Simio
Forward Thinking

Simio API Note: Simio API Helper

May 2018 (Dhouck)

Contents

Simio
Forward Thinking

Overview

This API Note describes a utility that can:

1. Examine/test the DLLs that are used by the APIs.
2. Construct sample “headless” folders
3. Test the operation of the headless methods.

This is a very brief note, as this utility was quickly assembled and being put in place to help a few Simio users debug some API issues.

The intent is that this helper will expand/extend as needed.

Simio
Forward Thinking

Using The API Helper.

The Simio API Tester is not part of the Simio product. Rather it is simply a test tool that was built to help debug problems.

You can use it to verify that your DLLs are:

1. In the right place
2. Is accessible
3. Has the Interface implemented
4. Referencing other Assemblies correctly

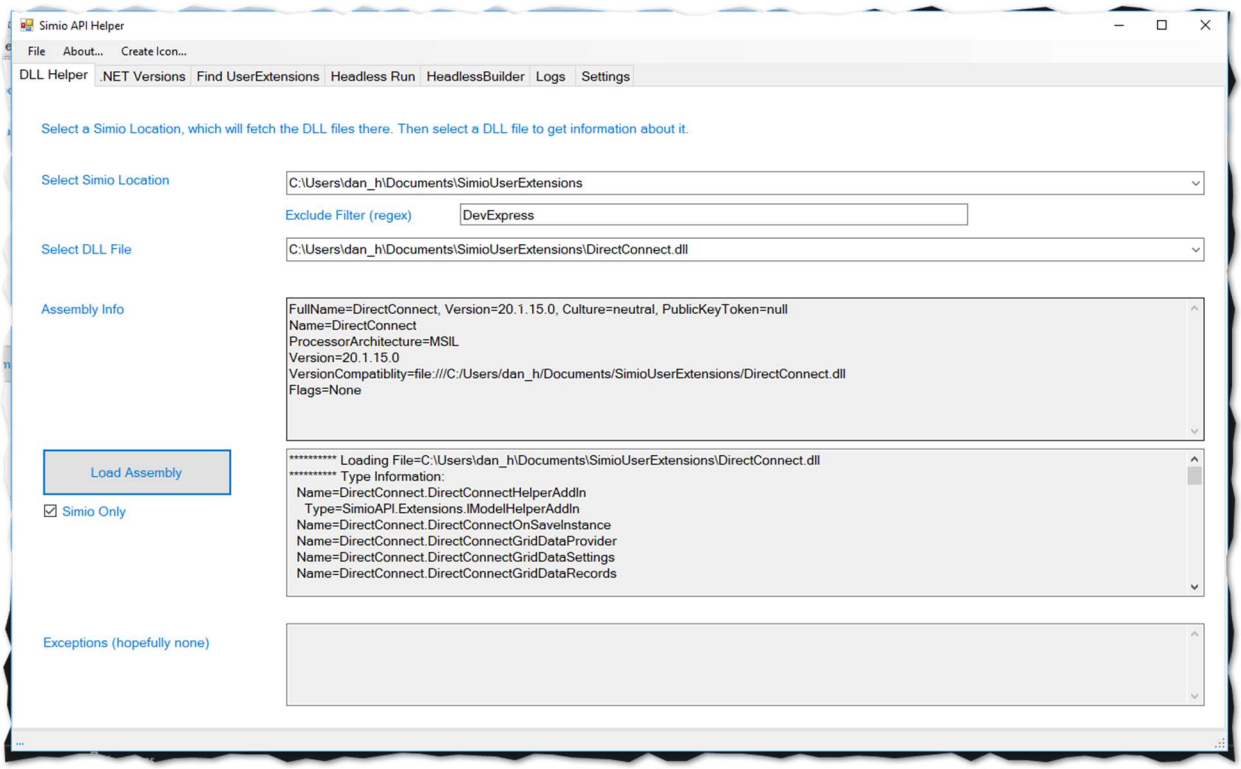
The UI architecture is a simple WinForms tabs, and the sections below are organized according to those tabs.

Tab DLL Helper

This tab is used to examine and load DLL assemblies.

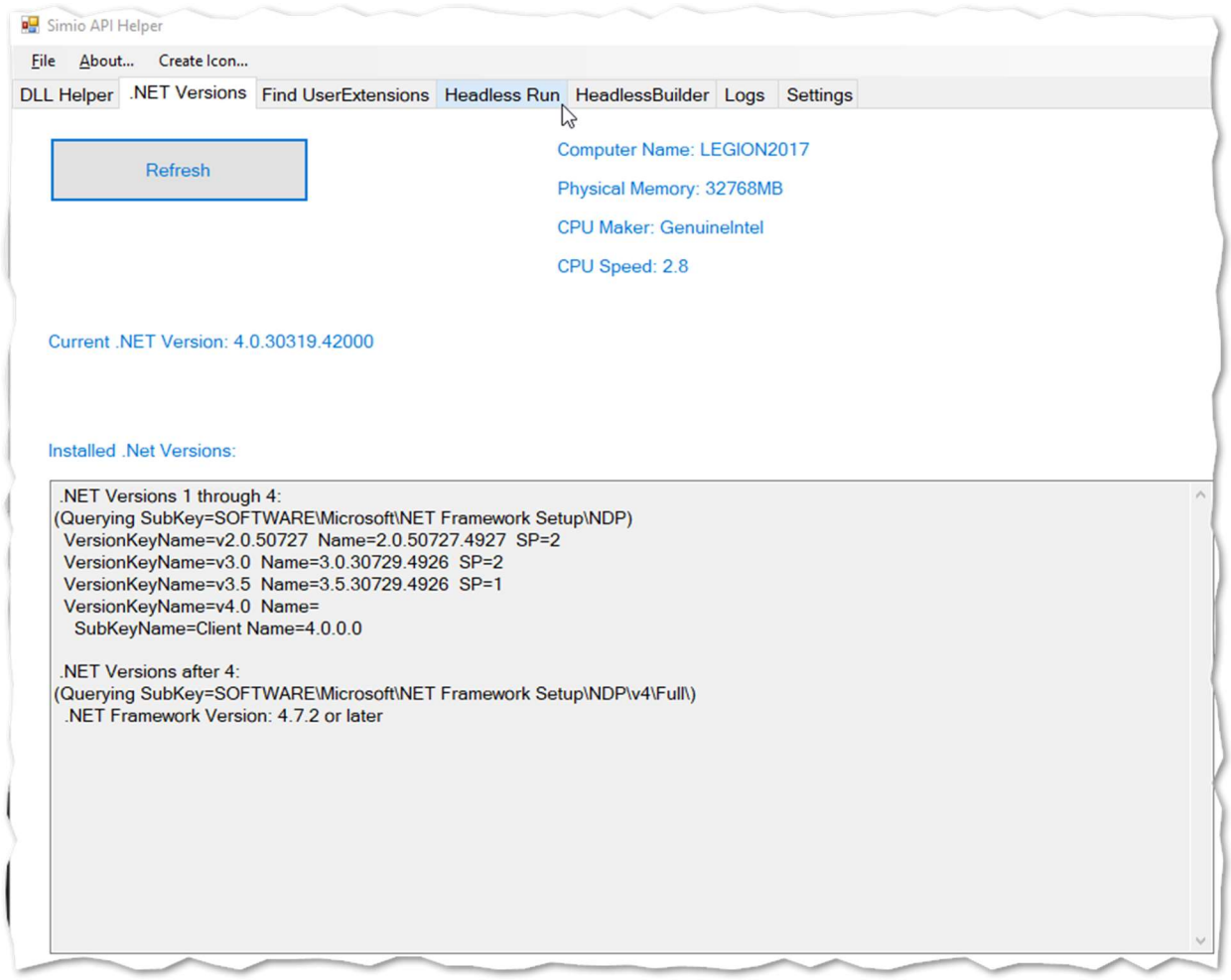
The top drop-down displays the locations where Simio DLLs can be found, and the next drop-down then shows the DLL files within that location. The Exclude filter can be used to reduce the number of DLL files displayed.

Once a file is chosen, general information about the contents of the DLL is shown, along with the definitions found within the file. If you wish to see only Simio information, check the “Simio Only” checkbox.



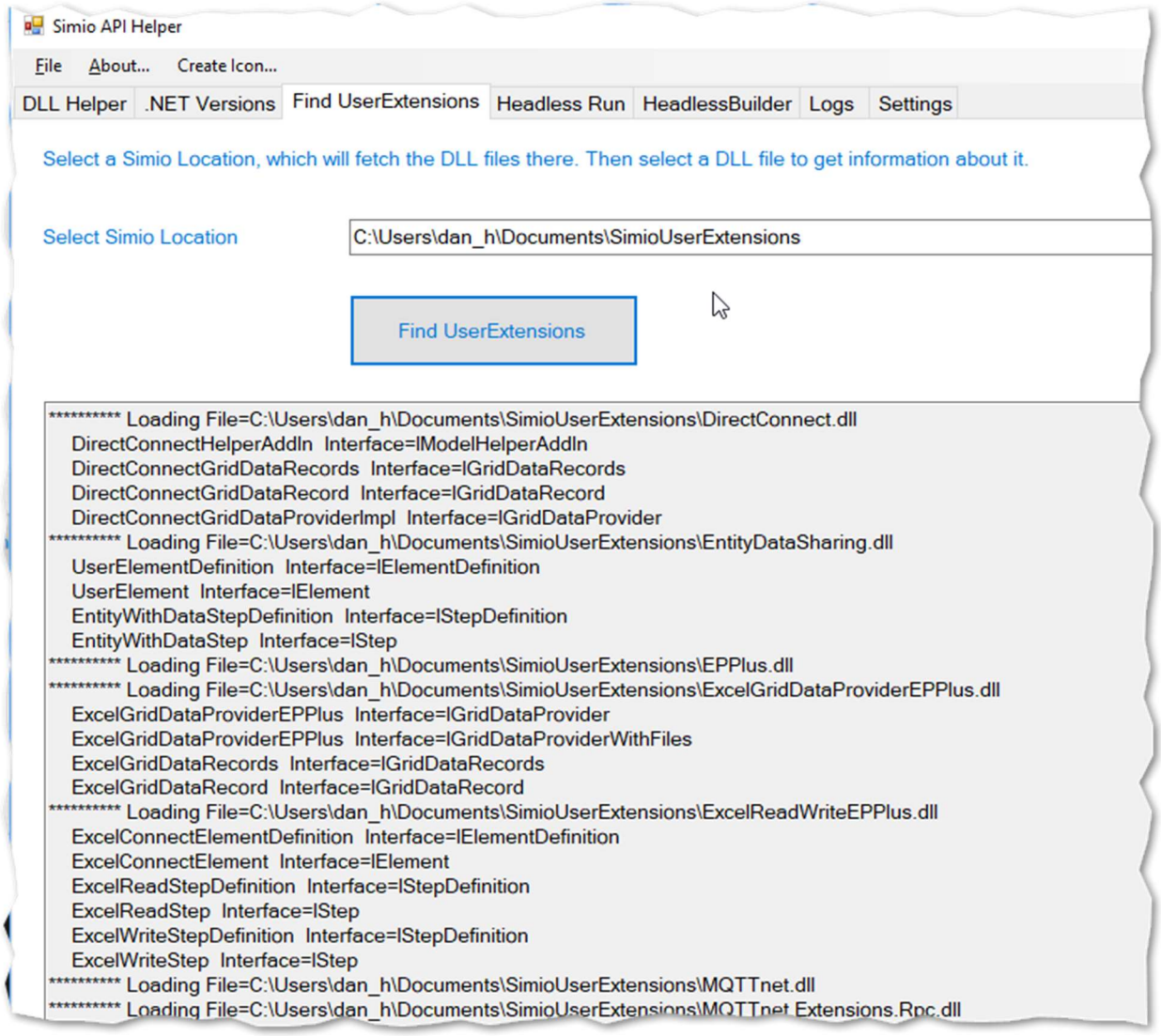
Simio
Forward Thinking

Tab .Net Versions



Simio
Forward Thinking

Tab Find User Extensions



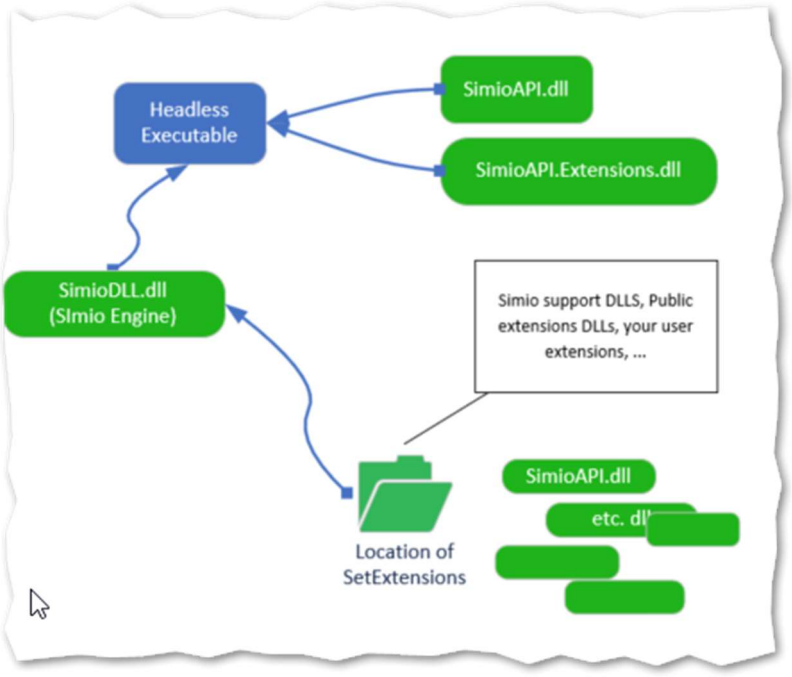
Simio
Forward Thinking

Headless Workflow and Recommendations

When you are running Simio from desktop it is not obvious that you are running the Simio Engine with the Simio UI. This is convenient for designing, building, and interacting with your model. However there are several use cases for running your model without the Simio UI using your own custom Headless executable.

Since Simio is data driven, a common scenario is to have a headless application wait for new data, and then load your Simio model (which binds to your data), run the model and then write the resulting data out to another database. Simio has a range of APIs to assist this with this headless mode, and this API helper will describe how to construct such applications, as well as providing utilities and sample projects.

Achieving success when building a headless application often depends upon selecting and using the correct components (DLLs). It is a bit confusing because Simio is very modularized and determining which DLLs to use can be a confusing task. It is further confused because the executable that you build will reference Simio API DLLs to load the Simio Engine, and then the engine will in turn require more DLLs.



First off, a recommendation: put all your components (your custom application and the Simio DLLs into a separate folder). In theory you could use the installation folder, but this would mean that every time you update Simio you will have to re-test your headless application. And perhaps more importantly, when the Simio Engine is running it looks for DLLs (such as custom user extensions) in many places.

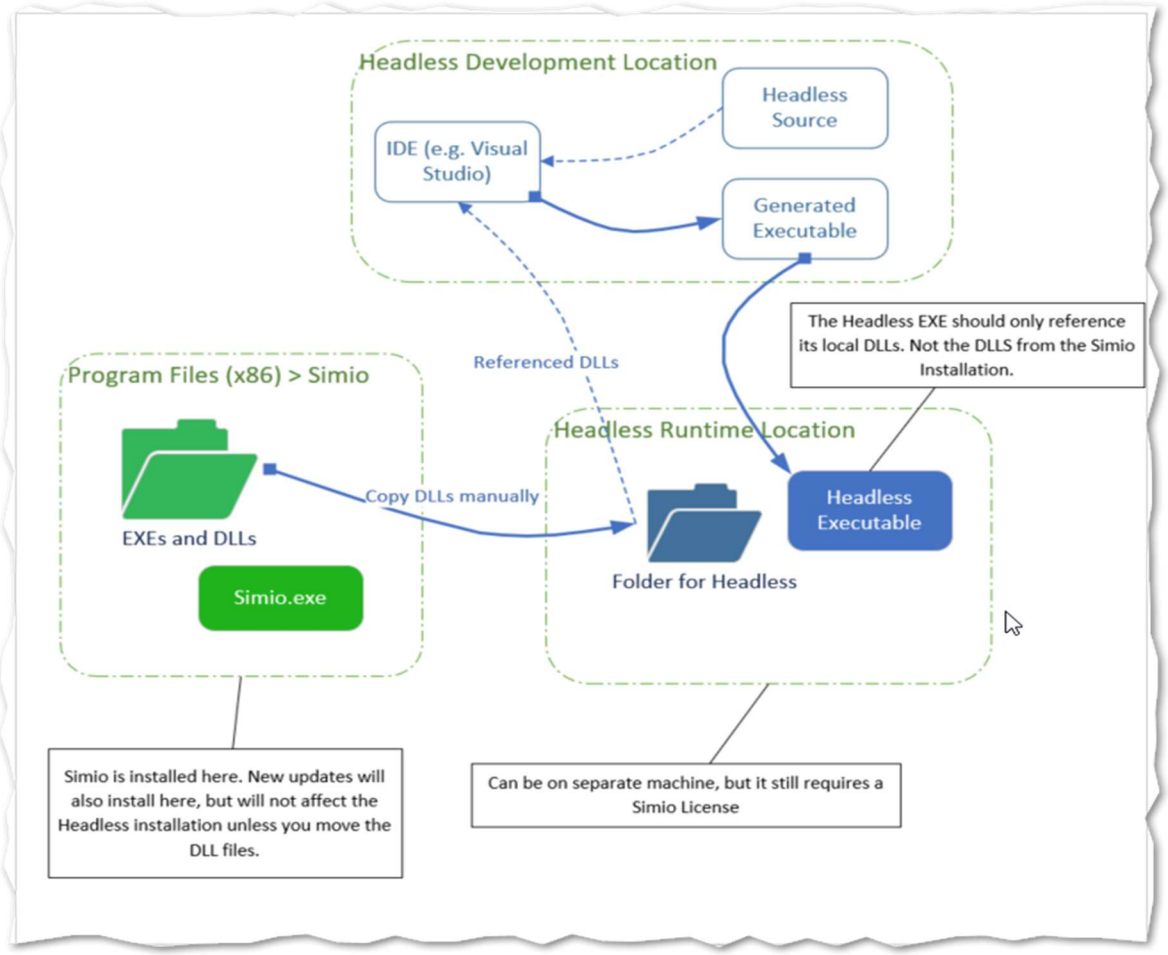
When you are using headless this searching does not occur! So, you would have to move all the DLLs that your application uses into the Simio installation folder, causing unnecessary clutter and confusion.

Note that the Simio API Helper has a utility that can help you harvest these DLLs and put copies in a folder of your choice.

This has the advantage of protecting your application (which are often production oriented) from updates or upgrades to the Simio software, and you have all of your dependent DLLs in a convenient package for backing up.

Additionally, when you are building your application (e.g. with Visual Studio) you should reference the DLLs from your headless environment folder (and **not** the installation folder) since this is what your generated executable will expect.

The diagram below illustrates this workflow.



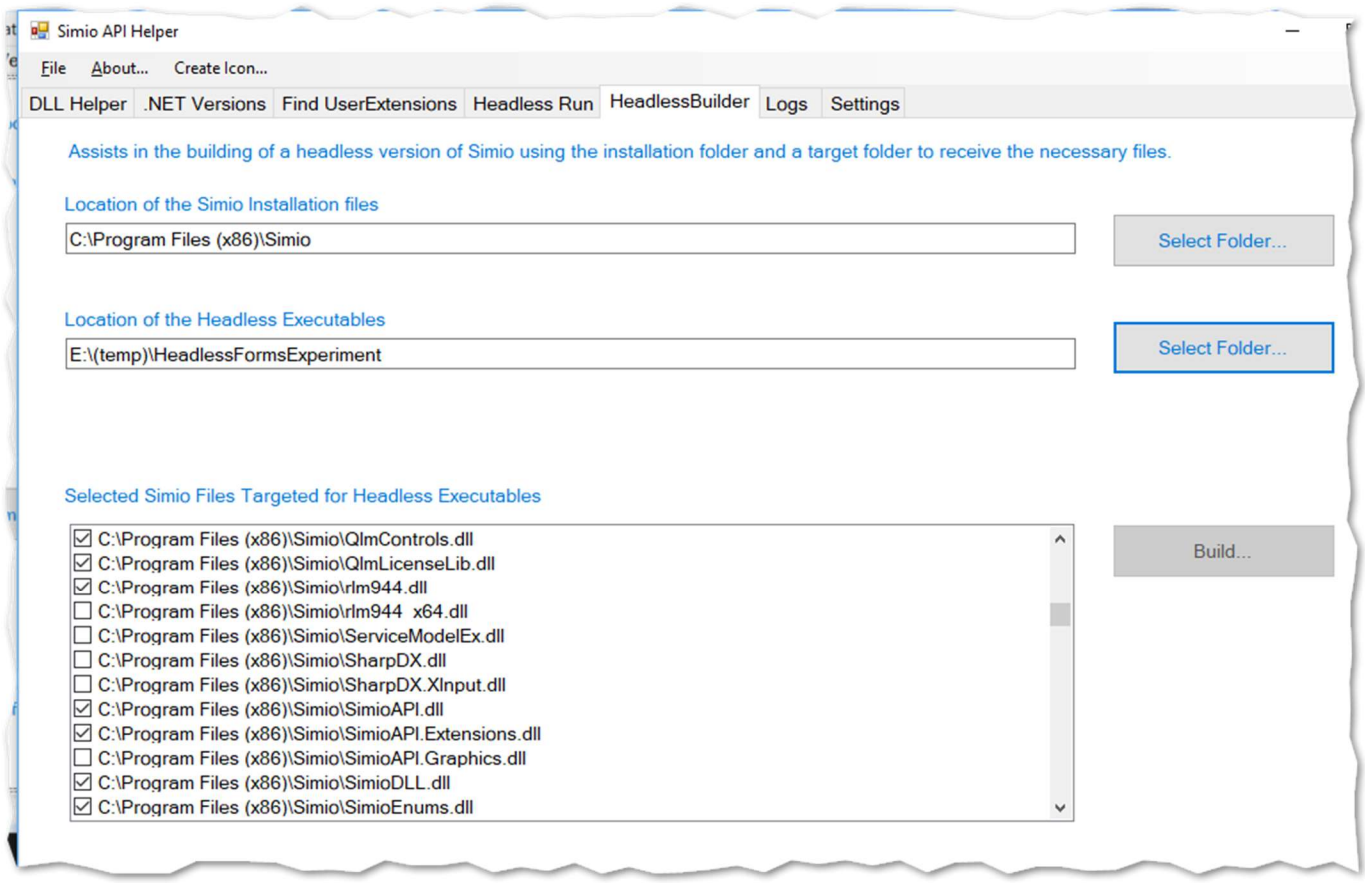
The actual code is quite simple and of the form:

1. Set the extensions folder (where to look for the DLLs)
2. Load the Model (and check for errors)
3. Run the Experiment or Plan for the model.
4. Optionally Save the results

The most common error is to omit Step 1, which tells the Simio Engine where to find its DLLs.

Simio
Forward Thinking

Tabs Headless Builder and Headless Run



Simio
Forward Thinking

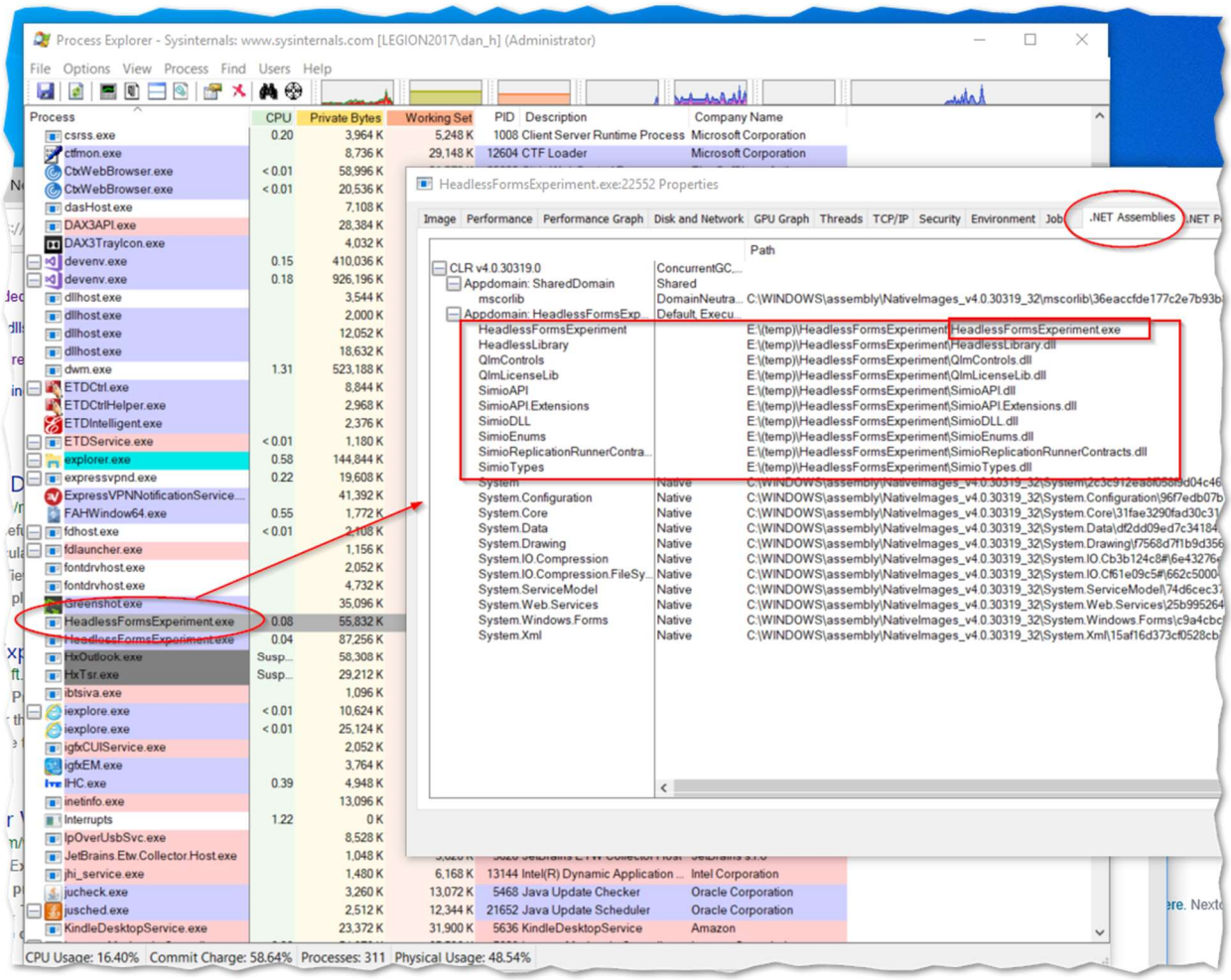
Headless Debugging

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)
2. DotnetPeek from JetBrains

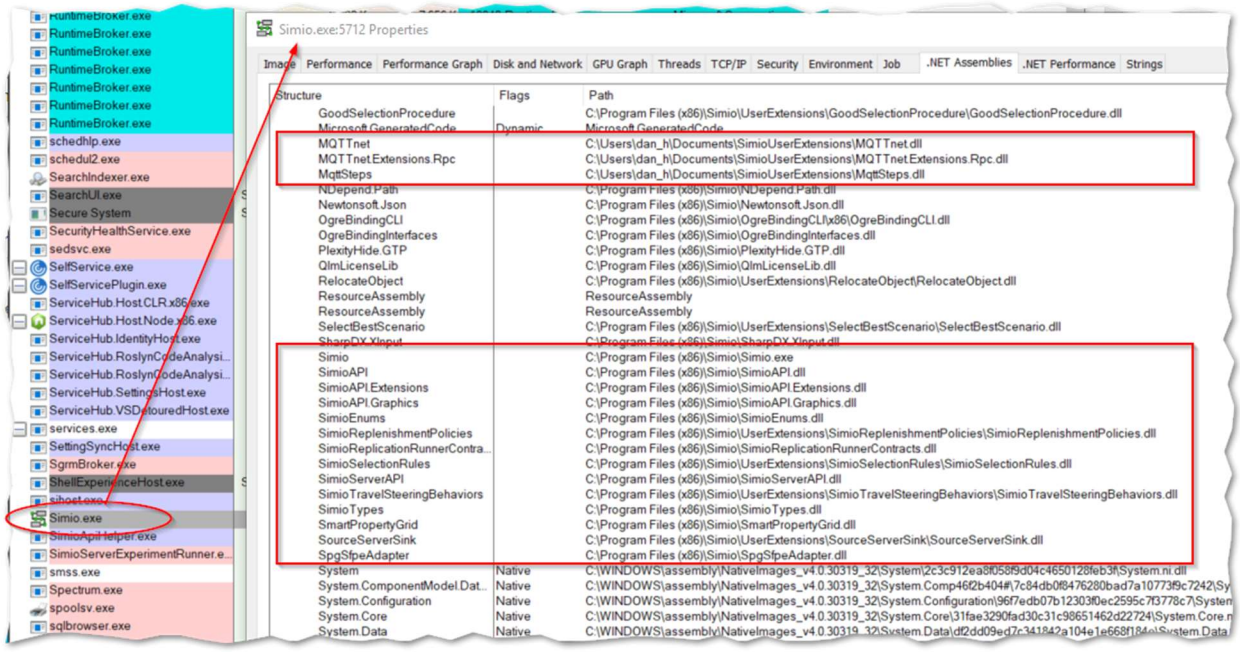
ProcessExplorer can be used to examine a running program. This is incredibly useful because we can see what DLLs are employed regardless of when they were loaded.



A

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

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



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

Simio
Forward Thinking

Tab Logs and Settings

Simio
Forward Thinking

Installing the Simio API Helper

The helper is just an EXE (SimioApiHelper.exe) and is built with .NET 4.5.2.

There are no other dependencies (e.g. DLLs) beyond .NET.