Simio
Forward Thinking

Simio API Note: SimEngine Controller

Last Update: 02 April 2021

Contents

Simio
Forward Thinking

SimEngine Controller and SimEngine Interfaces

Overview

The SimEngine Controller and Interfaces provide a more plug-and-play approach to get you started faster and with fewer programming requirements.

The SimEngine Controller is a .NET process waits for run requests that must be delivered in the SimioEngineRequest format.

The SimEngineInterface* projects are example programs that show how you can interface with the SimEngine controller in various ways. For example, the SimEngineInterfaceFileDrop program shows how a simple C# program could be constructed that consumes text files and launches the SimEngine Controller to run experiments. The other SimEngine interface programs are similar, including a SimEngineInterfaceFileDrop.py Python program that does the same.

Running the SimEngine Controller

The SimEngine Controller is merely a wrapper around the Simio API commands that run the bare Simio Simulation Engine.

The SimEngine Controller requires only four settings that are all full paths to folders. These paths must be created prior to running the Controller. For convenience, the path can begin with a meta-folder that points to one of the “special” Windows folders. For example, the default starts with <ApplicationData> which is a special folder that points to the users folder c:\users\yourUserName\AppData\Roaming.

The four settings are:

1. A path to where specially formatted Request files (in JSON format) are placed. Default is <ApplicationData>\SimioEngineController\Requests
2. The path to where all files (e.g. DLLs) needed to run the Simio Engine are placed. Default is <ApplicationData>\SimioEngineController\DLLs
3. The path to where all Simio project that are referenced in the Requests reside. Default is <ApplicationData>\SimioEngineController\Projects
4. The path to where logging of activity is written (default is <ApplicationData>\SimioEngineController\Logs.

These requests are delivered by a SimEngine Interface which uses both the SimioEngineInterfaceHelper library (DLL) to format requests according to what the SimEngine Controller requires, as well as the

Simio
Forward Thinking

SimEngineLibrary to execute Simio Experiments and Plans. You can use these libraries directly in your program or employ one of the examples already provided.

- Text FileDrop: provide the request by dropping a simple text file into a folder.
- MQTT: provide the request with an MQTT request message

Installation

You can install the SimEngineController directly to your computer by:

??

In order to run, it will require that you:

1. Create the four folders mentioned above
2. You must "harvest" special files from your Simio desktop installation to the Controller's DLLs folder mentioned above. This is done with the SimioApiHelper utility.
3. You must also place any Simio project files (e.g. .SPFX files) that you wish to run in the Controller's Projects folder.

An Example

To test the installation, we'll employ a very simple example, which is an Simio project that runs an Experiment and is included in all Simio desktop installations.

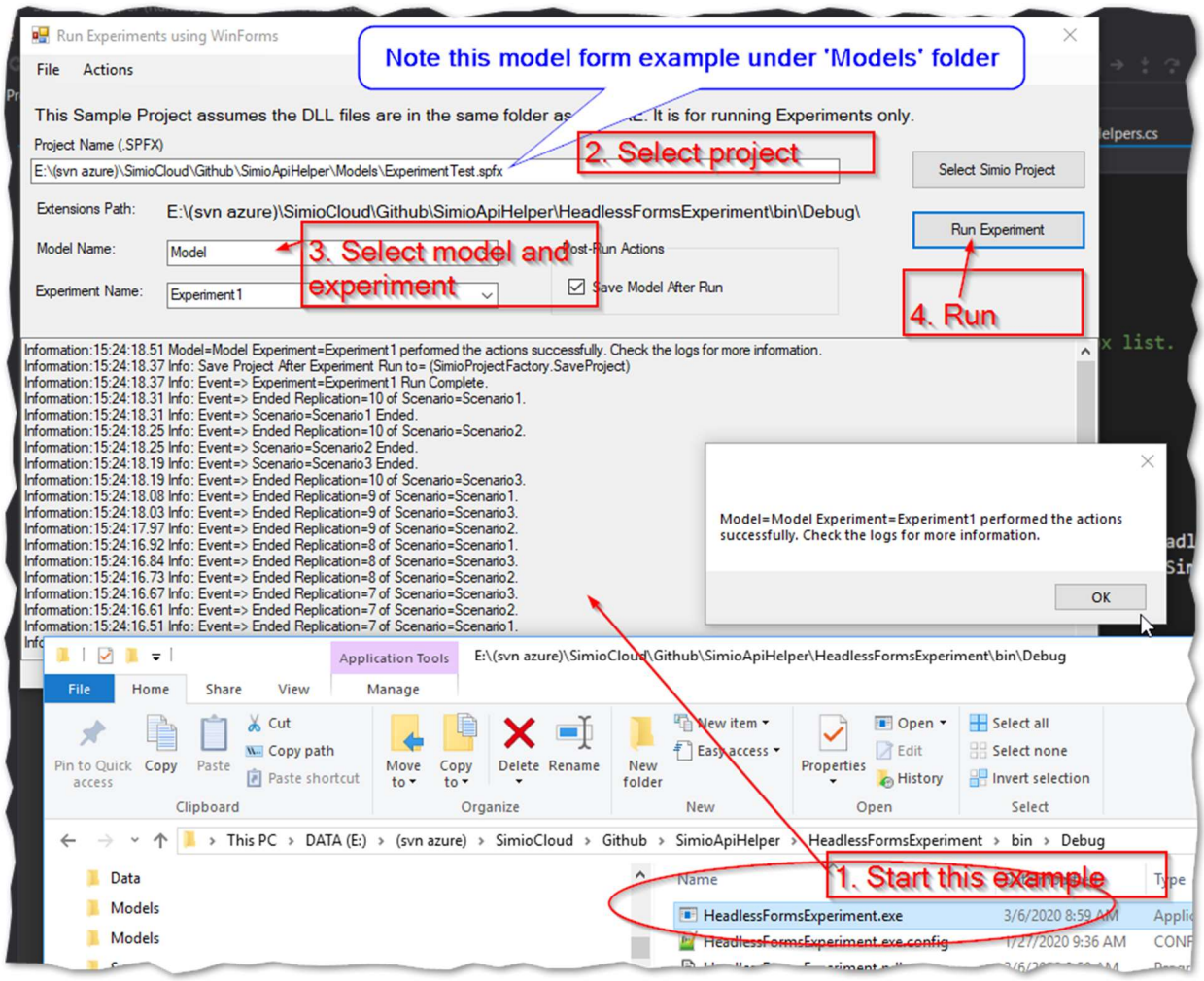
Simio
Forward Thinking

Appendix: SimEngine Debugging

One of the hardest things to determine is what DLLs are required, and/or what the dependencies between the DLLs is.

There are two free tools that can help with this:

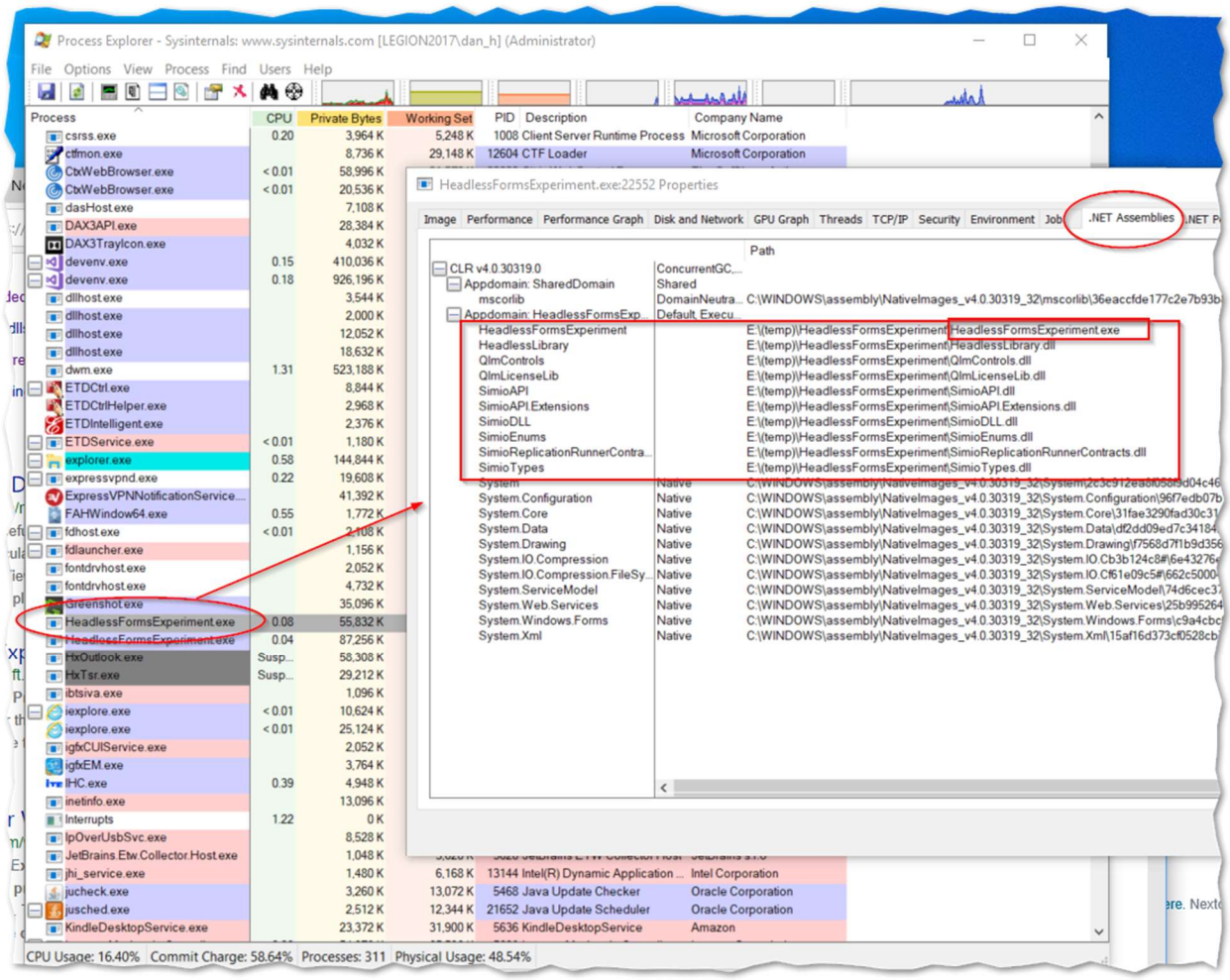
1. Process Explorer from SysInternals (Microsoft) to examine DLL dependencies
2. DotnetPeek from JetBrains to examine assemblies (such as DLLs)



Simio
Forward Thinking

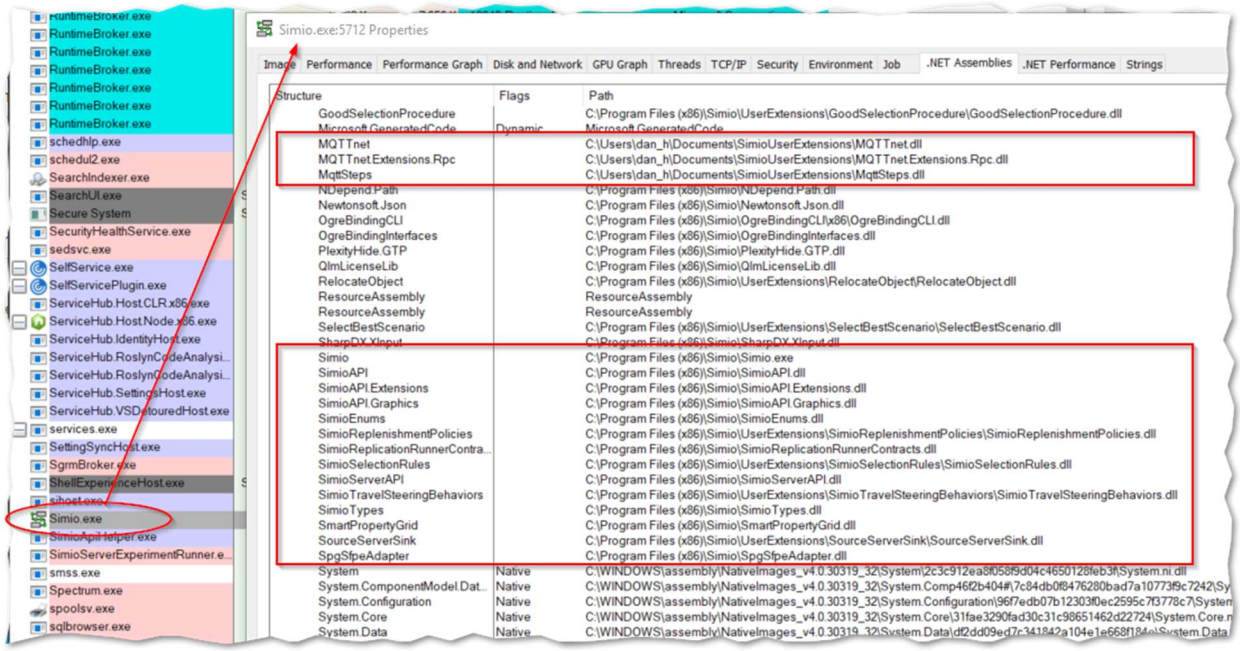
Process Explorer can be used to examine a running program. This is incredibly useful because we can see what DLLs are employed regardless of when they were loaded. It can be downloaded for free from:

https://docs.microsoft.com/en-us/sysinternals/downloads/process-explorer



So, in the example above the included DLLs are shown.

Below is Process Explorer being applied against Simio with the same model being run.



And below is the result of one of the example projects “SimEngineFormsExperiment” which uses the call “System.Appdomain.CurrentDomain.BaseDirectory” to pick up the location of the SimEngineFormsExperiment.exe to locate all of the DLLs.

Appendix – Simio Licensing (Server and Node-Locked)

This appendix describes some of the issues surrounding Simio licensing. When running in bare SimEngine mode, Simio of course will still require a license, so it is important to make sure that your program is able to properly find its license. Machine based licensing (also called Node-Locked) is usually without problems, as the license information is simply on the machine. The difficulties usually arise with Server licensing, where the machine must locate the License Server on the network. The usual suspects of network location and firewalls then come into play.

Server Licensing

There are currently two types of Service licenses in play, as Simio transitions from RLM licensing to QLM licensing. Each is discussed in turn.

RLM Licensing

The best documentation about RLM Server licensing can be found in this document:

Here are some key points:

The licenses are stored on the License Server under ProgramData > Simio LLC > Simio Network Licensing.

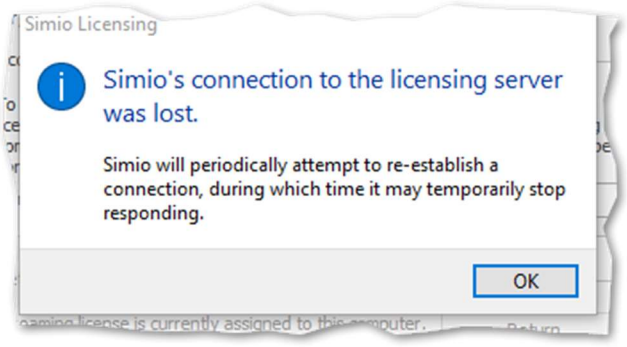
If there is a file SimioConfig.lic, it should hold configuration information about the “random” port used by the license simulator.

The files rlm.dlog and simio.dlog hold debugging information.

The files that have the extension “.LIC” hold the licenses.

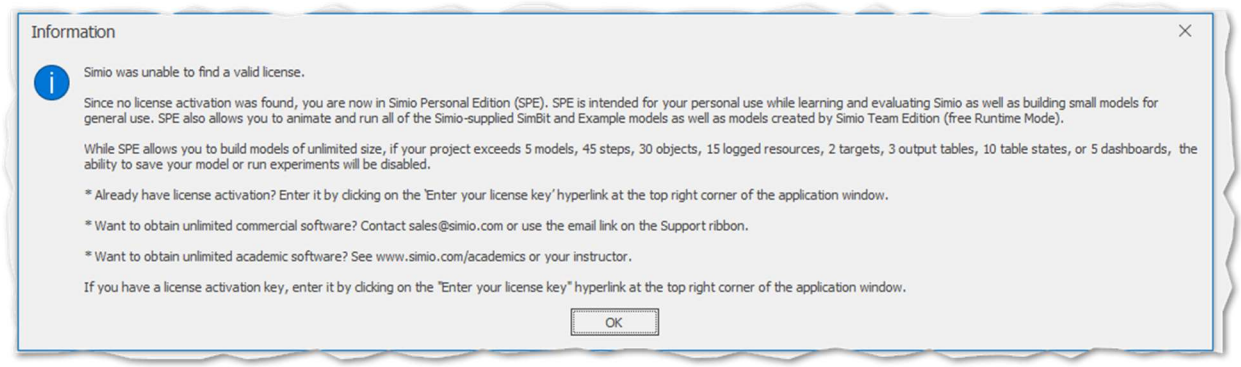
The Service on the License Server that runs licensing is named “RLM Simio”. It obviously must be running.

If it is stopped, the desktop Simio programs will raise a message box:



If the Server has no licenses, you get the message:

Simio
Forward Thinking



For more information, please see the document “Simio Network License Server.pdf” that accompanies this document.