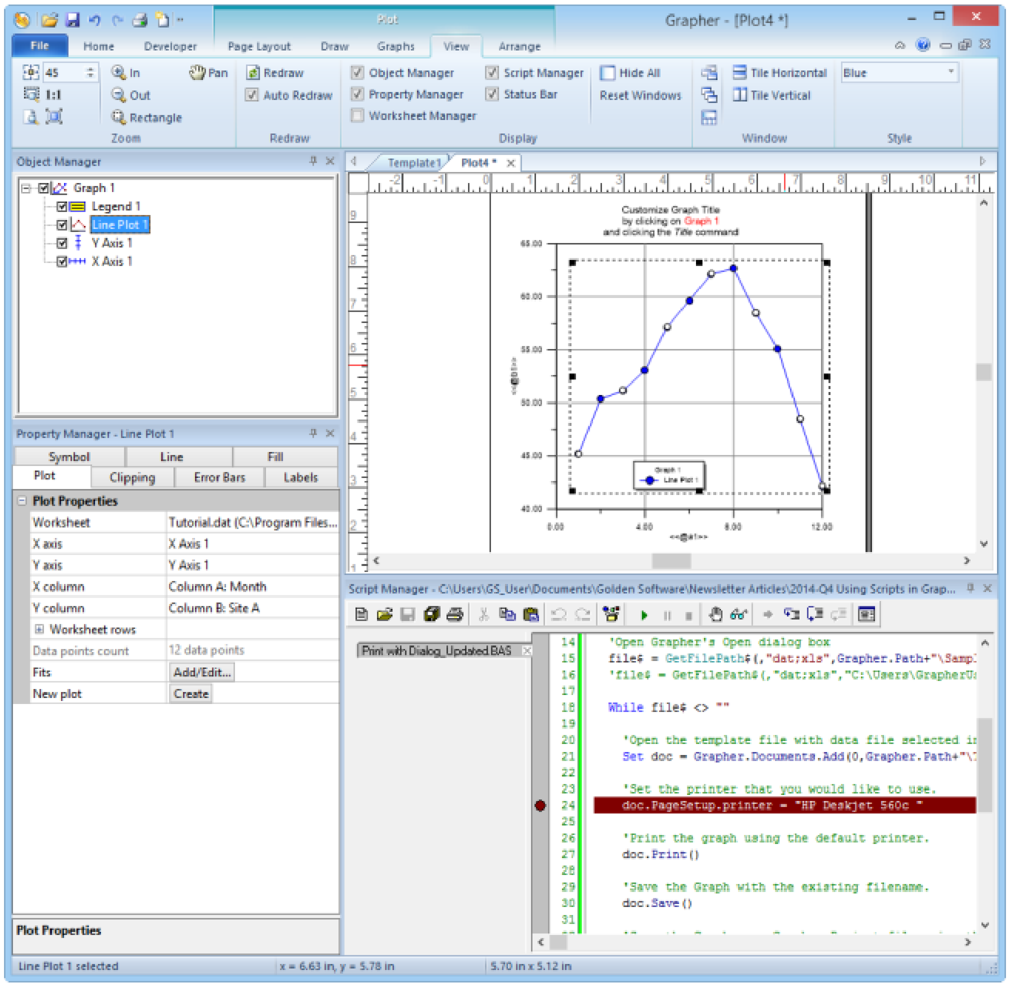
Print your data files automatically using Scripter, a Grapher Template, and the example script: *print with dialog.bas*

One of the great features in Grapher 11 is the ability to use scripts to automate the creation of graphs. A script is a text file containing a series of instructions carried out sequentially to automate commands in Grapher. Instructions are written in a Visual BASIC-like programming language. You can do almost everything with a script that you can do manually with the mouse or from the keyboard. Scripts are typically used to perform repetitive tasks, or to ensure consistency when creating graphs from new datasets. Using scripts can improve your efficiency when you want to create the same graph using multiple datasets.

To create and run scripts, you can use Golden Software's **Scripter**. This is a program for developing and running scripts that comes free with Grapher. You can run Scripter as an independent application, or you can display the **Script Manage**r within Grapher to view and edit scripts. Grapher also comes with a script recorder, which will record each command you use when in recording mode. You can then run this script at a later time to create a graph using the exact same steps.



Worksheet	Tutorial.dat (C:\Program Files...
X axis	X Axis 1
Y axis	Y Axis 1
X column	Column A: Month
Y column	Column B: Site A
Worksheet rows	
Data points count	12 data points
Fits	Add/Edit...
New plot	Create

Script Manager - C:\Users\GS_User\Documents\Golden Software\Newsletter Articles\2014-Q4 Using Scripts in Grap...

Print with Dialog_Updated.BAS

```
'Open Grapher's Open dialog box
file$ = GetFilePath$(,"dat;xls",Grapher.Path+"\Samp
'file$ = GetFilePath$(,"dat;xls","C:\Users\GrapherU

While file$ <> ""

  'Open the template file with data file selected i
  Set doc = Grapher.Documents.Add(0,Grapher.Path+"\

  'Set the printer that you would like to use.
  doc.PageSetup.printer = "HP Deskjet 560c "

  'Print the graph using the default printer.
  doc.Print()

  'Save the Graph with the existing filename.
  doc.Save()
```

Run the script directly in Grapher using the **Script Manager**.

In the following article, I have described opening and running a script that is available on our Free Scripts webpage that uses a Grapher template to create multiple plots with different datasets. This can be useful if you would like to create many similar graphs that each use data in a different data file or Excel worksheet. Example scripts are also included in the *Program Files* folder of Grapher 11, typical installed at *C:\Program Files\Golden Software\Grapher 11\Samples\Scripts.*

The example script described below opens a Grapher Template (*.grt) and allows the user to manually select the data files for each plot that they would like to create. The script reads the data files, creates a graph, and exports the graph to an Adobe PDF file. You can view, copy, or download the original script here. I have added some additional features to the script available on the website to illustrate some additional functionality that can be used via automation.

All of our example scripts begin with header comments that provide a general description of the script and what it is designed to do. These comments can be used to describe the purpose of the script, or outline what each section of the script is designed to do. Comments can be added in any line in the script by including an apostrophe at the beginning of the line to identify that line as containing a comment, for example:

```
'PRINT WITH DIALOG.BAS
'This script prints multiple graphs using a graph template and several data files.
'Press the Cancel button in the Open dialog box to end the loop.  Modify the
paths in the script to match your template file path and data file path.
```

Every script must contain a subroutine title `Sub` **`Main`**, which is used to setup the script and can define the main objects and arguments. In a simple script, this can act as the primary subroutine or you can call other subroutines that perform other arguments and statements throughout the script:

```
Sub Main

'All of your arguments, statements, or calls to other functions go within the
Main subroutine.

End Sub
```

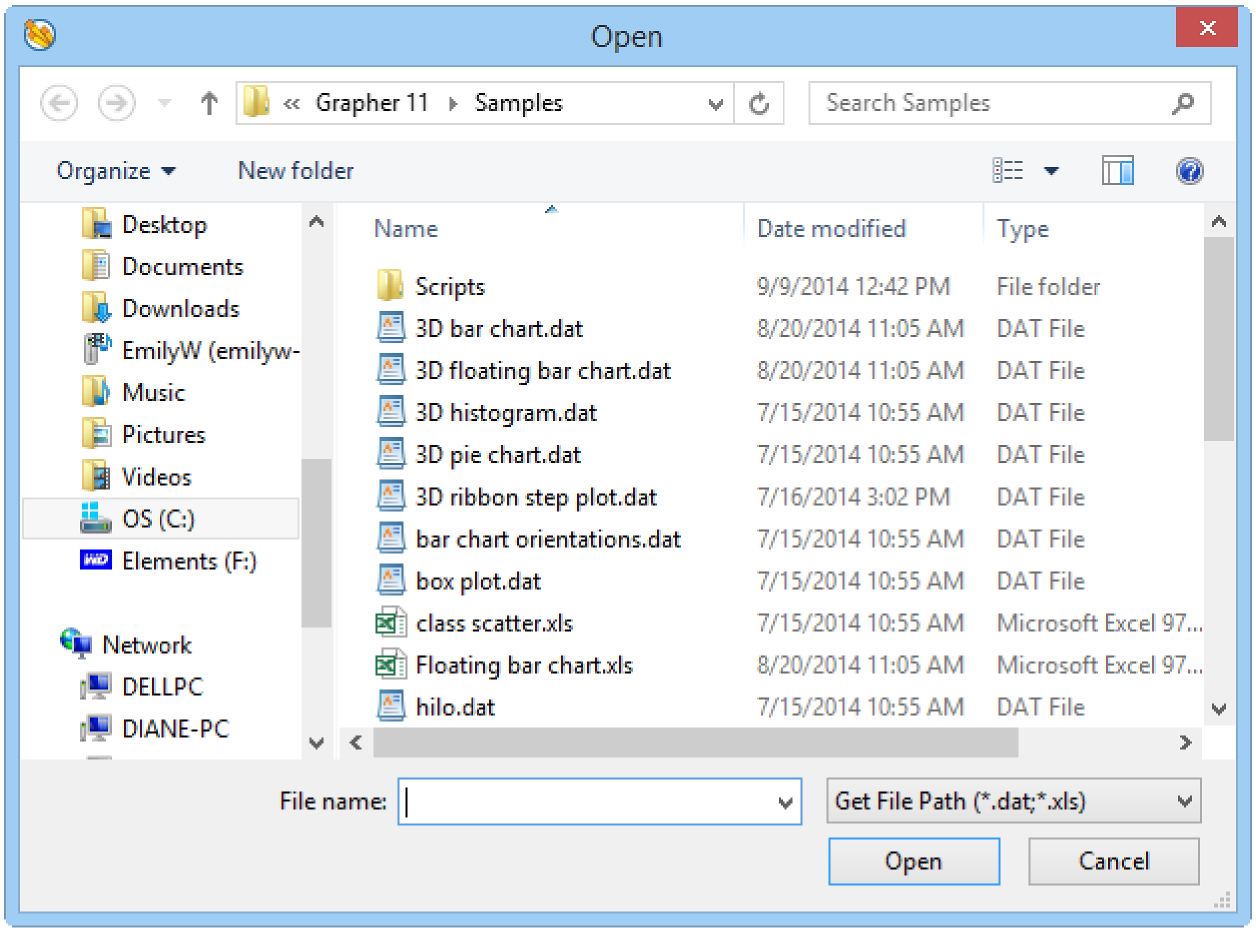
Within the **Main** subroutine, you need to declare the application object that you will use to run Grapher. First you must declare an object to be used to run Grapher. To declare an objection, you must use the `Dim` command as well as an object type:

```
  'Create Grapher as an object
  Dim Grapher As Object
```

After declaring objects that you would like to use in the script, you must call the Grapher application. Scripter uses many Visual BASIC compatible functions such as **CreateObject**. This is used to create a new instance of Grapher, where the remainder of your script will be carried out:

```
  'Start Grapher
  Set Grapher = CreateObject("Grapher.Application")
```

The next statement in the script uses the GetFilePath$() function to display the **Open** dialog. This will allow the user to navigate to a location and select a file for use by this script.



Name	Date modified	Type
Scripts	9/9/2014 12:42 PM	File folder
3D bar chart.dat	8/20/2014 11:05 AM	DAT File
3D floating bar chart.dat	8/20/2014 11:05 AM	DAT File
3D histogram.dat	7/15/2014 10:55 AM	DAT File
3D pie chart.dat	7/15/2014 10:55 AM	DAT File
3D ribbon step plot.dat	7/16/2014 3:02 PM	DAT File
bar chart orientations.dat	7/15/2014 10:55 AM	DAT File
box plot.dat	7/15/2014 10:55 AM	DAT File
class scatter.xls	7/15/2014 10:55 AM	Microsoft Excel 97...
Floating bar chart.xls	8/20/2014 11:05 AM	Microsoft Excel 97...
hilo.dat	7/15/2014 10:55 AM	DAT File

In the **Open** dialog, select the *.*dat* or *.*xls* data file you would like to use.

When a file is selected, the full path and filename for the selected file will be returned to the variable **`file$`**. This will then be used later in flow control for the script to open additional files:

```
'Open Grapher's Open dialog box
file$ = GetFilePath$(,"dat;xls",Grapher.Path+"\Samples\","Open",0)
```

When using the `GetFilePath$()` function, you can specify options for the files that you would like to open. This includes the file extensions for the types of files that you would like to open. For example, including the option string `"dat;xls"`, you can limit the types of files that can be opened to ASCII data files (*.dat) and Microsoft Excel files (*.xls). You can enter the full path for the filename, or if you are opening files in the Grapher directory, you can use the predefined `Grapher.Path` variable that is set in the Scripter options:

```
file$ = GetFilePath$(,"dat;xls","C:\Users\GrapherUser\Documents\","Open",0)
```

The next portion of the script performs a loop to open each of the DAT or XLS files that are in the designated directory. There are several flow control statements in the Scripter BASIC language that can be used to change repeat commands using different files or different variables. Scripter's BASIC language can use the following types of flow control statements:

- IF…ELSE… END IF

- SELECT CASE…END SELECT
- DO…LOOP
- WHILE…WEND
- FOR…NEXT
- FOR EACH…NEXT

The script will illustrate how to use the **WHILE…WEND** flow control statement to repeat the creation and saving of the plots. Using the **file$** variable that was defined earlier in the script, a **WHILE** statement can be used to perform a set of commands with a particular file that has been selected:

```
While file$ <> ""

'Include commands to perform within the script.

Wend
```

In this example script, for every ASCII data or Excel file that is opened, a Grapher Template (*.grt) is used to provide formatting for the graph created from each data file. The template provides all of the formatting for the plot, as well as the references for where to look in the file for the data, for example:

```
'Open the template file with data file selected in the Open dialog box
Set doc = Grapher.Documents.Add(0,Grapher.Path+"\Templates\line scatter
plot.grt",file$)
```

If the template is set up to read the data in specific columns in the data file, each file opened in the template will used data from the same columns (i.e. *X column = column A* and *Y column = column B*). The template is also used to define the formatting that will be used for the plots, as well including any drawing objects, titles or axis properties.

Though there are many commands that can be performed with the data while using the **WHILE…WEND** flow control, I will only highlight a few of the commands for basic file control. Below I have described how you can use the script to print, save, and close the current data file and plot window. The print, save, and close commands are all part of the **Documents** collection; therefore, they use the ***doc*** object that contained the reference for the Grapher template.

To print the graph, the `doc.Print()` command can be used to print the file using the default printer listed when using the **File** | **Print** command. You can also manually define the printer that you would like to use when printing from the selected Documents object. You can use the syntax shown below to define the printer manually in your script:

```
'Set the printer that you would like to use.
doc.PageSetup.printer = "HP Deskjet 560c "

'Print the graph using the default printer.
doc.Print()
```

You can also save your plot from the script so that any new graphs that have been created are saved for later use. Both the **File** | **Save** and **Save As** commands can be called using automation. The `doc.Save()` command will save the file with the existing file name, while the `doc.SaveAs()` command can be used to specify a new file name, location, or file type for the saved file. When using the `SaveAs` method, the file format should be:

```
'Save the Graph as a Grapher Project file using the SaveAs() Command.
doc.SaveAs("C:\Users\GrapherUser\Documents\new.gpj")

'Save the Graph with the Grapher 8 file file format using the SaveAs()
Command.
doc.SaveAs("C:\Users\GrapherUser\Documents\new.grf",grfversion8)
```

Once you have completed your script, you can use the `doc.Close()` command close the current document that is open in Grapher. Because you are working with the files in a loop, you will want to close the file that you created after you have saved the Grapher file.

After closing the current plot in Grapher, the script will continue through the loop to open the next data file in the folder. Again you will want to use the the `GetFilePath$()` function, to open the next data file in the directory. The **`WHILE…WEND`** loop that has been setup will continue until each of the data files in the directory has been opened in the template. When the `GetFilePath$()` function does not return any more data files, the loop will end and the next statement in the script will be executed.

Once the **`WHILE`** loop has been completed, the `Grapher.Quit()` command is used to close the current instance of Grapher. You will want to comment out this line if you wish to keep Grapher open to continue working:

```
'Close Grapher
Grapher.Quit()
```

Finally, you will want to include the `End Sub` statement to close the **Main** subroutine for this script. When you reach this line, your script is finished. Any files you have saved will be located in their respective directories.

This example script can help you to automate the opening, plotting, and saving of your data files with Grapher, so that you can streamline your workflow and create great graphs from many data files. Not only will using a script save you time, but a script will ensure that there is consistency between your files and can limit the number of edits that need to be made across each individual file.

If you would like to check out some additional scripts that can be used in Grapher, please visit the **Free Scripts** page on our website. These scripts have all been created as examples for our users. You are welcome to download and modify these scripts to suit your needs.

Hopefully, this has been an instructive and informative overview of using a script to improve your workflow in Grapher. If you have any questions about using a script in Grapher 11 to automate commands, or if you are having trouble with one of your scripts, please feel free to contact Golden Software Support.