# Parameter Locations

There are two places where parameters can be set for the SimWebServer. The first is a file that overrides parameters that are set internally. This is very similar to Sim.properties. The second is the “.ini” file that SimWebServer reads and is in a different place. The only value that can be adjusted here is the amount of memory that SimWebServer is to use.

We will discuss the two sets of parameters separately.

# SimCaseManager.properties

In the install directory, there is a directory called “data.” This directory comes with files “CopyOfSimJarSim.properties,” “CopyOfJarFileLog4j.properties,” and “CopyOfJarFileSimCaseManager.properties.” These are all text files and can be edited with any text editor.

These 3 files are ignored by Sim and SimWebServer. They are put there only because a user might want to override the parameters. To override parameters found in SimCaseManager.properties, an administrator should rename “CopyOfJarFileSimCaseManager.properties” to “SimCaseManager.properties.” There’s no need to worry about “destroying” “CopyOfJarFileSimCaseManager.properties.” Not only do Sim and SimWebServer not use this file, but it will be recreated the next time either program runs anyway.

After renaming “CopyOfJarFileSimCaseManager.properties” to “SimCaseManager.properties,” the administrator can override any parameter found in SimCaseManager.properties. For parameters that the administrator does *not* wish to override, the administrator can leave the line alone, comment it out by putting a ‘#’ in the first column of that line, or delete the line entirely.

As an aside, the above 3 paragraphs apply to the other “CopyOf” properties files that appear in the data directory.

## Parameters in SimCaseManager.properties

There are 6 parameters in SimCaseManager.properties. To configure a heavy-use system such as SAR School’s server, it is useful to understand these parameters and probably edit them. For a single use system, the defaults will usually suffice.

### MaxNParticleFiles

The administrator can probably leave MaxNParticleFiles alone for the “heavy duty” cases, but we describe it here for completeness. When SimWebServer runs a simulation, it writes the particle file to the disc and caches it in internal memory. When Sim or SimWebServer needs a particle file for a Planner run (or any step in the Planner work flow), It checks if there is an appropriate one in the cache. If there is and the date of the one on the disc (if it exists) is not more recent, it simply uses the one in the cache.

This parameter controls how many particle files can be cached. If we set this parameter to a large number, we speed up Planner’s work flow, but we will need more memory (See Section 3; configuring the .ini file).

### MaxNProcessorsToUse

The rest of the parameters *will* require some experimentation for a heavy-use case. MaxNProcessorsToUse is the trickiest to understand and the other parameters rely heavily on its value.

The default is computed as follows: A java call retrieves the number of cpus’s that appear in the task manager’s “cpu usage history,” and subtracts one. Roughly speaking, this is a measure of the computer’s “concurrency” ability. But this number can be overridden with a smaller number (we might not want to claim all the cpu’s in a very large machine) or a larger number (if the computer has very little concurrency available).

Therefore, this number is initially *related* but not exactly equal to the number of cores of the computer.

### MaxNEngines

SimWebServer can run “cases” simultaneously. A case is either a Simulator case or a Planner case. Each case is run in an “Engine,” and the Engines are set aside when SimWebServer is started up.

“MaxNEngines” is the most number of engines that will be set up when SimWebServer is started up. If, at some point, more cases (Sim or Planner) are scheduled to run on this SimWebServer, the remaining ones will be queued, waiting for an engine to become available. The number of engines will not be greater than MaxNEngines, but it might be less. We discuss how SimWebServer computes the number of engines after we discuss each parameter.

### MinNThreadsPerEngine

SimWebServer will not set up more engines than MaxNProcessorsToUse. In fact, SimWebServer will not set up more engines than MaxNProcessorsToUse/MinNThreadsPerEngine.

If one is feeling that the machine is powerful, the operating system is reasonably intelligent, and that there is very little other load for this machine, he can set MaxNEngines high and MinNThreadsPerEngine to 1. However, that could create a lot of engines and there might be a lot of engines self-destructively competing for the “worker threads” (discussed below that Sim and Planner use during the individual cases’ parallel processing.

### MaxNThreadsInOneCall

SimWebServer starts with MaxNProcessorsToUse threads, and allocates one for each Engine. The rest go into a “worker thread” pool that is used by individual cases during the execution of the case. Both the Simulator and the Planner make extensive use of parallel processing and use the worker threads from this pool for their parallel processing.

When a Simulator or Planner case enters a part of its algorithm where it does parallel processing, it asks for up to MaxNThreadsInOneCall threads that it can use. For example, when doing something to all of the particles, the Simulator will ask for a set of threads; say it gets *n.* It will then split the particles into (*n*+1) sets and assign a set to each thread, and take one set for itself. Only when all of the threads are done with their sets will the Simulator continue with the next step.

This parameter keeps an Engine from grabbing all available worker threads. If it did, another Engine would have to wait until the threads became available.

## Determination of the Number of Engines that SimWebServer sets up

With the above parameters set up, the determination of the number of Engines is easy. It must be at least 1 and can be no bigger than MaxNEngines. Other than that, NEngines is simply MaxNProcessorsToUse/MinNThreadsPerEngine. The number of threads in the worker thread pool must be at least 2 and, other than that, is simply MaxNProcessorsToUse – Nengines.

# SimWebServer.ini

In the main “install directory,” there is a SimWebServer.ini. This is a text file that can be edited. One value in that file that can be modified is the amount of memory that the virtual machine running SimWebServer will claim. That is the number following the “Xmx” and typically refers to the number of megabytes. For example, -Xmx1150M means that the virtual machine will use up to 1150 Megabytes. 1150 Megabytes is approximately the limit for a 32-bit Windows machine, but multiple engines and multiple cached particle files) require more memory. The setting in the 64-bit distribution is currently 8000M. That should suffice for up to 10 engines running.