Paper A10-2007

Develop, Create and Deploy Sample Custom Tasks in SAS Enterprise Guide®

Steve Gunawan, Resort Condominiums International, Carmel, IN

ABSTRACT

SAS Enterprise Guide is a terrific tool that allows you to access the powerful analytics and data manipulation offerings of SAS. With the advent of tasks within SAS Enterprise Guide, you no longer need to recall the syntactic details of every proc, rather you can use a wizard-like menu to choose parameters and let the task generate the code. This has allowed SAS to rival the likes of Microsoft Excel in terms of ease of use, and has opened the door to a much larger user community. In addition to the out-of-the box tasks, users can now extend SAS Enterprise Guide by developing custom task solutions that are tailored to the specific needs of their organization or industry. In this paper, we will discuss the tools required to develop custom tasks, and how to create and deploy a sample custom task.

INTRODUCTION

SAS Enterprise Guide is an application that was built to simplify the way you interact with the powerful SAS engine. It enables novice users to easily utilize all the powerful functions available in SAS engine through a point and click interface, removing the need to remember the exact syntax of the functions. On the other hand, you are still given the option to write your own SAS code through the code editor, or combine your user-written code with any of the rich built-in task wizards.

Starting with SAS Enterprise Guide 2, you can even extend the application with your own custom task, enabling rapid and easier deployment of commonly used SAS task, code, or macros used in your organization. The custom task front-end has to be built as a Microsoft .Net class library (also called application extension/dll project) using either C# or VB.Net programming languages, however you can insert SAS codes, macros, and logics into the custom task as necessary.

In this paper, you are going to learn more about what is needed to get started creating your own custom task, how to create a custom task for SAS Enterprise Guide 4.1, how you can take advantage of it, and tips & tricks to help your custom task development easier.

This paper specifically targets the development of custom for SAS Enterprise Guide 4.1 using Visual Basic.Net in Microsoft Visual Studio .Net Professional Edition 2005 and assumes you have some familiarity with them.

GETTING STARTED

To start creating a custom task, you need to have these applications:

1. SAS Enterprise Guide 4.1 installed on the computer of the end user of the custom task
2. Microsoft Visual Studio .Net Professional Edition 2003 or higher installed on your computer
3. MSBuild Extras - Toolkit for .NET 1.1 “MSBee” installed on your computer, only if you use Microsoft Visual Studio .Net Professional Edition 2005

SAS ENTERPRISE GUIDE 4.1

Custom task has been supported since SAS Enterprise Guide version 2 and above. However since SAS Enterprise Guide 4.1 is the latest version from SAS, all the examples in this paper were tested in version 4.1.

MICROSOFT VISUAL STUDIO .NET PROFESSIONAL EDITION 2003 OR HIGHER

As of the time of this writing, SAS Enterprise Guide 4.1 only supports custom task in Microsoft .Net 1.1 class library written in C# or Visual Basic.Net. If you already have a copy of Microsoft Visual Studio .Net Professional Edition 2003, then you are all set. There is even a downloadable wizard that you can load into Visual Studio 2003 to help you getting started developing a SAS Enterprise Guide custom task project. In theory, you should also have no problem developing the custom task in an older version of Microsoft Visual Studio .Net 2002.

If you are using Microsoft Visual Studio .Net Professional Edition 2005, there are a couple things to note here. Microsoft Visual Studio .Net Professional Edition 2005 is designed to be used and deployed as Microsoft .Net 2.0 class library. Since SAS Enterprise Guide 4.1 only supports Microsoft .Net 1.1 class library, you need to use a tool called MSBee (see next section) to compile Visual Studio 2005 project into Microsoft .Net 1.1 class library. There are a few controls and functionalities in Visual Studio 2005 that you can’t use in the SAS Enterprise Guide 4.1 custom task because of this.

The examples in this paper were developed using Visual Basic.Net as Class Library project in Visual Studio 2005.

MSBUILD EXTRAS - TOOLKIT FOR .NET 1.1 "MSBEE"

This toolkit is only required if you are using Microsoft Visual Studio .Net Professional Edition 2005. This freely distributed toolkit from Microsoft allows developers to build managed applications in Visual Studio 2005 that target .NET 1.1 The idea is that you will do the front-end development within Visual Studio 2005, but you will use MSBee to compile the project to make is compatible with SAS Enterprise Guide 4.1.

CUSTOM TASKS EXAMPLE

Before getting started with writing your own custom task, let's take a look at a Classic Hat custom task example provided by SAS. Upon opening it, you will see a small dialog box popping up, asking you to enter your hat size. This dialog box is the custom task front end, developed in Visual Basic.Net using Microsoft Visual Studio .Net.

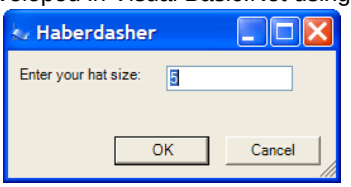

Figure 1. Classic hat custom task interface

After entering your hat size, hit OK, and you can see under the task status that SAS Enterprise Guide 4.1 is running a SAS code. Upon running the code completion, you will see a graph showing a hat picture with the size you specified.

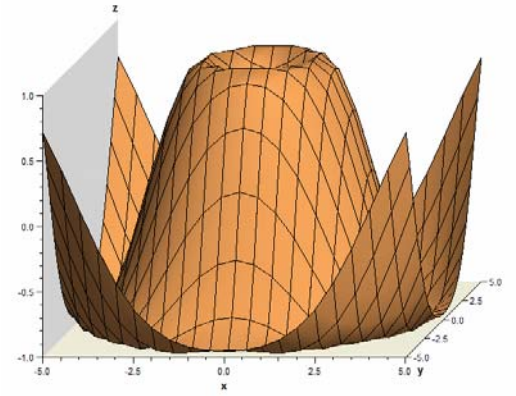

Figure 2. Classic hat custom task result

Rerun the Classic Hat custom task and enter different hat size to see the different result. Under Project Explorer, click the Last Submitted Code icon to see the SAS code that was recently ran to generate the hat picture:

```
data one;
data hat;
do x = -5 to 5 by .5;
do y = -5 to 5 by .5;
z = sin(sqrt(y*y + x*x));
output;
end;
proc g3d; plot y*x=z; run;
```

The maximum and minimum value of x and y of 5 was obtained from the custom task interface.

One thing to note here is that the SAS code was run only after the custom task interface was closed (upon hitting the OK button). On the next section, you are going to see custom task that runs SAS code while its interface is open.

CREATING SAS ENTERPRISE GUIDE 4.1 CUSTOM TASK

To make this section easier to follow, please download the zipped project file first (see Reference for link) and extract it to your desktop.

ADDING SAS REFERENCES

Whenever starting a new custom task project, you need to add the SAS Enterprise Guide 4.1 .Net APIs to the project's reference. To do this, right click on the project's name in Project Explorer, and then click Add Reference. Click on the Browse tab and browse to C:\Program Files\SAS\Shared Files\BIClientTasks\4\ folder and add SAS.Interop.SAS.dll and SAS.Shared.AddIns.dll

SAS CUSTOM TASK API

In order for SAS Enterprise Guide 4.1 to recognize the custom task properly, you need to implement a few interface

functions in CustomTask1.vb file. Most of them are just the property of the custom task (name, description). Since this custom task example doesn't run any SAS code upon exiting the interface, the Show function in ISASTask implementation returns ShowResult.Canceled instead of ShowResult.RunNow (which is what the Classic Hat example does).

FORM DESIGN

This custom task example is a simple custom task that can be used to display and edit the content of SAS tables. Thus you need to add the appropriate controls to the form.

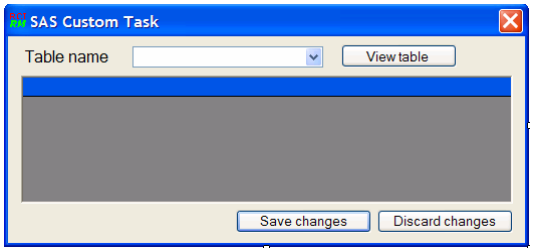

Figure 3. SAS custom task example form

In this example, we have a few buttons, a combo box, and a DataGrid. DataGrid is obsolete in Visual Studio 2005 and is replaced with DataGridView, however since DataGridView is a .Net 2.0 control, you can't use it in the project or otherwise you won't be able to compile the project as a .Net 1.1 class library. One trick to be able to use DataGrid in Visual Studio 2005 is to first add regular DataGridView to the form, then view the underlying code of the form and replace all "DataGridView" with "DataGrid".

EVENT HANDLER CODE

In this example, we are going to run some SAS code in the background while keeping the user interface open. Thus we have to add the event handler on the form. The first one is the Form Load event. You want to prepare connection to the SAS server along with populating any controls on the form. In this case, you need to populate the combo box with the table names that can be modified using the custom task.

The View Table button will fetch the table selected in the combo box and fill the datagrid with its contents. Save Changes button will get all rows modified on the datagrid and update the underlying SAS table with the changes made. Discard changes will reset the datagrid to its original content and discard the changes made on the datagrid.

MODULARITY OF SAS CODE AND VISUAL STUDIO CODE

When developing custom task it is recommended that you separate the SAS code/logic into its own file, which you can easily call from Visual Studio code. This allows the custom task development to be divided amongst Visual Studio developer(s) and SAS coder(s).

The best way to modularize the SAS code is to write it first in SAS Enterprise Guide 4.1, then save it as a regular .sas file. Then in Visual Studio 2005, right click on the project's name in Project Explorer, and click on Add -> Existing Item -> browse to the SAS code file you just wrote. One important thing to note here is that you need to change the Build Action property of all the SAS files to Embedded Resource, otherwise after compilation the custom task won't be able to find/read the SAS file.

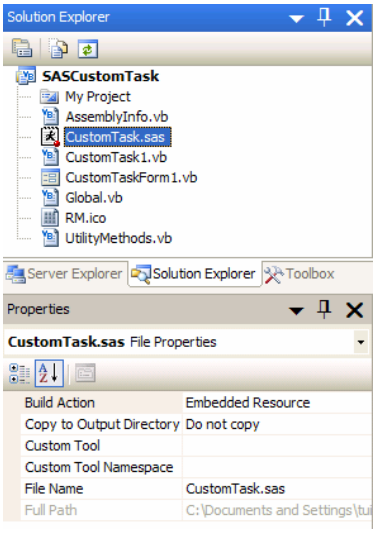

Figure 4. SAS code file Build Action set to Embedded Resource

COMMONLY USED FUNCTIONS

One major advantage of custom task is that we can use a lot of ADO.Net functionalities, such as DataReader and DataSet. This allows easier and powerful data access from the underlying SAS tables to the custom task interface.

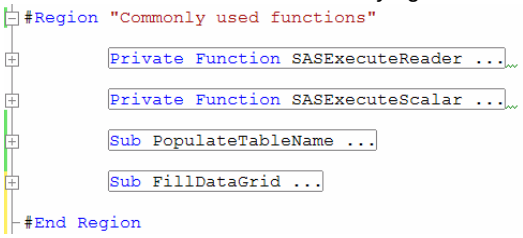

Figure 5. Commonly used functions grouped together in a region

One thing that you would want to do frequently is to obtain inputs from the user interface, then pass these values to the SAS code and run it. To do this, the input value on the custom task interface can be added to the prefix of the SAS code file as macro variables. Thus an event handler for a button click can contain the following code:

```
Dim macroVars As String
macroVars = String.Format("%let var1={0};" & Environment.NewLine,
Trim(txtBox1.Text))
macroVars += String.Format("%let var2={0};" & Environment.NewLine,
Trim(txtBox2.Text))
Me._submit.SubmitCode(macroVars & [Global].ReadFileFromAssembly("CustomTask.sas"),
Me._currServer, False)
```

where CustomTask.sas file contains:

```
proc sql;
  create table output_table as
    select *
    from input_table
    where col1=&var1 or
          col2=&var2;
quit;
```

COMPILATION

Since the compiler in Visual Studio 2005 generates a .Net 2.0 class library that is not compatible with SAS Enterprise Guide 4.1, you need to use MSBee to compile the project. In order to use MSBee, there is one thing you need to modify in the SASCustomTask.vbproj file. Open the SASCustomTask.vbproj file with Notepad, and then look for the following line toward the bottom of the file:

```
<Import Project="$(MSBuildBinPath)\Microsoft.VisualBasic.targets" />
```

Immediately under this line, insert the line below.

```
<Import
Project="$(MSBuildExtensionsPath)\MSBee\MSBuildExtras.FX1_1.VisualBasic.targets"
Condition=" '$(BuildingInsideVisualStudio)' == '' AND '$(TargetFX1_1)'=='true'" />
```

For further details, please read the “MSBee ReadMe.doc” that you can find in the folder where MSBee is installed (default path is C:\Program Files\MSBuild\MSBee).

To compile the project in an easy way, copy the following lines to notepad and save as compile_customtask.bat:

```
C:\WINDOWS\Microsoft.NET\Framework\v2.0.50727\msbuild "C:\Documents and
Settings\<username>\Desktop\SASCustomTask\SASCustomTask.vbproj" /t:Rebuild
/p:TargetFX1_1=true
Pause
```

Later on you can easily (re)compile the project just by double clicking the batch file compile_customtask.bat.

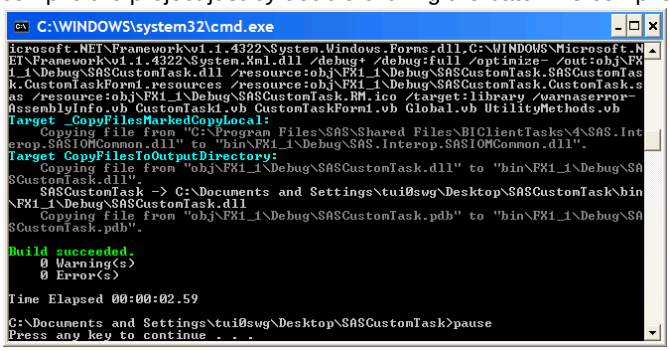

Figure 6. Successful compilation of the project with MSBee

LOADING THE CUSTOM TASK INTO SAS ENTERPRISE GUIDE 4.1

To load the custom task once compiled successfully, click on the Add-In menu in SAS Enterprise Guide 4.1 and select Add-In Manager. Browse to the custom task project folder. When compiled with MSBee, the result is stored in a folder called \obj\FX1_1\Debug. Select the file SASCustomTask.dll and exit the Add-In Manager. Now you can access the custom task you just loaded either from the Add-In menu or from the Task List.

When the custom task is loaded, the dll file is locked by SAS Enterprise Guide 4.1. The only way you can recompile the project and rewrite the dll file is by closing all of your SAS Enterprise Guide 4.1 sessions.

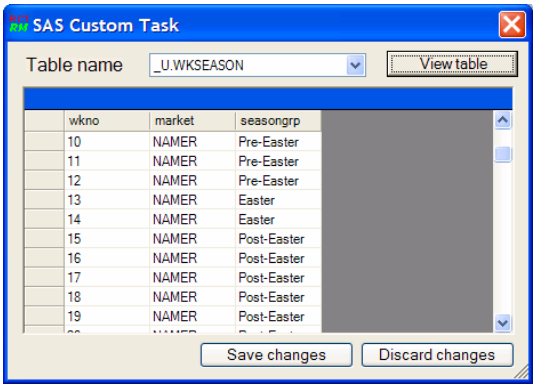

Figure 7. SAS Custom Task example loaded in action

DEBUGGING

Debugging the custom task is not trivial since we are not using the built-in compiler in Visual Studio 2005. So if you need to debug your code, you need to attach Visual Studio 2005 to the SAS Enterprise Guide 4.1 process. This can be done from the Tools menu in Visual Studio 2005, then selecting Attach to Process and selecting SEGuide.exe. Then you can set the breakpoints in Visual Studio 2005 and upon running the custom task in SAS Enterprise Guide 4.1, Visual Studio 2005 will stop at the preset breakpoints.

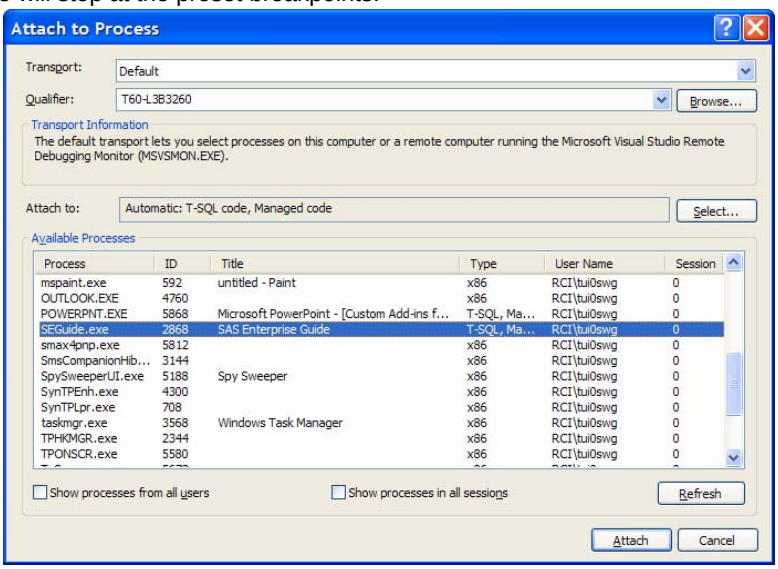

Figure 8. Attach to Process for debugging purposes

LEARNINGS

Custom task development would be easier if done in Visual Studio 2003 since SAS Enterprise Guide 4.1 API only supports .Net 1.1. However if you are stuck with Visual Studio 2005, it is possible to develop custom task for SAS Enterprise Guide 4.1 with the help of MSBee and using only .Net 1.1 compatible controls. DataGrid is one of the most useful controls in custom task development, and even though it's not available directly from Visual Studio 2005 controls, one can trick the underlying user form code to make DataGrid available inside Visual Studio 2005.

A couple of useful and commonly used VB.Net functions in custom task development can be found in the custom task example discussed in this paper. Another trick is to compile project using MSBee using a batch file. To debug the custom task project, one can attach the SAS Enterprise Guide 4.1 process to Visual Studio 2005 and set appropriate breakpoints.

CONCLUSION

The custom task in SAS Enterprise Guide can be used to wrap a lot of SAS code, macro, functions, and logic into a simple and easy to use interface. The ability to create your own custom task enables you to tailor the interface to fit your organization's need. This has opened the door to a much larger user community, enabling quicker user acceptance and rapid deployment.

REFERENCES

SAS Institute Inc. "Creating Custom Add-In Tasks for SAS Enterprise Guide."
http://support.sas.com/documentation/onlinedoc/guide/customtasks/index.htm

CodePlex. "MSBuild Extras - Toolkit for .NET 1.1 "MSBee"." http://www.codeplex.com/MSBee

Steve Gunawan. "SAS Custom Task project." http://padangboy.com/MWSUG/SASCustomTask.zip

ACKNOWLEDGMENTS

Special thanks to my mentor, Jeremy TerBush for guiding me in exploring SAS Enterprise Guide® and encouraging me to submit a paper and present for the MWSUG conference.

RECOMMENDED READING

David Shannon. "My Enterprise Guide." http://www2.sas.com/proceedings/sugi31/169-31.pdf

Alex Dmitrienko. "Introduction to stored processes and custom tasks."
http://www.mwsug.org/Conference/2005/Application_Development/AA200.pdf

Peter Eberhardt. "Be Your Own Task Master - Adding Custom Tasks to EG."
http://www8.sas.com/scholars/Proceedings/2006/Applications/AP05_06.PDF

CONTACT INFORMATION

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

Steve Gunawan
RCI
9998 N. Michigan Road
Carmel, IN 46032
Work Phone: 317-805-9764
Fax: 317-805-8173
E-mail: steve.gunawan@rci.com
Web: http://padangboy.com