Common reasons why your application pool may unexpectedly recycle

Authored by: ***ASPHostServer Administrator*** *[asphostserver@gmail.com]*
Saved From: *http://faq.asphosthelpdesk.com/article.php?id=129*

If your application crashes, hangs and deadlocks it will cause/require the application pool to recycle in order to be resolved, but sometimes your application pool inexplicably recycles for no obvious reason. This is usually a configuration issue ordue to the fact that you're performing file system operations in the application directory.

The following is the most common reasons for an application pool to recycle.

- Recycle worker processes (in minutes)
- Recycle worker process (in requests)
- Recycle worker processes at the following times
- Maximum virtual memory
- Maximum used memory

These settings should be pretty self explanatory, but if you want to read more, please take a look atthis MSDN article

- memoryLimit
- requestLimit
- timeout

memoryLimit
The default value of memoryLimit is 60. This value is only of interest if you have fairly little memory on a 32 bit machine. 60 stands for 60% of total system memory. So if you have 1 GB of memory the worker process will automatically restart once it reaches a memory usage of 600 MB. If you have 8 GB, on the other hand, the process would *theoretically* restart when it reaches 4,8 GB, but since it is a 32 bit process it will never grow that big

requestLimit
This setting is "infinite" by default, but if it is set to 5000 for example, then ASP.NET will launch a new worker process once it's served 5000 requests

timeout
The default timeout is "infinite", but here you can set the lifetime of the worker process. Once the timeout is reached ASP.NET will launch a new worker process, so setting this to "00:05:00" would recycle the application every five minutes.

Other properties
There are other properties within the processModel element that will cause your application pool to recycle, like responseDeadlockInterval. But these other settings usually depend on something going wrong or being out of the ordinary to trigger. If you have a deadlock then that's your main concern. Changing the responseDeadlockIntervalsetting wouldn't do much to resolve the situation. You'd need to deal with the deadlock itself.

Editing and updating
ASP.NET 2.0 depends on File Change Notifications (FCN) to see if the application has been updated. Depending on the change the application pool will recycle. If you or your application is adding and removing directories to the application folder, then you will be restarting your application pool every time, so be careful with those temporary files

Altering the following files will also trigger an immediate restart of the application pool:

- web.config
- machine.config
- global.asax
- Anything in the bin directory or it's sub-directories

Updating the .aspx files, etc.causing a recompile will eventually trigger a restart of the application pool as well. There is a property of the compilation element under system.web that is called numRecompilesBeforeAppRestart. The default value is 20. This means that after 20 recompiles the application pool will recycle

A workaround to the sub-directory issue

If your application really depends on adding and removing sub-directories you can use linkd to create a directory junction. Here's how:

- Create a directory you'd like to exclude from the FCN, E.g. c:\inetpub\wwwroot\WebApp\MyDir
- Create a separate folder somewhere outside the wwwroot. E.g. c:\MyExcludedDir
- Use linkd to link the two: linkd c:\inetpub\wwwroot\WebApp\MyDir c:\MyExcludedDir

- Any changes made in the c:\inetpub\wwwroot\WebApp\MyDir will actually occur in c:\MyExcludedDir so they will go unnoticed by theFCN

Is recycling the application pool really that bad?
You really shouldn't have to recycle the application pool, but if you're dealing with a memory leak in your application and need to buy time to fix it, then by all means recycling the application pool could be a good idea