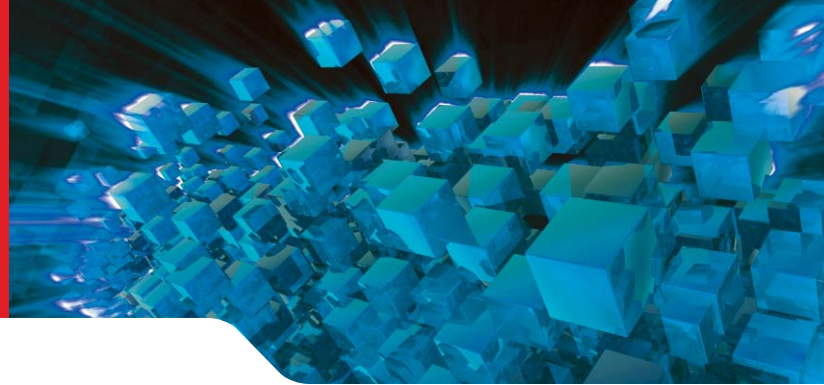




Business Analytics



SAS® Enterprise Guide® OLE Automation Using C#

Kevin Kramer, COMSYS, Kalamazoo, MI

Do business. **We're IT.**

ABSTRACT

Like many other applications including the Microsoft Office Suite, SAS has provided access into the internal methods and properties of Enterprise Guide through the use of OLE Automation. With OLE Automation, clients have a way to programmatically control Enterprise Guide applications. This feature allows many products to work together as a single entity to solve a common task. There are now hundreds of applications that can be used in this way. Microsoft now claims that every function of every application in the Office suite is available to a developer via the application's object model.

The key to controlling an Automation application is to have a firm understanding of the application's object model. Each application's object model is different. To expose the object model of an application is to expose the objects, methods, and properties to the outside world. SAS Enterprise Guide exposes many of its properties and methods, which enables us to develop applications that can take advantage of the capabilities that it provides.

This paper's intent is to shed some light on using OLE Automation with Enterprise Guide. It will focus on how to develop Windows applications using Enterprise Guide as the Automation server. It touches on several of the objects that have been made available through OLE Automation and provides an example of how each can be used to solve a specific task. The reader of this paper should be familiar with the constructs of object oriented programming but knowledge of C# is not required.

INTRODUCTION

SAS Enterprise Guide is a thin-client Windows application that brings the power of the SAS System to your desktop by providing a point-and-click user interface to all SAS System servers. It allows users the ability to generate SAS code and formatted results all at the click of a mouse. SAS Enterprise Guide projects consist of a collection of related data, tasks, code, and result objects. Each task is made up of Datasteps and Procedures that can produce simple data listings to the most complex analytical routines.

In the following sections, I will discuss the Enterprise Guide object model and the components used in automating an Enterprise Guide project. I will also provide several examples of using C# to automate an Enterprise Guide project. I will touch on 5 basic components of the Enterprise Guide Object model, giving examples of using

each one to perform a specific task. Keep in mind there are many objects and methods not covered in this paper.

The five components of the object model presented in this paper are:

- Application – Get the collection objects and shut down the automation server
- Project – Access all project elements
- Task – Control individual tasks
- Code – Insert or modify existing code
- Data – Access a data element in a project

ENTERPRISE GUIDE OBJECT MODEL

What Is An Object Model?

In Enterprise Guide, the application's functionality is exposed through an object interface. For the purposes of this paper, an object is defined as an item that can be programmed or controlled, such as a task, data, output, or even log object. An object model is a representation—or conceptual map—of an application's functionality in terms of objects. By interacting with an application's object model, you can manipulate the application to add custom functionality, automate processes, or integrate applications across networks.

The Object Model Hierarchy

SAS Enterprise Guide has many different objects that are organized into various levels. You can think of these levels as tiers in a hierarchy. The topmost tier of each application object hierarchy is typically occupied by a single object: **Application**. The **Application** object represents the application itself, and all other objects for that application are below the **Application** object. The second tier consists of the **Project** object. The project object is the access point for all the elements in an Enterprise Guide project. The tiers below that include a variety of additional objects used to access functionality that the Project object contains including task collections, data collections, code collections and more.

Components of the Object Model

The key to controlling an Automation application is a firm understanding of the application's object model. Each application's object model is different. Enterprise Guide provides two ways developers can use to interact with its environment. A developer can use the Component Object Model (COM) add-in or OLE automation. Each way provides a different object model for developers to

use along with different functionality. Since the focus of this paper is on Automating Enterprise Guide, the Component Object Model will not be discussed. The help page describing the properties and methods in the object model can be referenced at: <http://support.sas.com/documentation/onlinedoc/guide/index.html>

OLE AUTOMATION USING ENTERPRISE GUIDE

OLE automation allows a client to programmatically control the Enterprise Guide application itself. It provides the means for running existing projects or tasks, dynamically creating code nodes, retrieving information about data objects, and various other properties and methods used to extract or save information contained in Enterprise Guide projects.

Getting Started

Before we can begin developing OLE Automation projects we need to know how to set a reference to the Enterprise Guide automation server. We can set up a reference to the Server by selecting the SAS EG scripting library from the Enterprise Guide program folder.

The following steps describe the process of setting up such a reference:

1. Right click on references in our solution explorer
2. Select Add references
3. Select Browse
4. Browse to the Enterprise Guide folder and select the SASEGScripting.dll

Application Object

The Application object represents the SAS Enterprise Guide application. It is the root object of the automation object hierarchy.

The main methods of the Application object include the following:

- Opening an existing project
- Creating a new project
- Shutting down the automation server.

The Application objects can also control a subset of properties for setting up the environment. A couple examples would be setting the sort method to use Proc SQL for sorting or setting the default footnotes for a project, but the main use of the Application object is opening or creating a new project to gain access to the tasks within it.

Below is an example of using the Application object to start an existing EG project.

```
' Get a reference to the application object
SAS.EG.Scripting.Application EGApp = new SAS.EG.Scripting.Application();

' Open a project in the application
SAS.EG.Scripting.Project EGProject = EGApp.Open("c:\project.egp","");

< perform other EG tasks here >

' Exit the application
EGApp.Quit();
```

Project Object

The Project object is a child of the Application object. It represents a SAS Enterprise Guide project. All of the other Enterprise Guide objects used to manipulate data, code, tasks, and other functions in an Enterprise Guide project flow from the Project object.

The main methods of the Project object include the following:

- Run all tasks in a project
- Close a project
- Save a project
- Save a project to a new file
- Send the contents of a project as an attached file in an e-mail message

The properties of a Project object can also be used to provide information about a project including the path to the project, whether a project was modified or not, a collection of Code objects or Data objects contained in the project or even a collection of Process Flow objects.

Below is an example of using the Project object to run all the tasks in a Enterprise Guide project.

```
' Get a reference to an application object
SAS.EG.Scripting.Application EGApp = new SAS.EG.Scripting.Application();

' Open a project in the application
SAS.EG.Scripting.Project EGProject = EGApp.Open("c:\project.egp","");

' Run the project
Int retCode = EGProject.Run();
```

Task Object

The Task object represents the task that is associated with a data set or a query in the project. The Task object is a child of the Tasks collection object.

The main methods of the Task object include the following:

- Run the task
- Get the results

The task object also provides properties for retrieving the code, log, or even output datasets associated with a task. The properties can be used to gather information about a specific task and used to generate other tasks. One of the main properties of the Task object is the Taskcode object. This returns the code associated with a task.

Below is an example of using the Task object to retrieve the SAS code for a task

```
' Get the first task object from the EGProject
object
SAS.EG.Scripting.Task EGTask = EGProject.Tasks(0)

' Retrieve the SAS code from the Tasks Taskcode
object
Console.Wri te(EGTask.TaskCode.Text.ToString())
```

Code Object

The Code object represents a code element in the project. The Code object is a child of the CodeCollection and object.

The main methods of the Code object include the following:

- Run code on a server
- Save the code to a file

The Code object has many properties. Some of the properties that would be used more frequently would include the Log property for return a log object, the Results property for returning the results from the run of the Code object, and the TaskCode object that returns the object containing the SAS code.

Below is an example of creating a new Code object in a project.

```
' Insert a code object into the current EG
project
EGProject.Code code = EGProject.CodeCol lecti on.Add

' Populate the code object with the customized
code
code.Text = "Data test; var1 = 'Hello World'; put
var1; run;"
```

```
' Run the custom code module
Int retCode = code.Run()
```

Data Object

The data object represents the data element in the project. The main methods of the Data object include the following:

- Save the dataset to a file
- Save the dataset to another dataset
- Determine the location of a dataset

Like the other objects mentioned above, the Data object also allows users the ability to set or retrieve information about a Data set. Some of those properties would include retrieving the location or filename for the data or setting/retrieving a password to the data.

Below is an example of using the Data Collection to iterate over each Data object and display the filename or location associated with it.

```
try {
// Get the data Collection for the project
SAS.EG.Scrip ting.DataCol lecti on obj DataCol =
EGApp.DataCol lecti on;

// Retrieve all the data objects from the
project
foreach (SAS.EG.Scrip ting.Data obj Data in
obj DataCol ) {

// display the filenames for the data objects
Console.Wri te(obj Data.fileName.ToString());
}
}
catch (Exception ex) {
throw new Exception("Error retrieving
data.");
}
```

Bringing It All Together

Now that we know how to use each of the 5 objects described above to gain access to an Enterprise Guide project, it's time to bring them all together to perform a specific task. One of the most common tasks in OLE automation is extracting some bit of information from the automation server. In this case the automation server is Enterprise Guide and the information we are going to extract is the programs, logs, and output for all the tasks in a project.

In order for us to accomplish this task, the following example will use the methods and properties from the objects described above.

```
// Get the application object and open the project
SAS. EG. Scripting. Application EGApp = new SAS.
EG. Scripting. Application();
EGApp.Open("c:\\EGSample\\project.egp", "");

// Run the project to generate the logs and output
EGApp.Project.Run();

int i=0;

// Retrieve all the data objects from the project
foreach (SAS. EG. Scripting. Data objData in EGApp.
Project.DataCollection)
{
// Loop through all the tasks for the data
objects
foreach (SAS. EG. Scripting. Task objTask in objData.
Tasks)
{
// Save the sas log and code for the task
objTask.TaskCode.SaveAs("c:\\EGSample\\TaskCode_" +
i + ".sas");
objTask.Log.SaveAs("c:\\EGSample\\TaskCode_" + i +
".log");

// Loop through all the results for the task and
output the HTML RTF or PDF info to files
foreach (SAS. EG. Scripting. Result objResult in
objTask.Results)
{
int j = 0;
string ext = ".txt";

// Check the type of output and assign the
appropriate extension
if (objResult.Name.StartsWith("HTML"))
ext = ".html";
else if (objResult.Name.StartsWith("RTF"))
ext = ".rtf";
else if (objResult.Name.StartsWith("PDF"))
ext = ".pdf";

// Save the output from the task to a file with
the appropriate extension
objResult.SaveAs("c:\\EGSample\\" + objResult.Name +
"_" + i + "_" + j + ext);
j = j + 1;
}

i = i + 1;
}
}

// Quit the Enterprise Guide application
EGApp.Quit();
}
```

CONCLUSION

This paper describes automating Enterprise Guide projects using OLE Automation with C#. The examples above provided a look into the object model that is provided by Enterprise Guide and some of the ways one can get started automating Enterprise Guide projects through other applications. This paper only touched on a few of the basic components that will get you started. For a more in depth description of the objects, methods, and properties provided in Enterprise Guide, visit the SAS website at <http://support.sas.com>.

For any questions regarding these techniques, please contact the author.

CONTACT INFORMATION

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

Kevin Kramer
Application Developer
COMSYS
5278 Lovers Lane
Kalamazoo, MI 49002

Email: kkramer@comsys.com
Phone: 269-344-4100 ext 540

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.