Tech Note 806
Introduction to Using Asynchronous Scripts in the Wonderware Application Server

Topic#: 002593
Created: November 2011

Introduction

Asynchronous scripting mode is a group of scripts running on the same, lower priority execution thread. These scripts only support Execute triggering and run independently from each other. This *Tech Note* covers some of the frequently-asked questions regarding Asynchronous scripting.

Application Versions

- Wonderware Application Server 3.1 and later.

Questions and Answers

1. *If a script exceeds the execution timeout period, does it get executed completely?*

   The script does not complete if it exceeds the timeout period and it would be an indeterminate situation.

2. *What happens to the rest of the scripts and logic that are supposed to run when a script times out. What about "On True" and "On False" conditions? Will these conditions (one-shots) execute during the next scan? What happens to all of the other logic that was supposed to execute this scan? What happens to the I/O updates?*

   In all cases, when a script times out, then it aborts. So, some lines of the script may have been executed and other lines of the script not executed. If an "On True" or "On False" script times out, then it will not be retried. It is a one-shot execution.
   Following are the two techniques for avoiding script timeouts.
   1. Put Objects with long running scripts into a separate Engine to better facilitate large script timeouts for certain scripts.
   2. Use Asynchronous scripts, which have a separate script timeout setting.

3. *Do we need to put in some type of interlock in the script trigger to keep asynchronous scripts from launching multiple instances?*

   No you do not need to create interlocks to prevent asynchronous scripts from launching multiple instances. This is already built in with the product.

4. *How many asynchronous scripts can be active on an Engine at any given moment in time?*

   There is a setting on the Engine called **Maximum asynchronous thread count**. This is the maximum number of asynchronous scripts that can be running on an Engine at a particular time.

5. *If the Engine's settings for* ***Maximum asynchronous thread count*** *is 10, what happens if the 11th asynchronous script is launched while the existing 10 asynchronous are still running?*

   The 11th script must wait until one of the other existing asynchronous script is finished.

6. *Is there any limit to the number of asynchronous scripts that can be configured within an Object? (platform, Engine or Object limit)*

   The maximum number of asynchronous scripts that can be running at a time for an Engine is **1000**. This setting is located in the Engine Tab and the Engine Configuration **General** tab.

   The maximum number of scripts that can be defined in an Object is 50. This means you could have 20 Objects, each having 50 asynchronous scripts, running under an Engine.

7. *How can I determine the number of asynchronous scripts that are currently being executed on an Engine?*

   Engine.AsyncScriptsRunnningCnt provides the total number of asynchronous scripts that are currently executing on the Object.

8. *How to determine the number of asynchronous scripts that are waiting to be executed on an Engine?*

   Engine.AsyncScriptsWaitingCnt provides the total number of asynchronous scripts that are currently queued to execute on the Object, but are not yet executing because they are waiting for a free thread.

9. *Does the execution of an asynchronous script have any effect on the overall execution time of the Engine? (other than normal CPU load).*

   No. The scan period will still be used to execute the synchronous scripts and will not be affected by the asynchronous scripts.

10. *Does the execution count get bumped up at beginning or end of an asynchronous script?*

    The **ExecutionCnt** UDA for a given script will only increment once the script has been fully executed – even if completing the script spans several Engine scan cycles.

11. *Does each asynchronous script execute on its own individual thread or is there some other mechanism that is used?*

    Each asynchronous script is executed on its own individual thread.

12. *Can an asynchronous script be triggered again while it is still running?*

    Asynchronous script WILL NOT spawn multiple threads of the same script, if one instance of the script is already running.

13. *An asynchronous script can be configured as "periodic". What happens when the period is shorter than the execution time of the script?*

    The asynchronous script will run to completion even if the execution time exceeds the period. It will not run to completion if it exceeds the asynchronous script timeout limit, which is a setting in the asynchronous script configuration.

14. *We have experienced some problems with asynchronous scripts getting hung in a busy state (script.State = Busy). Why doesn't the script ever timeout?*

This may occur if the script calls a .NET function or a custom DLL and does not return.

15. *When dealing with UDAs used inside of an asynchronous script, should we treat them like local or remote references. Let us assume that the UDA has no extensions.*

    *Me.UDADouble_1 = 1.0 + ((Me.UDADouble_2 - 60.0) * Me.UDAConst);*
    Me.UDADouble_1 = Round(Me.UDADouble_1, 0.0001);

    • Is Me.UDADouble_1 going to update as expected or is there a delay between when the value is written and when the value(s) is usable?

    • How do the variables changed inside an asynchronous script get communicated back to the Galaxy?

    In this example, Me.UDADouble_1 will get updated as expected. In other words, the first statement will immediately assign the value to Me.UDADouble_1. Then it can be used in the second statement. This is not true if the attribute on the left side of the equation is on another Engine. In that case, it will take a scan or two for the assignment to take place and then be usable later.

P. Karthikeyan

Back to top