# Scenario 06 – Custom application hanging

The objectives of this scenario is to highlight the use of Snap. Scenario 06 requires the student to use Snap in different ways. The second lab in this scenario will have them do analysis of real dump files (anonymized) to show how problematic code paths can easily be found.

## Origin

The idea for this scenario came from an engagement with a customer, experiencing that most often during peak load, the SharePoint front page would hang and users would get timeouts. Recycling the AppPool would have them run for a short period before the problem reoccurred.

With Snap we took process dumps of the AppPool while the problem occurred, and we could quickly identify it to be a Logging framework with a lock statement and a server side fetch of JSON resource, where the service was not responding fast enough, and thus creating hanging threads.

## Prerequisite Lessons

For the students to work with this scenario they should have been through the **Snap** lessons.

## Lab 1 Tasks

By leveraging demonstrated troubleshooting tools, the students should complete the following steps as part of this troubleshooting scenario:

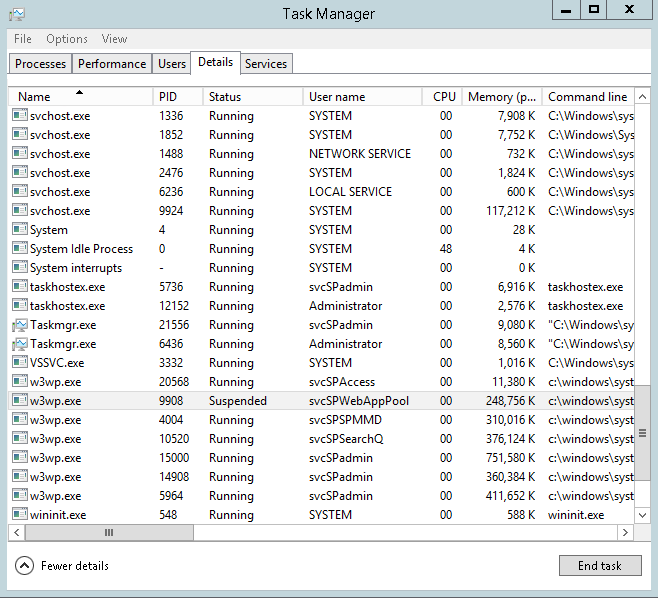
1. Take a few process dumps of the <http://intranet.contoso.com> AppPool process
2. Analyze the dump files – what running methods do you see?
3. Take an emon dump of the <http://intranet.contoso.com> AppPool process
4. Analyze the emon file – what are the most common exception?
5. Are there any specific part of SharePoint that has more exceptions?
6. Take a memory dump of the <http://intranet.contoso.com> AppPool process
7. What special “Prod” precautions did you take in this and the previous actions?

## Lab 1 Answers and key discussion points

This section highlights some of the key discussion points of the scenario

### Answers

1. …
2. This can be varying, but most likely you will see methods for Taxonomy, PerformanceCounter, and Object Cache.
3. …
4. This is also varying. Usually there are some Timeout exceptions from the Service App layer and some IO errors from the Config Cache
5. Could be config cache, but again – this varies
6. …
7. For process dumps, some CPU and Disk IO load is applied. For emon dumps, the process will be killed on exit – so this should be carefully planned in Production (watch Task Manager when exiting. The process disappears and reappear with a new PID). For memory dumps, the process will be suspended, so users will just “wait”. This should also be carefully planned in Production! (see picture below)



## Lab 2 Tasks

By leveraging demonstrated troubleshooting tools, the students should complete the following steps as part of this troubleshooting scenario:

1. Do you see any pattern in the log files?
2. What part(s) of the custom code should the developers revisit?
3. What options could the developers have considered in order to avoid the hang?

## Lab 2 Answers and key discussion points

This section highlights some of the key discussion points of the scenario

### Answers

In the first few Log files, there are many processes hanging in the Contoso.News.Logging.LoggingService constructor, where it needs to get an instance of the Practices logger. This is in a lock statement and all threads are just blocked.

In all the files, there are a huge amount of threads waiting for Network resources. In particular, the Branding methods are waiting for JSON results from an external source.

### Alternative code patterns

If the developers know that there is a chance of intermediate long running operations, then several patterns can be applied. The code can be run async client side – this way the page will load and the result of this control will just be delayed.

Caching the result will also ensure that only few users are affected.

Another option could be to limit the time it can run – e.g. 1 second, and then return a message like “This operation is taking longer than expected, please return back later to try again” to the user.

There is also an opportunity to do some of this client side, so the App does not appear to be “hanging”.