ScriptWorX64
Features in the ScriptWorX64

APPLICATION NOTE

September 2013

Description: Guide to features in ScriptWorX64.
OS Requirement: Vista x64/ Windows 7 x64/ Windows 8 x64/ Windows Server 2008 x64/ Windows Server 2008 R2 x64/ Windows Server 2012
General Requirement: ScriptWorX64 and general knowledge of VBA coding.

Introduction

ScriptWorX64 is the server based scripting application in GENESIS64 suite. If you used ScriptWorX32 or ScriptWorX2010 in GENESIS32 you should be able to use ScriptWorX64 without any difficulties. ScriptWorX2010 and ScriptWorX64 are almost identical.

This application notes mainly focuses on new features of ScriptWorX64 in comparison with ScriptWorX32. It also describes conversion path between GENESIS32 and GENESIS64

Upgrading from GENESIS32 to GENESIS64

If you are converting your application from GENESIS32 to GENESIS64 you can follow the next paragraphs to convert your scripts.

- ScriptWorX2010 configuration can be loaded in ScriptWorX64 directly.
- ScriptWorX64 will not import or work with ScriptWorX32 *.swx files. Since the threading model, triggering mechanisms, variables and thread options are so vastly different between the two versions we could not make ScriptWorX32 configurations directly upgradeable. However, both versions are still based on VBA, you can certainly take code snippets from ScriptWorX32 and use them in ScriptWorX64 but many internal ScriptWorX32 functions will not be the same in ScriptWorX64.

New Threading Model

In ScriptWorX32, each time a script was triggered it would be executed in a new thread. The maximum number of threads allowed was specified in the settings, and if there were no threads available then scripts would queue up and wait for a thread to become free. This could sometimes cause scripts to execute in an unpredictable order.

In ScriptWorX64, scripts are organized into VBA designers, and each designer runs in its own thread. The number of designers in your project specifies the number of threads available. If multiple scripts are triggered within a thread, the scripts will queue up within the thread. The limit of queuing scripts is specified on a per-designer basis, as is the behavior when this limit is exceeded. This new threading model gives the user much more control over the threads used by ScriptWorX64. This is a major advantage over ScriptWorX32.

New Configurator Layout

Configurator has been completely reorganized. in ScriptWorX64 to better accommodate the new threading model.

The new hierarchy is Project → Designer/Thread → Script.

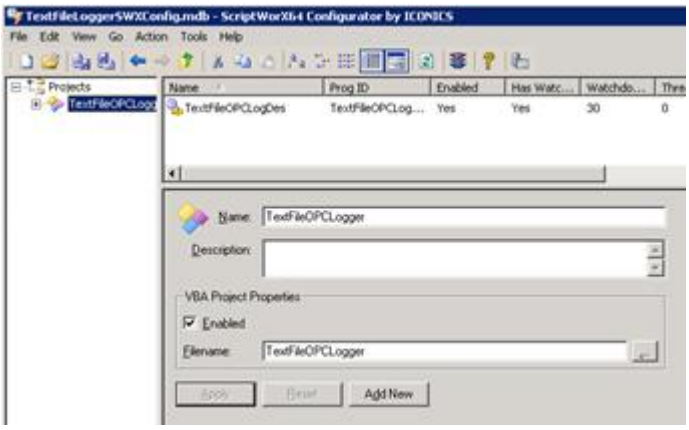

Figure 1 – ScriptWorX64 Configurator

Projects are associated with ".vba" files that contain all of the code for the project.

Within a project are **Designer/Threads**. The .vba file for a project is split into multiple designers, each of which is associated with their own thread in runtime. Any functions, subroutines, and global variables within a designer are available to all scripts in that designer.

Scripts link a particular subroutine within the designer to a trigger created with the **Unified Data Manager**. Whenever the trigger happens for a Script, the associated subroutine will enter the queue for the Designer/Thread.

ScriptWorX64 Global Variables

- Are available to all scripts within a Designer/Thread.
- The global variables can be used to share, watch, and change their values by each script.
- Any script can change the value of a global variable that is used in another script. You can use a global variable

to get the total count of scripts executed within a Designer/Thread.
- Are available to OPC clients as OPC data through **GenBroker** Plug-in)
- Support the OnVarChange events, which can catch any change in any global variable value. The event is activated by scanning two parameters: the global variable name and the thread ID of the script. The information is used to deduce if changes of the global variable value were executed from another thread.

We have introduced the idea of **Thread identification** as an input parameter. Having thread IDs aids in identifying which thread has executed a ScriptWorX64 script. For example, you can set up a condition to decipher if the script is called from the same Designer/Thread or from another thread.

We have added new **Events**. You can run your script on a time period using OnTimer function. Also, when a global variable changes, OnVarChange is activated, from which you can call your script.

Trigger Data introduces the following advantages when a script is executed by firing a trigger:
- Passing the trigger data to the triggered script as an input parameter (i.e. "Public Sub NewScript(td As TriggerData)"
- Triggers are managed by Unified Data Manager application which is distributed with ScriptWorX64

Designer/Thread Options

Every Designer/Thread has the following options in ScriptWorX64:

Priority: Each Designer/Thread can be given one of 7 priorities (Realtime, High, Above Normal, Normal, Below Normal, Low, and Idle), allowing the user to better manage when the threads are given processor time.

Startup Squelch: ScriptWorX64 will prevent the thread from executing within a given number of seconds after ScriptWorX64 is started. A squelch of zero indicates the thread may start as soon as possible.

Timer: A timer can be created for each Designer/Thread. You can use it as a simple way to time events within a script without re-triggering the entire script.

WatchDog: The WatchDog prevents scripts from running out of control. If a WatchDog is enabled for a Designer/Thread, any scripts within the Designer/Thread that run longer than the specified number of seconds will be killed. Individual scripts can override the WatchDog if needed.

Processing Queue: If multiple scripts within a Designer/Thread are triggered to run at the same time, they will queue up. The queue can be configured to hold only a certain amount of scripts and to either ignore new scripts or drop the old ones when it reaches the limit.

ScriptWorX64 and OPC-DA

ScriptWorX64 has much more built-in functionality when working with OPC-DA tags.

- VBA global variables are exposed as OPC-DA tags, which are available to any ICONICS client such as GraphWorX64 or AlarmWorX64.
- The status of a thread is also exposed as OPC-DA tags. OPC clients can access the length of the queue, the remaining time in the WatchDog, how many times a script was executed, etc.
- ScriptWorX64 has built-in functions to read OPC-DA tags.

Copyright 2013 ICONICS, Inc.