# Simio API Note: SimEngine Controller

Last Update: 02 April 2021

Contents

[Simio API Note: SimEngine Controller 1](#_Toc68433962)

[SimEngine Controller and SimEngine Interfaces 2](#_Toc68433963)

[Overview 2](#_Toc68433964)

[Running the SimEngine Controller 2](#_Toc68433965)

[Getting Started 3](#_Toc68433966)

[Some Examples 4](#_Toc68433967)

[Appendix: SimEngine Debugging 5](#_Toc68433968)

[Appendix – Simio Licensing (Server and Node-Locked) 8](#_Toc68433969)

[Server Licensing 8](#_Toc68433970)

[RLM Licensing 8](#_Toc68433971)

# 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 that waits for run requests to run Simio models. These requests that must be delivered as a specially formatted JSON file to a dedicated folder. The Controller can be completely installed by the provided Windows Install msi file.

The Controller employs the same SimioFactory methods to run Simio Experiments and Plans as any other Simio product (such as Desktop or Portal). The code is provided for you, but you can simply deploy the SimEngine Controller as a binary (EXE). You can invoke the Controller using the SimEngineController.exe from a console, or it can also be installed as a Windows service.

As of this writing, the Controller uses Simio version 12.220.

There are three other folders you need to know about.

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

As mentioned, 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>\SimEngineController\Requests
2. The path to where all files (e.g. DLLs) needed to run the Simio Engine are placed. Default is <ApplicationData>\SimEngineController\DLLs
3. The path to where all Simio project that are referenced in the Requests reside. Default is <ApplicationData>\SimEngineController\Projects
4. The path to where logging activity is written (default is <ApplicationData>\SimEngineController\Logs.

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

However, if you have the skill, there is nothing that prevents you from formatting your requests as these JSON formatted files and dropping them into the Requests folder yourself.

## Getting Started

Before you send a request, you must do the following:

1. Put any Simio Project that you are going to run into the Controller’s Projects folder.

*Why do we need this?* Because the requests only include the name of the Simio Project, such as “MyExperimentSample1.spfx”. It is assumed that this model already exists in the Projects folder.

1. Harvest all of the DLLs that your project needs into the Controller’s Extensions folder*.*

*Why do we need this?* Because when the Simio Engine runs, it requires all sorts of DLLs. These include – for example – all user extensions and their supporting files. For example, custom Steps, custom design-time Add-Ins, etc. This can most easily be done using the SimioApiHelper, which has a tab specifically designed for this ‘harvesting’ task.

To make this easier, the Controller will attempt to do a harvest on startup if you provide it with a full path to where your Simio product was installed in the setting SimioInstallPath. For example, this might be:

C:\Program Files (x86)\Simio

1. Make sure your Simio product is licensed correctly. For example, if you are going to run Plans, make sure you have a Simio license that permits this. The easiest way to do this is to simply attempt to run the Project on a machine that has Simio desktop and verify that it runs correctly.

Text

Description automatically generated

## Some Examples

To test the installation, we’ll employ some very simple examples, which are Simio projects that runs simple Experiments and Plans which are included in all Simio desktop installations.

# 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)

A screenshot of a computer

Description automatically generated

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>

A screenshot of a computer

Description automatically generated

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

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

A screenshot of a computer

Description automatically generated

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:

A screenshot of a cell phone

Description automatically generated

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

A screenshot of a social media post

Description automatically generated

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