A How To Guide for Submitting Enterprise Guide Programs and Tasks to a SAS Grid

Q. Does SAS grid computing work with Enterprise Guide?

A. Yes, SAS programs that are generated by Enterprise Guide can be submitted to a grid to provide load balancing of jobs submitted by multiple EG users. This can be enabled to happen automatically by following the steps outlined below.

In order to accelerate a single Enterprise Guide program via parallelization, the program would have to be manually modified to add SAS/CONNECT and grid-specific statements to identify the sub-tasks to run in parallel on the grid.

Q. What are the technical steps for enabling Enterprise Guide to automatically submit jobs to the grid?

A. The following steps enable Enterprise Guide to automatically submit SAS programs to the grid.

1. The first step is to specify SAS statements to be added to the beginning and the end of the EG program or task. This "wrapper" code will submit the EG program or task to the grid but will not change the actual SAS program logic in any way.

   For EG programs, go to Tools->Options->SAS Programs, select "Insert custom SAS code before submitted code" and enter the following SAS statements.
   Similarly for EG Tasks, go to Tools->Options->Tasks, select "Insert custom SAS code before submitted code" and enter the following SAS statements.

   ```
   /*Modify the sample option values below to match your environment*/

   options metaserver='server1.domain.com';
   options metaport=8561;
   options metarepository='Foundation';
   options metauser='userid1';
   options metapass='pwd';
   /* The grdsvc_enable call will connect to the SAS Metadata */
   /* Server and find the SAS Grid Server definition. A return */
   /* code 0 means that all signons will use the grid.A non-0  */
   /* return code means that there is a problem that should be   */
   /* investigated. */
   %let rc=%sysfunc(grdsvc_enable(_all_,resource=SASMain));
   signon task1;
   rsubmit;
   ```

   For EG programs, go to Tools->Options->SAS Programs, select "Insert custom SAS code after submitted code" and enter the following SAS statements.

Similarly for EG Tasks, go to Tools->Options->Tasks, select "Insert custom SAS code after submitted code" and enter the following SAS statements.
Note: the %put is there to prove that the program was executed on the grid. You should delete the %put in a production environment.

```
endrsubmit;
%put This code ran on the machine %sysfunc(grdsvc_getname(task1));
signoff;
```

2. The next step is to set the METAAUTORESOURCES option for the invocation of the SAS sessions on the grid. This is important because it allows the SAS sessions spawned on the grid to have knowledge of predefined librefs in SAS metadata that are used in the code generated by EG. Because the SAS session will need to connect to the metadata server in order to retrieve the pre-defined librefs, you also must specify the options needed to connect to the metadata server. Add the METASERVER, METAPORT, METAUSER, METAPASS AND METAAUTORESOURCES options to the SAS invocation in each script (sasgrid.sh on Unix) or file (sasgrid.bat on Windows) used to invoke SAS on each grid node. Alternatively, you could create a single sasgrid.sh or sasgrid.bat on a share that all grid nodes can see and then you only have to update a single file.

For example, on a Unix system your SAS invocation in sasgrid.sh would look like the following, with all options on a single line:

```
cmd="/home/sas/SAS/DemoGrid/Lev1/SASMain/sas.sh  -dmr -sasuser
work -noterminal -nosyntaxcheck -ipaddress -set INHERIT 0 -
metaserver 'server1.domain.com' -metaport 8561 -metauser 'userid'
-metapass '\{sas001\}RWxpamFoMQ==' -metautoresources SASMain"
```

For example, on a Windows system your SAS invocation in sasgrid.bat would look like the following, with all options on a single line:

```
call sas.bat -dmr -sasuser work -noterminal -nosyntaxcheck -
ipaddress -set INHERIT 0 -metaserver 'server1.domain.com' -
metaport 8561 -metauser 'userid' -metapass
'\{sas001\}RWxpamFoMQ==' -metautoresources SASMain
```

NOTE – Replace the above sample values with the values that match your environment. If you are using an appserver other than SASMain then replace "SASMain" with your appserver name.
NOTE - The METAAUTORESOURCES option requires that the libraries defined in the metadata have the "pre-assigned" check box selected. In order to do this use SASMC and go to the "Properties" dialog for the library in question, select the "Advanced Options" dialog and check the box labeled "Library is pre-assigned".

3. You also need to ensure that work files needed from one node to the next can be accessed from all grid nodes. You can either point the EGTASK libref to a permanent shared storage location or define a new libref in the default list of librefs

used for WORK and point to a permanent shared storage location. There are multiple ways of defining the EGTASK libname to the SAS sessions on the grid:

- Use the METAAUTORESOURCES option on the sas command in the sasgrid.sh/sasgrid.bat file as detailed in step #2. The advantage of this is a single point of update if you want to change the directory that is being used.

- Add an environment variable to each sasgrid.sh/sasgrid.bat file that defines the EGTASK libname.

- Add the EGTASK libname statement to the autoexec file and add the –AUTOEXEC option to the SAS invocation command in each sasgrid.sh/sasgrid.bat file.

4. Finally, EG generates macros that are stored in EGAUTO.SAS and get called from the SAS program generated by EG. These macro definitions must be made available to the SAS sessions launched on the grid in order for the macro invocations to execute successfully. Locate the EGAUTO.SAS file and copy the macro definitions into the autoexec.sas file for each of the grid nodes and add the –AUTOEXEC option to the SAS invocation command in each sasgrid.sh/sasgrid.bat file.

The EGAUTO.SAS file is stored in a per-user location:

```
%appdata%\SAS\Enterprise Guide\4
```

For example:

```
C:\Documents and Settings\sascrh\Application Data\sas\Enterprise Guide\4
```