The Microsoft Windows Dynamic Cache Service will manage the working set size of the Windows System File Cache. For 64 bit systems this service helps to address the problem of excessive cached read I/O that could eventually consume all of physical memory. This problem is described in this blog post:

<http://blogs.msdn.com/ntdebugging/archive/2007/11/27/too-much-cache.aspx>

This service will only run on Windows Server 2008 R2 or earlier versions of Windows. Do not attempt to run this service on a version of Windows after Windows Server 2008 R2 as it will most likely cause performance problems.

The included source code and compiled executables demonstrate one example of using Microsoft Windows APIs to affect the Windows System File Cache behavior.

**Installation**

1) Copy DynCache.exe to %SystemRoot%\System32.

2) From a command prompt, run:

sc create DynCache binpath= %SystemRoot%\System32\DynCache.exe start= auto type= own DisplayName= "Dynamic Cache Service"

3) Import the DynCache.reg registry file. This registry file contains default settings that you will probably want to modify.

If you wish to uninstall this service, execute the following commands:   
sc stop DynCache  
sc delete DynCache

**Configuration**

The Dynamic Cache Service will monitor the registry for configuration changes. You do not need to restart the service if you change any of these settings.

Open Regedit.exe and go to this registry location:

HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Services\DynCache\Parameters

RegValue: **MaxSystemCacheMBytes**Type: REG\_DWORD  
Values:  
0 = Limit to 90% of Physical RAM (default)  
1-99 = Limit the maximum size of the System File Cache to this percentage of Physical RAM  
> 200 = Limit the maximum size of the System File Cache to x Mbytes

How this works:  
This setting is the absolute maximum that the System File Cache’s working set could be set to. The default is 0, limiting it to 90% of physical RAM with an upper limit of total Physical RAM minus 300 Mbytes. The lower limit for absolute values is 200 Mbytes and it must be at least 100 Mbytes greater than the **MinSystemCacheMBytes** value, which defaults to 100 Mbytes.

RegValue: **MinSystemCacheMBytes**Type: REG\_DWORD  
Values:  
0 = 100 Mbytes (default)  
x = Limit the minimum size of the System File Cache to x Mbytes

How this works:  
This setting is the minimum size of the System File Cache’s working set. The absolute lower limit is 100 Mbytes.The service does not dynamically change this value.

RegValue: **BackOffMBytesOnLowMemory**  
Type: REG\_DWORD  
Values:  
0 = Do not back off on low memory notification  
x = Back off x MBytes for each notification. Keep backing off until the minimum limit is reached.

How this works:   
If this value is set, the Dynamic Cache Service will register for a low memory notification. If the available memory drops below a threshold, the Dynamic Cache Service will reduce the maximum working set size of the System File Cache by the amount defined here (in Mbytes). As long as available memory is below this threshold, this service will continue to reduce the maximum working set size by this amount. Once available memory is above the threshold, the service will restore the max working set size in increments of this amount over time. The default low memory threshold that triggers the notification is 64 MB (if you have at least 2GB of physical RAM). You can modify the low memory threshold by adding this registry value:

HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Control\Session Manager\Memory Management

**LowMemoryThreshold** (REG\_DWORD) – This value is in Mbytes

RegValue: **SampleIntervalSecs**Type: REG\_DWORD  
Values:  
0 = Do not monitor processes’ working sets. Just set the System File Cache limits once and exit.  
x = Check the processes’ working sets every x seconds. Adjust the working set of the System File Cache accordingly.

How this works:  
If this is value is defined, this service will use performance monitor counters to check the working sets of active processes. It will adjust the size of the System File Cache’s working set accordingly. Setting this value too low could result in more overhead in monitoring and constant adjusting of the System File Cache’s working set. Setting this value too high may not allow this service to adjust the working set size of the System File Cache quick enough. The example settings file DynCache.reg sets it to 60 seconds.

RegValue: **CacheUpdateThresholdMBytes**Type: REG\_DWORD  
Values:  
0 = Default to 100 Mbytes  
x = Update the System File Cache limits when the new calculations change by x Mbytes.

How this works:  
This value defines the minimum amount of working set size change that is required before this service will update the size of the System File Cache’s working set. This setting is used to reduce the frequency of adjusting the System File Cache’s working set that can be caused by small changes in process working sets. If you use the default setting of 100 Mbytes, then the process’ working sets will need to increase or decrease by at least 100 Mbytes before this service will update the System File Cache’s working set size.

RegKeys: **<Process Image Name>**  
Create registry subkeys under the Parameters registry key for each process that you want to back off from. This service will monitor the working sets of each of these processes and back off the System File Cache’s working set by their combined amount. If there are multiple instances of one process, the combined working sets for all instances will be used. In essence you are trying to prioritizing the working sets for these processes to be higher than the System File Cache’s working set. The registry file includes 3 examples: Notepad, Microsoft SQL Server and the System. Each process can use these additional settings.

RegValue: **<Process Image Name>\AdditionalBackoffMBytes**  
Type: REG\_DWORD  
Values:  
0 = No additional back off bytes for this process  
x = Back off an additional x Mbytes if this process is running

How this works:  
If this value is defined and this process is currently active this service will back off the process’s working set plus this additional amount. You can use this setting as additional slack space for only when this process is running. Multiple instances of this process have no affect on this amount. It is only used for the first instance of this process.

RegValue: **<Process Image Name>\AdditionalBackOffCounter**  
Type: REG\_SZ  
Value:  
<formatted counter string> = Whatever this counter value is, back off by that amount.

How this works:  
This string needs to be a correctly formatted performance monitor string. Refer to the SQL Server sample for an example of using an additional counter to monitor. If this process is active, this service will use this additional counter.

**SQL Server Example:**

SQL Server’s process’s working set does not include the size of the database that is in physical RAM. The SQL Server performance counter exposes a different counter that keeps track of database memory usage. That counter is: "\SQLServer:Memory Manager\Total Server Memory (KB)". If this counter is defined as an AdditionalBackOffCounter for sqlservr (as provided in the default registry file), then this service will back off from SQL Server’s working set and the working set of its database. If you want to prioritize SQL’s (or similar application) working set over the System File Cache’s working set then you should use this counter to properly account for the memory usage of this process.

Since this service is set to start automatically and it is very light weight, it will probably start before SQL Server. The service will attempt to query the SQL counter, but the SQL may not be fully started. The SQL counter may return invalid numbers. If you use this service to monitor SQL Server, then you need to set this service’s dependency on SQL Server’s service. This way the SQL counter will be able to query the SQL service and return valid numbers. To set a dependency on the SQL service, go to the following registry key and add this registry value:

HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Services\DynCache\Parameters

RegValue: DependOnService  
Type: REG\_MULTI\_SZ  
Value:  
MSSQLSERVER

**Source Code**

The source code for the Microsoft Windows Dynamic Cache Service is provided along with the compiled binaries. In order to build this project, it is recommended that you install both the Windows Driver Kit (WDK) and the Windows Software Development Kit (SDK). Use the WDK build environment and add the following environment variables:

PSDK\_INC\_PATH – Set this to the location of the include files for the Windows SDK.  
PSDK\_LIB\_PATH – Set this to the location of the libraries for the Windows SDK.

This project requires the Windows SDK because it uses the Performance Data Helper (PDH) library to collect performance counter data.

**Troubleshooting/Verification**

The debug build of this service includes debug logging of what this service is doing. Copy the debug build of this service to your System32 folder and use DebugView (from [www.sysinternals.com](http://www.sysinternals.com)) to see what this service is doing.

**Disclaimer**

This service is experimental. The code and compiled binary are provided as a sample. Anyone is free to use this code in their own solution. Please note that you may experience some performance side effects while using this service. It cannot possibly address all usage scenarios. There may be some edge usage scenarios that are negatively impacted by this service. The service only attempts to improve the situation given the current limitations. Please report any bugs or observations to the blog (above). While we may not be able to fix every usage problem, we will try to offer best effort support.

**Side Effects may include**

Cache page churn - If the System File Cache's working set is too low and there is sustained cached read I/O, the memory manager may not be able to properly age pages. When forced to remove some pages in order to make room for new cache pages, the memory manager may inadvertently remove the wrong pages. This could result in cached page churn and decreased disk performance for all applications.