT_TEST;
FORMAT VAR1 XXYYZZ.;
VAR1 = 1;
OUTPUT;
RUN;
```

as can be seen in the log:

```
15          OPTIONS FMTERR;
16
17          DATA WORK.FMT_TEST;
18          FORMAT VAR1 XXYYZZ.;
                        _______
                        48
ERROR 48-59: The format XXYYZZ was not found or could not be
loaded.

19          VAR1 = 1;
20          OUTPUT;
21          RUN;

NOTE: The SAS System stopped processing this step because of
errors.
WARNING: The data set WORK.FMT_TEST may be incomplete.  When this
         step was stopped there were 0 observations and 1
         variables.
WARNING: Data set WORK.FMT_TEST was not replaced because this
step was stopped.
```

On the other hand, if you specify NOFMTERR,

```
OPTIONS NOFMTERR;

DATA WORK.FMT_TEST;
```

```
FORMAT VAR1 XXYYZZ.;
VAR1 = 1;
OUTPUT;
RUN;
```

this log results,

```
15          OPTIONS NOFMTERR;
16
17          DATA WORK.FMT_TEST;
18          FORMAT VAR1 XXYYZZ.;
                        _______
                        484
NOTE 484-185: Format XXYYZZ was not found or could not be loaded.

19          VAR1 = 1;
20          OUTPUT;
21          RUN;


NOTE: The data set WORK.FMT_TEST has 1 observations and 1
variables.
```

a NOTE is produced instead of an ERROR, and execution continues.

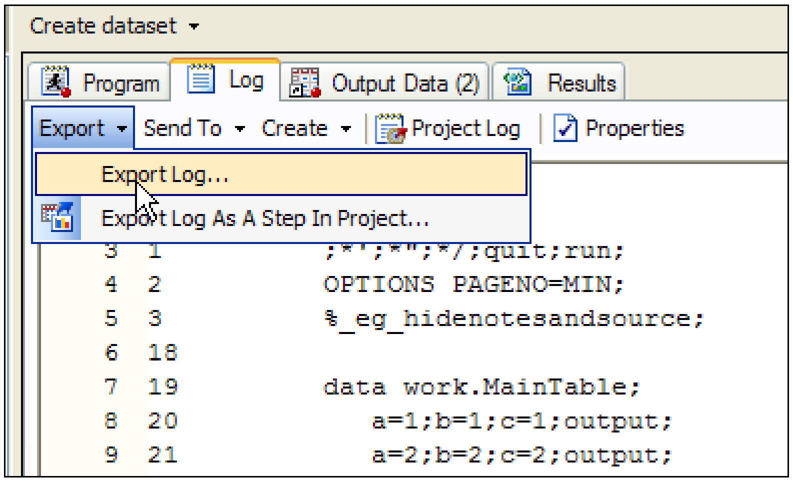
Question 7 : How to save log & output in SAS EG?

Each task in EG generates a log and more than likely an output of some sort (either a data or a report). Let's look at these different elements one by one.

Saving the log

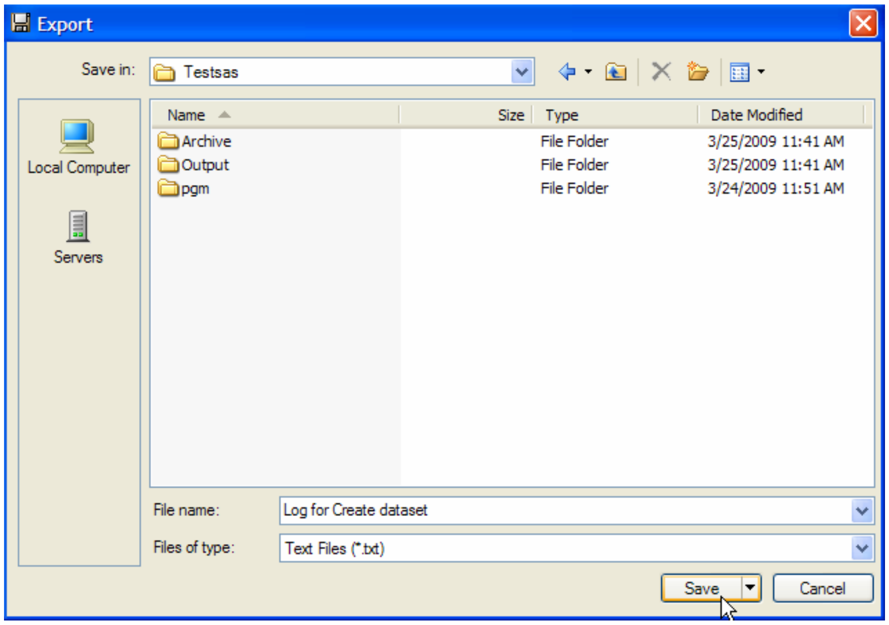
Saving the log generated by a task is easy from the log Windows.

Select the export facility.



```
1          ;*';*";*/;quit;run;
2          OPTIONS PAGENO=MIN;
3          %_eg_hidenotesandsource;
18
19         data work.MainTable;
20            a=1;b=1;c=1;output;
21            a=2;b=2;c=2;output;
```

Then select a file where to store the log.



Name	Size	Type	Date Modified
Archive		File Folder	3/25/2009 11:41 AM
Output		File Folder	3/25/2009 11:41 AM
pgm		File Folder	3/24/2009 11:51 AM

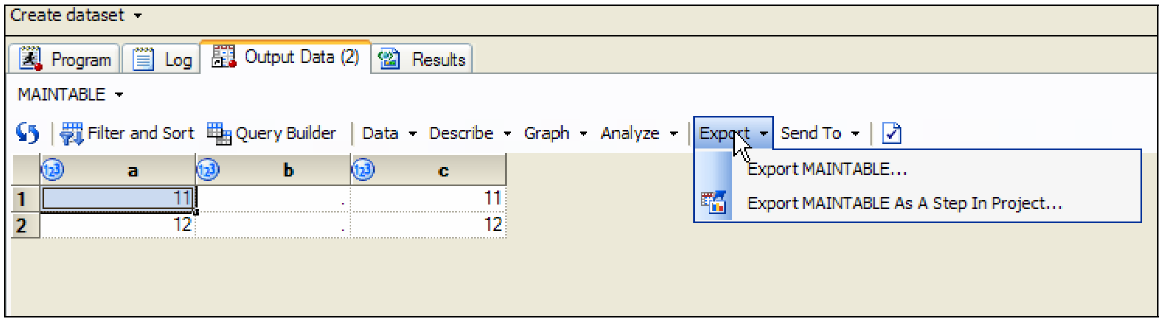
File name: Log for Create dataset

Files of type: Text Files (*.txt)

Saving the output data

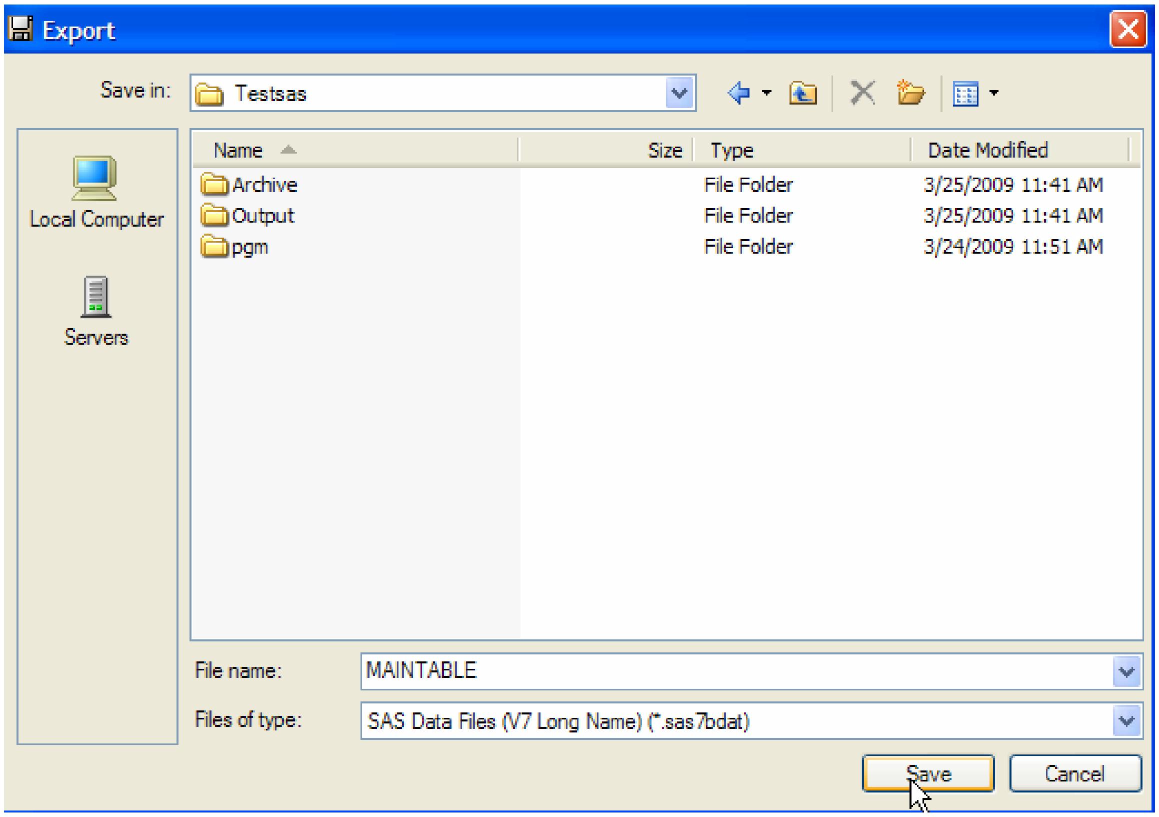
Assuming the output is a SAS dataset, this can be done in the same way as the log. Note that you can direct your output in a permanent library within as task, in which case you may not need to save it explicitly.

Select the export facility.



	a	b	c
1	11	.	11
2	12	.	12

Then select a file where to store the dataset.

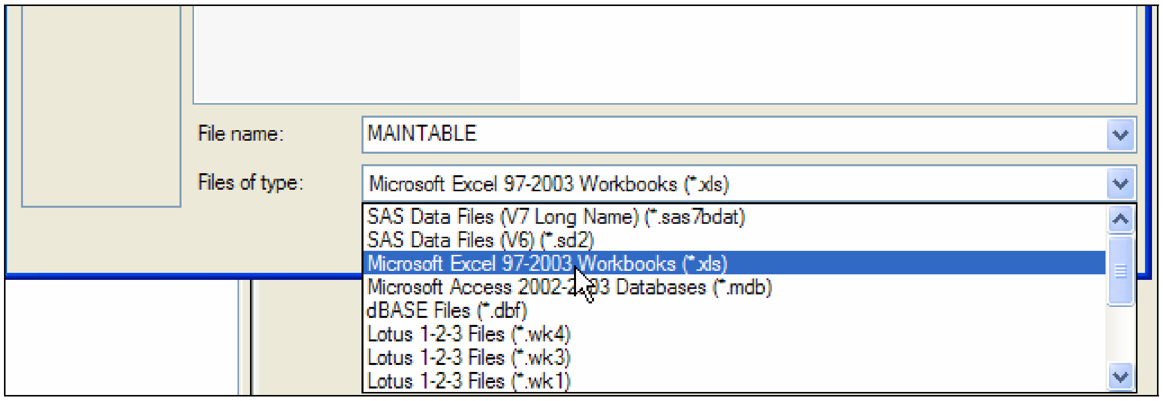


Name	Size	Type	Date Modified
Archive		File Folder	3/25/2009 11:41 AM
Output		File Folder	3/25/2009 11:41 AM
pgm		File Folder	3/24/2009 11:51 AM

File name: MAINTABLE

Files of type: SAS Data Files (V7 Long Name) (*.sas7bdat)

Note that you can select a type other than sas7bdat to save the data into.


File name: MAINTABLE

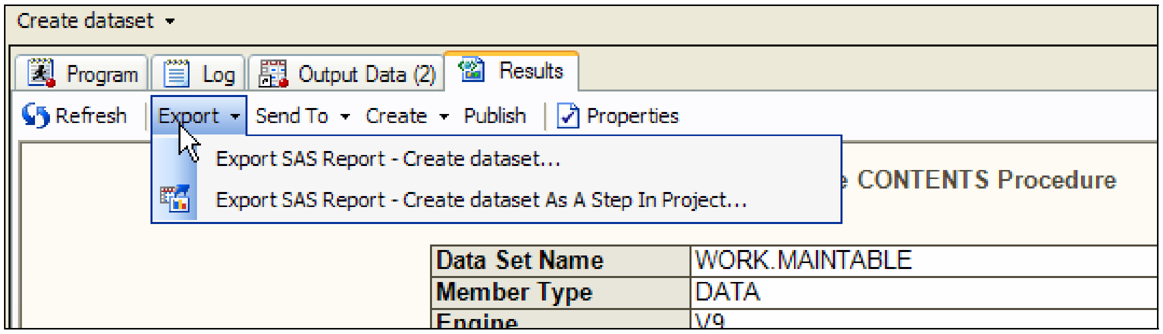
Files of type: Microsoft Excel 97-2003 Workbooks (*.xls)

SAS Data Files (V7 Long Name) (*.sas7bdat)
SAS Data Files (V6) (*.sd2)
Microsoft Excel 97-2003 Workbooks (*.xls)
Microsoft Access 2002-2003 Databases (*.mdb)
dBASE Files (*.dbf)
Lotus 1-2-3 Files (*.wk4)
Lotus 1-2-3 Files (*.wk3)
Lotus 1-2-3 Files (*.wk1)

Saving a report

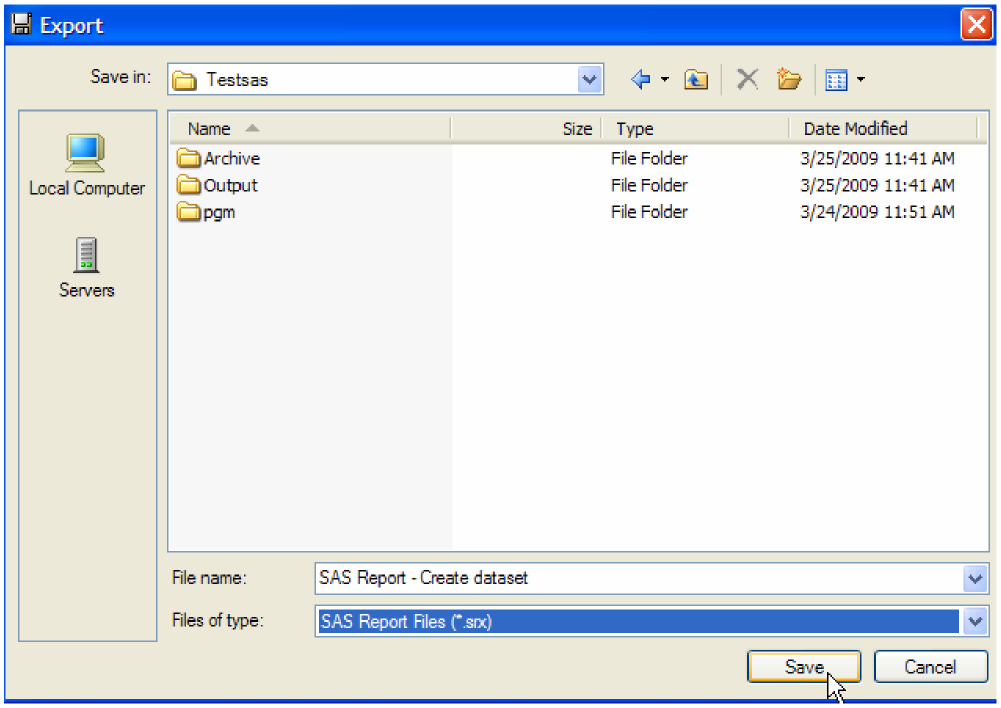
Assuming it is a SAS report that you want to save; a similar method can be used.

Select the export facility.



Data Set Name	WORK.MAINTABLE
Member Type	DATA
Engine	V9

Then select a file where to store the report.

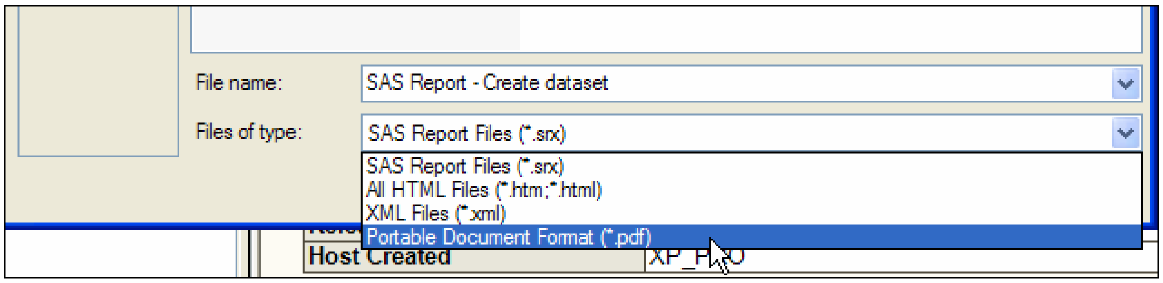


Name	Size	Type	Date Modified
Archive		File Folder	3/25/2009 11:41 AM
Output		File Folder	3/25/2009 11:41 AM
pgm		File Folder	3/24/2009 11:51 AM

File name: SAS Report - Create dataset

Files of type: SAS Report Files (*.srx)

You can select a type other than srx to save the report.



File name: SAS Report - Create dataset

Files of type: SAS Report Files (*.srx)

SAS Report Files (*.srx)
All HTML Files (*.htm;*.html)
XML Files (*.xml)
Portable Document Format (*.pdf)

Summary

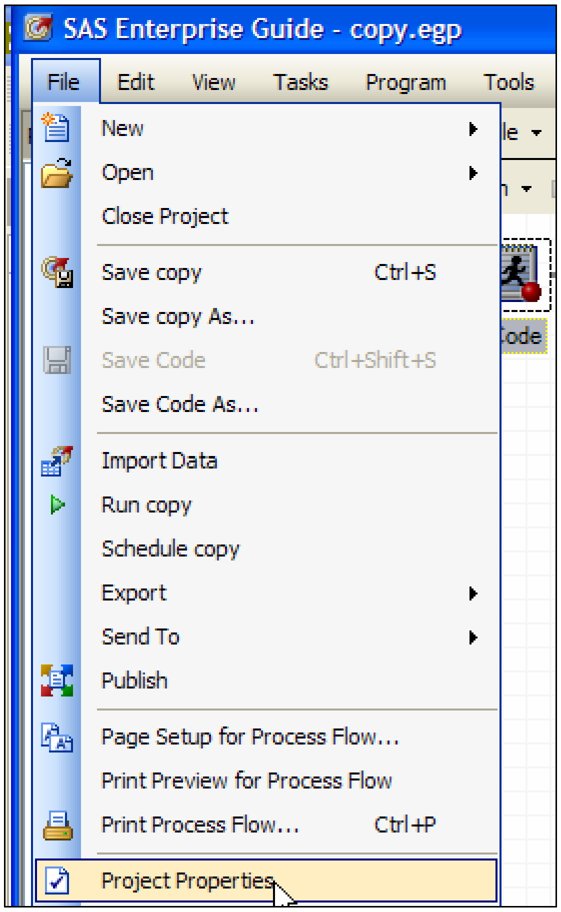
As you can see, saving your output and your log into an external file is easy! The approach is similar regardless of what you are trying to save. There are other features that can be used. For example, you can export as a step in your project so that every time an output gets creates, it is automatically exported. You can also send your output as attachment to an email message or directly to MS-Office (Word, PowerPoint or Excel depending on the type of output).

Question 8 : In SAS EG, how to extend the log instead of opening a new window or replace the old one?

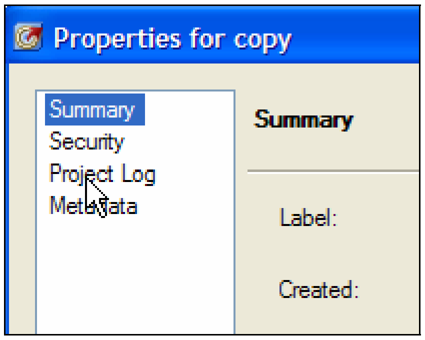
This can be achieved easily with a project log. A project log displays an aggregated log of tasks and code activity for the entire project. Each item that is recorded in the project log includes a date and time when the action occurred and additional details about the action.

The project log must be turned on for each project as followed:

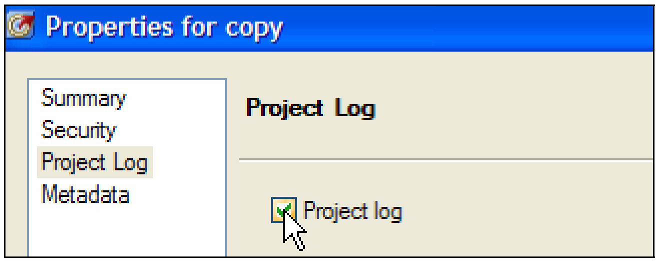
Access the properties of the project.



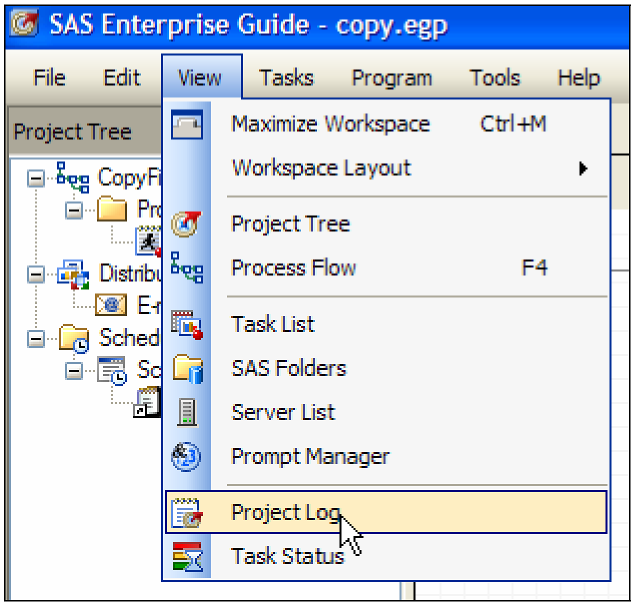
Select Project Log



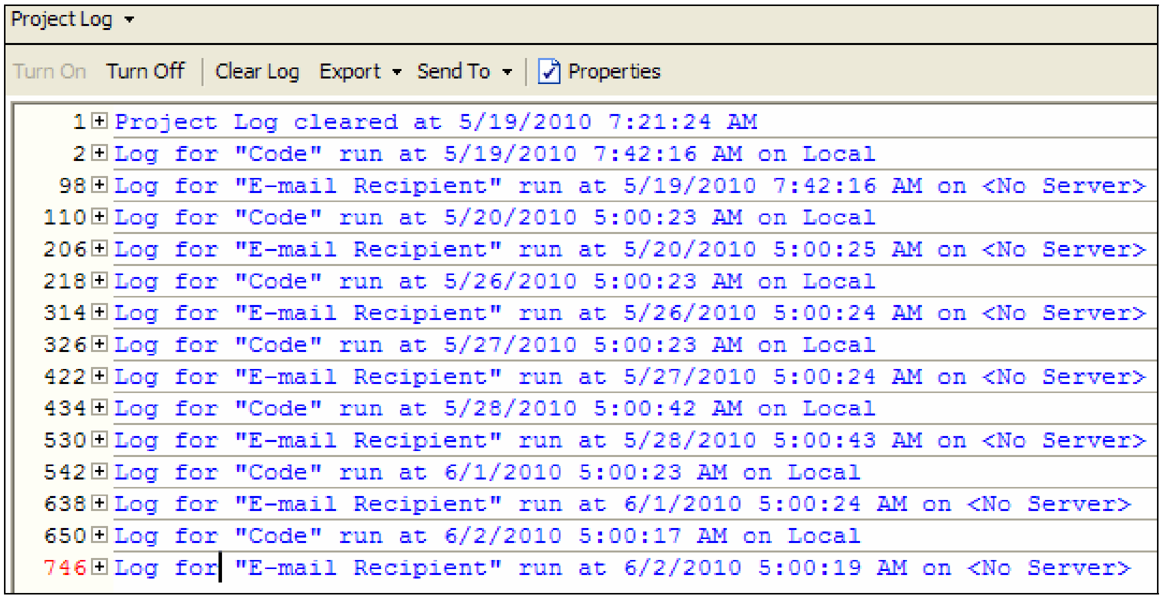
Click on the Project Log check box to turn on the Project Log.



You can access the Project Log viewer as followed:

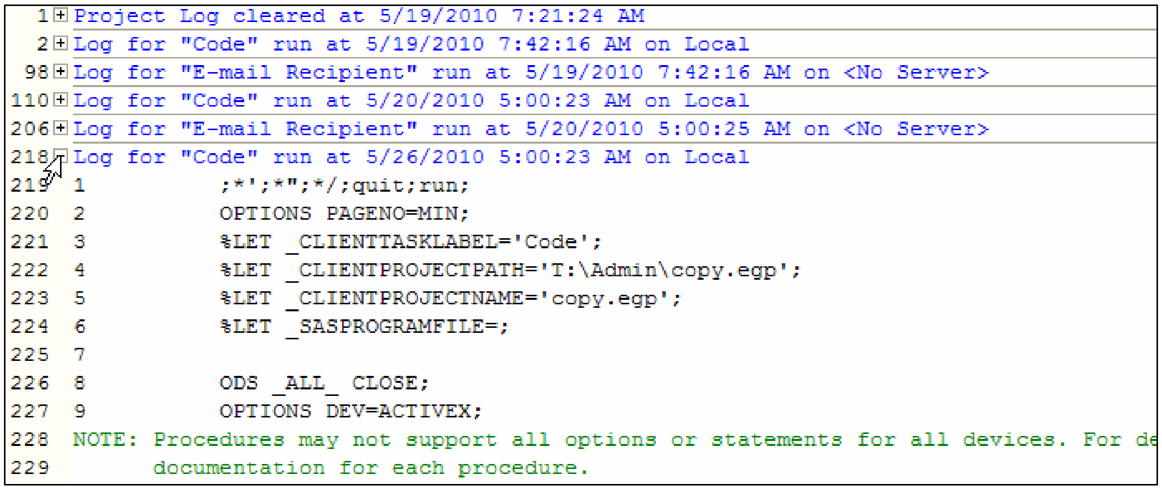


After running multiple tasks, multiple times; you could get results similar to this.



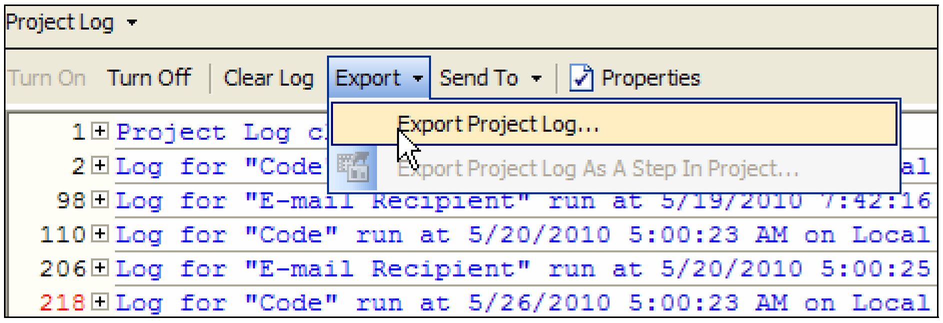
```
Project Log cleared at 5/19/2010 7:21:24 AM
Log for "Code" run at 5/19/2010 7:42:16 AM on Local
Log for "E-mail Recipient" run at 5/19/2010 7:42:16 AM on <No Server>
Log for "Code" run at 5/20/2010 5:00:23 AM on Local
Log for "E-mail Recipient" run at 5/20/2010 5:00:25 AM on <No Server>
Log for "Code" run at 5/26/2010 5:00:23 AM on Local
Log for "E-mail Recipient" run at 5/26/2010 5:00:24 AM on <No Server>
Log for "Code" run at 5/27/2010 5:00:23 AM on Local
Log for "E-mail Recipient" run at 5/27/2010 5:00:24 AM on <No Server>
Log for "Code" run at 5/28/2010 5:00:42 AM on Local
Log for "E-mail Recipient" run at 5/28/2010 5:00:43 AM on <No Server>
Log for "Code" run at 6/1/2010 5:00:23 AM on Local
Log for "E-mail Recipient" run at 6/1/2010 5:00:24 AM on <No Server>
Log for "Code" run at 6/2/2010 5:00:17 AM on Local
Log for "E-mail Recipient" run at 6/2/2010 5:00:19 AM on <No Server>
```

Every line represents a log for a specific run of a specific task in your project. You can expand each one to look at the details.



```
Project Log cleared at 5/19/2010 7:21:24 AM
Log for "Code" run at 5/19/2010 7:42:16 AM on Local
Log for "E-mail Recipient" run at 5/19/2010 7:42:16 AM on <No Server>
Log for "Code" run at 5/20/2010 5:00:23 AM on Local
Log for "E-mail Recipient" run at 5/20/2010 5:00:25 AM on <No Server>
Log for "Code" run at 5/26/2010 5:00:23 AM on Local
1          ;*';*";*/;quit;run;
2          OPTIONS PAGENO=MIN;
3          %LET _CLIENTTASKLABEL='Code';
4          %LET _CLIENTPROJECTPATH='T:\Admin\copy.egp';
5          %LET _CLIENTPROJECTNAME='copy.egp';
6          %LET _SASPROGRAMFILE=;
7
8          ODS _ALL_ CLOSE;
9          OPTIONS DEV=ACTIVEX;
NOTE: Procedures may not support all options or statements for all devices. For de
      documentation for each procedure.
```

You can export the project log to an external file.

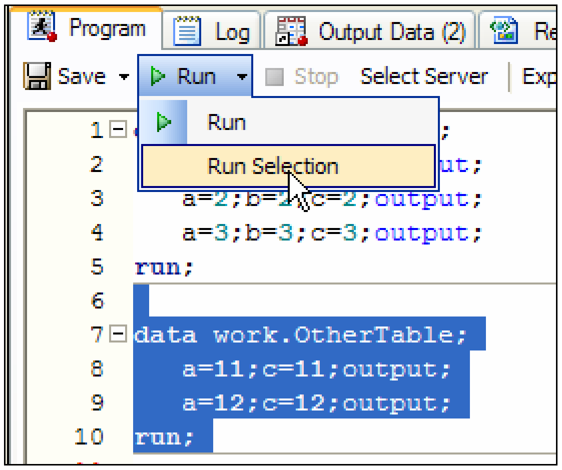


```
  1 ⊞ Project Log c
  2 ⊞ Log for "Code                                          al
 98 ⊞ Log for "E-mail Recipient" run at 5/19/2010 7:42:16
110 ⊞ Log for "Code" run at 5/20/2010 5:00:23 AM on Local
206 ⊞ Log for "E-mail Recipient" run at 5/20/2010 5:00:25
218 ⊞ Log for "Code" run at 5/26/2010 5:00:23 AM on Local
```

Question 9 : Is there a shortcut for “submit selection”? (There is one for “submit all”)

Yes, there is a shortcut to submit the current selection.

You can run your selection via the Run menu.



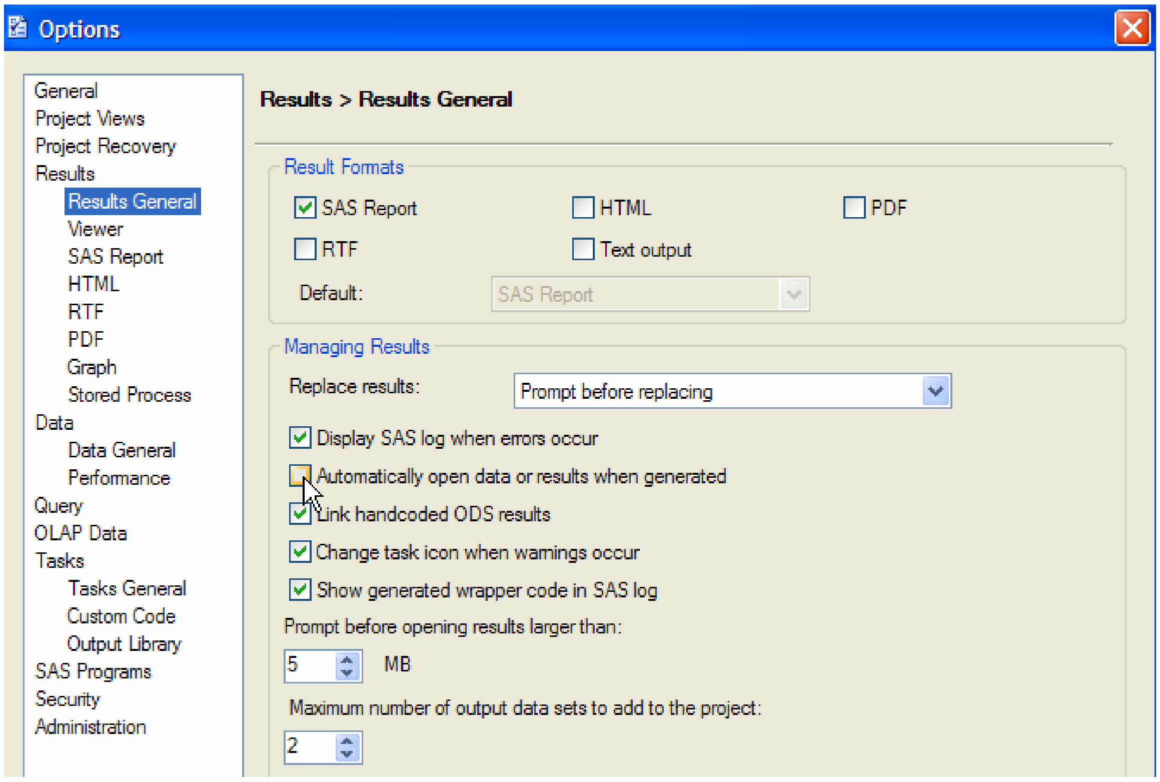
```
                              ;
                              ut;
    a=2;b=2;c=2;output;
    a=3;b=3;c=3;output;
run;

data work.OtherTable;
    a=11;c=11;output;
    a=12;c=12;output;
run;
```

You can also run the current selection using the F3 function key.

All the keyboard shortcuts are documented in the EG online Help under “Keyboard Shortcuts”.

Question 10 : Can you control the dataset popup after an EG node finishes?

This can be easily done by ticking or un-ticking the option “Automatically open data or results when generated”. This option is found under the Results General tab in Tools/Options.



Results > Results General

Result Formats

[✓] SAS Report [ ] HTML [ ] PDF

[ ] RTF [ ] Text output

Default: SAS Report

Managing Results

Replace results: Prompt before replacing

[✓] Display SAS log when errors occur

[ ] Automatically open data or results when generated

[✓] Link handcoded ODS results

[✓] Change task icon when warnings occur

[✓] Show generated wrapper code in SAS log

Prompt before opening results larger than:

5 MB

Maximum number of output data sets to add to the project:

2

Question 11 : What is the best way to migrate SAS tables from 9.1 to 9.2?

If you’re not changing machine architecture (for example moving from a Z/OS environment to UNIX), the best way is to use migration tools – namely the PROC MIGRATE procedure.

SUGI paper http://www2.sas.com/proceedings/sugi28/288-28.pdf provides more information on that topic.

Question 12 : When you open a data file, the headers you see are the labels, not the column names. How do we get the variable names by default?

This question was answered in the Fall 2008 OASUS Q&A (see question #1).