ScriptWorX64
Getting Started

APPLICATION NOTE

August 2014

Description: Guide to getting started with ScriptWorX64.
General Requirement: Basic knowledge of VBA scripting.

Introduction

ScriptWorX64 is the server based scripting application in GENESIS64 suite. It is used to run VBA code at specified times or in reaction to certain triggers. It is a fully multi-threaded service designed to run 24 hours a day, ready to execute your scripts at a moment's notice.

This application note will give you some basic information about ScriptWorX64, walk you through creating your first ScriptWorX64 configuration, and give you some troubleshooting tips.

ScriptWorX64 does not get installed automatically with GENESIS64, but it can be found as a separate installation on the GENESIS64 disc.

Comparison with Other ICONICS Scripting Tools

GraphWorX64 Scripting – GraphWorX64 Scripting is based on the JScript.NET language, whereas ScriptWorX64 is based on VBA. Code cannot be copied and pasted between GraphWorX64 and ScriptWorX64.

It is recommended to use GraphWorX64 Scripting when the script needs to interact with or change display elements, or when the script requires interaction with a user. An example that would be a good fit for GraphWorX64 is a script that allows the user to select a particular alarm in an AlarmWorX64 Viewer and add a custom comment, because it requires user input.

ScriptWorX64 is recommended for situations where the code affects the system as a whole or requires no user interaction. Examples include a script that will monitor an OPC tags and when their value hasn't changed in an hour it will create a file in a specified location, or a script that will toggle an OPC tag every day at 8 AM.

ScriptWorX2006 or **ScriptWorX2010** – These are the 32-bit equivalents to ScriptWorX64. They are functionally identical.

ScriptWorX32 – ScriptWorX64 and ScriptWorX32 are both designed to execute VBA code, but ScriptWorX32 is an older version of the application and has fewer helpful features. For more details about the features of ScriptWorX64 that are new over ScriptWorX32, see the application note entitled, "ScriptWorX64 – Features in the ScriptWorX64".

Files used by ScriptWorX64

ScriptWorX64 uses three types of files, a database, a .vba file, and a .dll file.

Like most other ICONICS tools, ScriptWorX64 has a configuration database. This configuration database stores basic information about the configuration, such as the project structure and general settings. ScriptWorX64 can use either an Access database (.mdb) or a SQL Server database.

When you create a project a .vba file will be created. This file contains all of the code used in your application.

When you start a project for the first time or compile a project, a .dll file will be created. This .dll file is the compiled version of your code. It is used by ScriptWorX64 in runtime to execute the code. If the .dll file does not get made properly, or cannot be accessed by ScriptWorX64 for some reason then you will get an error when attempting to put ScriptWorX64 into runtime.

Creating a ScriptWorX64 Configuration

If ScriptWorX64 is not already installed it can be installed from your GENESIS64 DVD. It does not come pre-installed as part of GENESIS64.

1. Open the **ScriptWorX64 Configurator** from **Start → Programs → ICONICS ScriptWorX64**.
2. Go to **File → New** to create a new ScriptWorX64 configuration database. Step through the database creation wizard. Be sure to check the "Make database active" box.
3. Right-click on Projects and select **New → Project**. Name it *MyTestProject*, and type the same name into the Filename field. Click **Apply**.

   NOTE: A project is the topmost container for a ScriptWorX64 configuration. You should only have one project in a ScriptWorX64 configuration file. If more than one project exists, only one of the existing projects will run.

4. Right-click on MyTestProject and select **New → Designer/Thread**. Name it *MyTestDesigner* and click **Apply**.

   NOTE: A Designer/Thread is the container for your scripts that run on the same thread. Scripts in the same Designer/Thread can share code. You can have more than one Designer/Thread in a project.

5. Right-click on MyTestDesigner and select **New → Script**. Name it *MyTestScript*, and type the same name into the Script field.
6. Click the **"…"** button next to Trigger Name, go to the **UDM Data** tab, expand My Computer → Time Triggers and select the **Every 10 Sec** trigger. Click **OK** and then **Apply**.

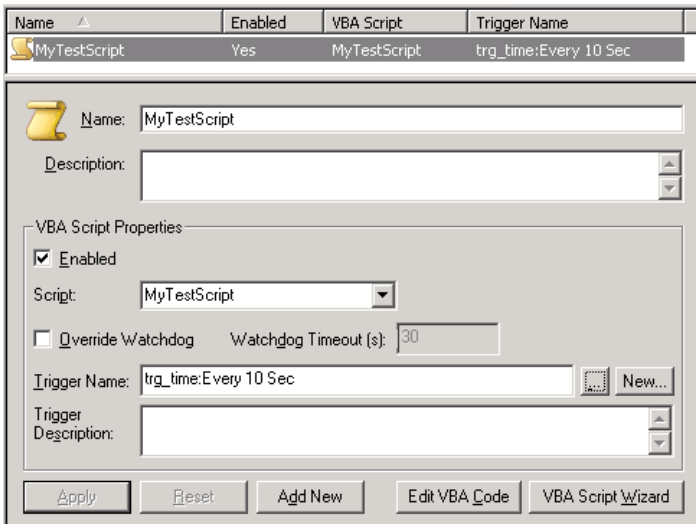

Figure 1 - MyTestScript object

7. Click **Edit VBA Code**. This will open the Microsoft Visual Basic Editor. It will also open the "MyTestDesigner" page of code.
8. Look for the "Public Sub MyTestScript" line. Two lines under it should be "' TODO: Add your code here". Delete that line, and add this code in its place:

```
Dim myValue, myQuality, myTimestamp

'Read the value, quality, and timestamp of a simulator tag.
myValue = g.OPC.Read("ICONICS.FwxServerOPC.1\@sim:Float.Ramp(10,-50.0,50.0,0).Value", myQuality, myTimestamp)

'Print the values to the console
g.ConsoleMsg MSG_INFORMATION, "MyTestScript", "Value: " & myValue
g.ConsoleMsg MSG_INFORMATION, "MyTestScript", "Quality: " & myQuality
g.ConsoleMsg MSG_INFORMATION, "MyTestScript", "Timestamp: " & myTimestamp
```

9. Hit the **Save** button in the toolbar of the Microsoft Visual Basic Editor. Close the editor.
10. Back in the ScriptWorX64 Configurator, click the stoplight so it turns green.
11. Go to **Start → Programs → ICONICS Tools → MonitorWorX**. (Do not launch the MonitorWorX Viewer under ICONICS Licensing – that's a different product.) This may take a moment to start, but it should add a MonitorWorX icon to your taskbar next to the clock and open the MonitorWorX window.
12. When the MonitorWorX window opens, go to the **Runtime** tab. Expand **ScriptWorX64 Runtime** and click on **Console**.
13. Wait at least 10 seconds for the script to trigger. When it does, you should see the value, quality, and timestamp of the simulator tag.

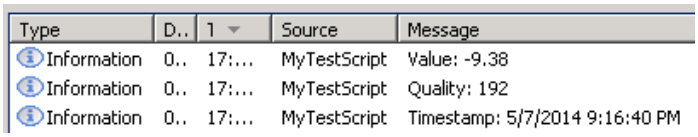

Figure 2 – MonitorWorX Console

Troubleshooting Tips

My message boxes don't appear, and my script freezes.

When ScriptWorX64 is running as a service or under a specific user account, not Interactive, it will not allow message boxes (MsgBox) to be visible to the user. However they still get invoked in the script and will cause the script to pause, waiting for user input. Since the user cannot see the box that user input will never come, and the script will freeze.

Instead of using a message box, write to the g.ConsoleMsg, and look in the MonitorWorX messages. See the "Files used by ScriptWorX64

ScriptWorX64 uses three types of files, a database, a .vba file, and a .dll file.

Like most other ICONICS tools, ScriptWorX64 has a configuration database. This configuration database stores basic information about the configuration, such as the project structure and general settings. ScriptWorX64 can use either an Access database (.mdb) or a SQL Server database.

When you create a project a .vba file will be created. This file contains all of the code used in your application.

When you start a project for the first time or compile a project, a .dll file will be created. This .dll file is the compiled version of your code. It is used by ScriptWorX64 in runtime to execute the code. If the .dll file does not get made properly, or cannot be accessed by ScriptWorX64 for some reason then you will get an error when attempting to put ScriptWorX64 into runtime.
Creating a ScriptWorX64 Configuration" section for an example of using g.ConsoleMsg.

My script ends too early. The MonitorWorX console says, "Watchdog time out has occurred…"

ScriptWorX64 contains a feature called the "Watchdog". This feature is designed to keep scripts from going into infinite loops. If a script takes longer to run than the specified watchdog timeout, ScriptWorX64 will forcefully end the script and post a message to the MonitorWorX console.

If this is happening to your script and you know the script is supposed to take a long time to run (such as a script that does a lot of sleeping) you can either increase the watchdog timeout or disable it altogether.

To increase the watchdog timeout, select either the Script object or the Designer/Thread object and change the Watchdog Timeout value. To disable the watchdog timeout altogether, select the Designer/Thread object and uncheck the "Has Watchdog" box.

> **NOTE:** It is not recommended to disable to watchdog timeout altogether, as this is a safety feature.

If your script is not supposed to run longer than the configured watchdog timeout and you are seeing this message you need to debug your script to find out why it is taking longer than expected.

My script isn't running.

Are you sure it isn't running? A lot of the time a script that appears to not be running actually is running and just not acting as expected.

To be absolutely sure if the script is running or not, go to **Start → Programs → ICONICS Tools → MonitorWorX**, go to the **Runtime** tab, expand **ScriptWorX64 Runtime → Threads**, and select the folder for your designer/thread. In here should be a count of how often your scripts have been executed. Wait for the trigger to occur and see if this count increases. If it does, your script has been successfully triggered, and something else inside the script must be going wrong.

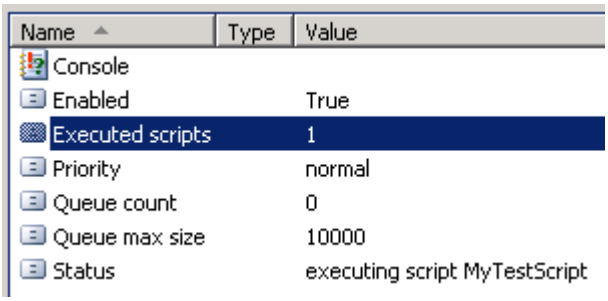

Figure 3 - Number of executed scripts

If you find out your script is not actually being triggered, check the following:

- Is the trigger configured correctly?
- Is there only one Project object in the ScriptWorX64 configuration?
- Is the "Enabled" box checked for the Designer/Thread object?
- Is the "Enabled" box checked for the Script object?

If your script is actually being triggered but not taking the actions you expect it to you need to debug your code. A good troubleshooting technique is to insert strategic places of your code and look for MonitorWorX Console, as described in the "Files used by ScriptWorX64

ScriptWorX64 uses three types of files, a database, a .vba file, and a .dll file.

Like most other ICONICS tools, ScriptWorX64 has a configuration database. This configuration database stores basic information about the configuration, such as the project structure and general settings. ScriptWorX64 can use either an Access database (.mdb) or a SQL Server database.

When you create a project a .vba file will be created. This file contains all of the code used in your application.

When you start a project for the first time or compile a project, a .dll file will be created. This .dll file is the compiled version of your code. It is used by ScriptWorX64 in runtime to execute the code. If the .dll file does not get made properly, or cannot be accessed by ScriptWorX64 for some reason then you will get an error when attempting to put ScriptWorX64 into runtime.

Creating a ScriptWorX64 Configuration" section. Do not use message boxes (MsgBox).

When I try to go into runtime, I get an error saying "Can't build VBA project."

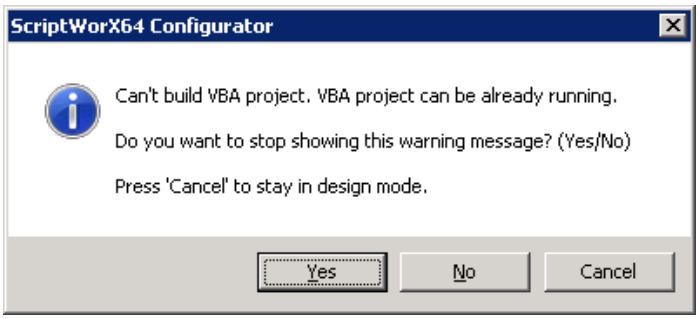

Figure 4 - Build error

If this error appears it usually means either that the ScriptWorX64 runtime engine didn't stop properly, or that you have errors in your code.

If the problem is that ScriptWorX64 Runtime didn't stop, go into Task Manager to the Processes tab and kill SwxRuntime.exe, then try to hit the stoplight again.

To check for errors in your code, open the VBA Editor (click the **Edit VBA Code** button) and go to **Debug → Compile *ProjectName***. Did this generate any errors? If so, correct them. If nothing happens when you select the Compile option it means no errors were found.

How can I use breakpoints?

Breakpoints are useful if you would like to stop the code at a particular point and check the values of certain variables.

Breakpoints in ScriptWorX64 work much the same as in any other product that uses the Microsoft Visual Basic Editor, but by default, ScriptWorX64 recompiles the VBA code when it enters runtime. This will remove any breakpoints you have set. To stop the programs from recompiling on runtime follow these steps:

1) From the main ScriptWorX64 configurator, go to **Tools → Options** and uncheck **When starting the Runtime** in the "Build VBA scripts settings" section.
2) Find your VBA .dll file and delete or rename it. By default it will be located in the same directory as your .vba file, configured at the Project level.
3) Open the Visual Basic Editor (**Alt-F11**, or find a script and hit **Edit VBA Code**) and go to **Run → Run Project**. Select **Wait for component to be created** then click **OK**.
4) Put a breakpoint in the code where you want to debug.
5) From the ScriptWorX64 main configurator screen press the Runtime Traffic light icon to start the script engine.
6) When your trigger occurs the breakpoint will stop the execution.

Please note that sometimes the Watchdog may kill your script thread while you are still debugging. See the above section entitled "My script ends too early…" for more information about the Watchdog and how to disable or extend it.

Can Technical Support help me troubleshoot my code?

A standard SupportWorX contract only allows technical support to troubleshoot a very small amount of code. For help troubleshooting large amounts of code, or even the writing of custom code, contact your regional sales manager or distributor and ask about Quality Professional Services.

If you think there is a problem with a specific method or if you have questions about the way a particular method is working the fastest way to get technical support assistance is to narrow down your code to as few lines as possible, or even come up with a separate short example script that demonstrates the problem or helps you illustrate your question, and does nothing else.

Additional Information

Discovering methods and properties

A valuable resource for scripting is the Object Browser feature of the Visual Basic Editor. Whether you are using VBScript or VBA, the Object Browser can be used to look up all the available methods for a given object.

1. Open the Visual Basic Editor by going to Tools → Macros → Visual Basic Editor or hitting Alt-F11.
2. Open the Object Browser by going to View → Object Browser, hitting F2, or clicking on the  button in the toolbar.
3. From here you can browse or search all of the classes and methods available to you in your current libraries. To limit your search to a specific library, select it from the top drop-down list. If a library you want to look at is not listed, you can add it by going to Tools → References.

Reading information from triggers

If you would like to read information from your trigger, such as which trigger was invoked (if you have multiple triggers), which alarm triggered an alarm trigger, the old and new value of an OPC tag for a data trigger, or many other bits of trigger info, see the application note entitled, "ScriptWorX64 – Reading Trigger Information".

The "g" object

At the top of every designer code window will be this declaration:

```
Dim g As SWXRuntimeLib.Global
```

The "g" object is used for many useful shortcuts, from writing to OPC tags to writing to the MonitorWorX console. To see what methods or properties are available for this object you can simply type "g." into the code designer and look at the IntelliSense dropdown window, or search the Object Browser.

How to make scripts pause or sleep

You can make your scripts sleep by adding this line to the top of the code window under the "Dim g" line:

```
Private Declare Sub Sleep Lib "kernel32" (ByVal
dwMilliseconds As Long)
```

(Note, the above code should be on one line.)

Then in your script where you want to sleep, call the Sleep method. It takes one parameter, which is the number of milliseconds you would like the script to sleep. For example, this subroutine will wait for 10 seconds, then log a console message:

```
Sub SleepTest

Sleep 10000
g.ConsoleMsg MSG_INFO_VERBOSE, "SleepTest", "Hello
World!"

End Sub
```

Note that the Watchdog will not realize your script is sleeping, and may kill it if it goes beyond the configured Watchdog timeout. See the earlier section entitled, "My script ends too early…"

Getting information from ScriptWorX64 into and out of GraphWorX64

ScriptWorX64 has the ability to create "global variables" which can be read by GraphWorX64 (and other OPC clients via the FrameWorX OPC-DA server). These variables can be used as OPC tags to communicate with running ScriptWorX64 scripts, or to provide output from scripts. They can also be used to communicate between two or more separately running scripts or threads.

See the application note entitled, "ScriptWorX64 – Easy Integration with OPC Enabled Global Variables" for more details.

More

See the other application notes starting with "ScriptWorX64" (or "ScriptWorX2010") for more information on things you can do with ScriptWorX64. You can also search our Knowledgebase for some common issues and their solutions.