Monitoring the K2 blackpearl Environment

USING PERFORMANCE MONITORING AND COUNTERS WITH K2 BLACKPEARL

September 18, 2008

Updated: May 4th, 2010

This document contains an overview to the recommended performance monitoring options for monitoring K2 blackpearl environments, including overall server monitoring, K2 specific counters for the K2 Server, SQL Server, Web Server and SharePoint Server counters.

CORPORATE HEADQUARTERS
2615 151st Place NE
Redmond, WA 98052
USA
PH +1 (425) 883 4200
FAX +1 (425) 671 0411

EMEA HEADQUARTERS
26 Worple Road
Wimbledon
London
UK
PH +1 44 (0) 845 612 0912
FAX +1 44 (0) 845 612 0911

APAC HEADQUARTERS
9 Shenton Way #06-02
Singapore
068813
PH +1 65 6327 4110
FAX +1 65 6327 4120

[WWW.K2.COM]

CONTENTS

INTRODUCTION

Performance Monitoring is a complex subject and in some ways more of an art than a science. With over a thousand performance counters to select from, choosing the right counters to monitor can be a significant task. The choice can be impacted by the role of the system being monitored and whether the focus is capacity planning, ensuring availability, scaling upwards, monitoring for possible problems, or troubleshooting issues that have arisen. Additionally, application performance monitoring aids in preventing performance issues before they ever arise, driving further optimization of the applications to provide an even better user experience, and determining hardware requirements to sustain the application. On-going and routine monitoring of system resources enables early detection of possible problems in the form of constraints and over or under usage of resources.

This document contains an overview of the recommended performance monitoring options for monitoring K2 blackpearl environments as well as some additional useful counters and their descriptions. Please note that all of the best practices included in this document were sourced from freely accessible support, blog and technical sites, as well as best practices. As there is a very wide scope in system counters to choose for variants of systems and applications, this document is aimed at pointing out the most useful counters for monitoring a typical K2 blackpearl installation.

The foundation for effectively monitoring performance and keeping the system running is having in-depth knowledge of the K2 blackpearl components. In-depth knowledge can be gained by attending training courses and reading all accompanying documentation, including whitepapers and product documentation. In addition to product knowledge, a solid understanding and capability to set up and use the following is advantageous:

- Windows security
- Networking
- Microsoft Office SharePoint Server (or WSS 3.0)
- InfoPath and XML
- Visual Studio and basic coding skills
- SQL Server
- IIS Server

Note: For the most up-to-date information regarding the technologies and software supported by K2, please see the Compatibility Matrix available at http://help.k2.com/en/blackpearlmatrix.aspx.

Note: Performance Counters need to be enabled in the K2server.setup file stored at: C:\Program Files (x86)\K2 blackpearl\Host Server\Bin. See http://www.k2underground.com/blogs/johnny/archive/2010/06/14/k2-4-5-tip-of-the-day-perfmon-counters-in-4-5.aspx for more information.

The following line needs to be edited in K2Server.setup:

```
<PerfMonCounters Enable="False" Interval="15000" />
```

- **Enable** can be either “True” or “False”
- **Interval** is measured in milliseconds and indicates the K2 performance counter data is to

be updated every 15 seconds.

For the performance counters to work, set Enable="True"

Once the file is changed the K2 Host Server Service must be restarted before trying to add counters.

PERFORMANCE MONITORING COUNTERS

Using a tool such as PerfMon, you can set up your performance monitoring to check for certain counters. The counters will vary based on the role the server plays in the K2 environment. A typical K2 blackpearl installation involves multiple servers. The target servers for monitoring in the K2 blackpearl context include:

- K2 Server
- SQL Server (including Reporting Services)
- Web Server
- SharePoint Server

Each of these servers offer a set of performance counters to chose from, and it is essential to select the appropriate counters to monitor.

It is also important to note that prolonged monitoring can impact the performance of a system. Be sure to monitor to set baselines and when troubleshooting, but do not run performance monitoring continuously on your production system.

K2 SERVER MONITORING

The K2 Server has several counters that can be used to monitor performance. The full list of counters is included in Appendix A. The key counters that should be used to monitor performance are highlighted in the below table:

Table 1: K2 Server Monitoring Counters

Counter	Description
K2 Processes Started	The number of K2 Process Instances started by the K2 Server across all process definitions, based on when the K2 Host Server service was last started. This is a running count based on up time which can help describe how performant your K2 server is. Combined with the other user counters, such as K2 Worklists Opens and K2 Worklist Items Finished, you can monitor how much load on the K2 Server is created by the end users.
K2 Worklist Items Finished	The number of K2 Worklist items that was finished by a user, also based on uptime. This is a running count of how many times an end user actioned a Worklist item.
K2 Worklists Opens	The number of worklists that have been opened by the K2 Server, either through the API, K2 Worklist Web Part, or on the Workspace. Large numbers of worklist items on a user's worklist can impact performance as it is a memory intensive process to open large worklists.

K2 Processes Total Active	The number of K2 Processes that are currently in an active state, not based on uptime. This describes the load of the machine based on active process instances, not just the end user impact.
TCP Bytes Received (or Sent) Per Second	The number of Transmission Control Protocol (TCP) Bytes received by the K2 Server per second, can indicate bandwidth bottlenecks. In a distributed environment, the network connectivity between K2 components can impact performance of the environment. Monitoring this counter can look at the network performance and indicate issues.
Process Memory Usage	The memory usage (in bytes) for the K2 Server Windows process. This counter is not related to process instances or process definitions, but for the K2 Host Server.
Process Modules Loaded	The number of modules loaded by the K2 Server, this number correlates to the number of processes deployed, including process versions with instances that are active.

These counters can be used to look at the K2 Server performance as well as to validate the investment in the K2 platform. Because the counters look across all business processes, you can see how the platform is being utilized as well as make informed decisions on when to scale out the environment, by adding K2 Server nodes, increasing memory, or increasing bandwidth.

The key performance counters are described in more detail below. The sections are named by the performance counter, followed by the type of object. This type is used to categorize counters in PerfMon.

K2 Processes Started (K2 Server)

This counter is a running count of how many process instances have been started since the K2 Server service was started. This counter is based on uptime, and counts across all process definitions deployed to the K2 Server. You can monitor this counter over a period of time to see how many processes were started and when. This can help identify peak usage times (such as submitting Expense Reports on the last day of the business cycle) which you can help mitigate through cultural changes. You can also identify when an additional node should be added to the K2 Server cluster based on how many instances your hardware is handling and when performance starts to degrade.

K2 Worklist Items Finished (K2 Server)

This counter is a running count of the number of K2 Worklist items that was finished by a user, also based on uptime. This counter is an indication of the end user impact on the system. If performance when actioning a worklist item is slow, monitor how many items are finished over a period of time and look for peaks. This counter can also be used when deciding when to scale out the K2 Server based on the requests.

K2 Worklists Opens (K2 Server)

This counter indicates the number of times a worklist has been opened by the K2 Server, either by the K2 Worklist Web Part, the Workspace, or via the API. Large numbers of worklist items on a user's worklist can impact performance as it is a memory intensive process to open large worklists. If worklists open slowly, look at the number of worklists opens as well as how much data is being retrieved based on the process. Limiting the data fields retrieved to only those critical to be viewed on the worklists will decrease the amount of data retrieved by

the K2 Server when a worklist is opened and can help speed up performance. Caching or having manual refreshes on the worklists rather than an automated refresh interval can also help with performance.

K2 Processes Total Active (K2 Server)

This counter is the number of K2 Process Instances that are currently in an active state. This describes the load of the machine based on active process instances, not just the end user impact. Long running processes will impact this counter, as this is a look at the number of instances currently active, not a running count. Carrying a high load of long running processes will impact the K2 Server performance.

TCP Bytes Received (or Sent) Per Second (K2 Server)

This counter is the number of Transmission Control Protocol (TCP) Bytes received (or Sent) by the K2 Server per second. This counter can indicate bandwidth bottlenecks. In a distributed environment, the network connectivity between K2 components can impact performance of the environment. Monitoring this counter can look at the network performance and indicate issues. This counter can also be an indication of how large worklists are if you monitor TCP Bytes Received/Sent and K2 Worklists Opens and look at the ratios. Decreasing the number of data fields sent with the worklist can decrease the network traffic and thereby reduce the network traffic bottleneck.

Process Memory Usage (K2 Server)

This counter monitors the memory usage (in bytes) for the K2 Server Windows process. This counter is not related to process instances or process definitions, but for the K2 Host Server. Monitoring the memory usage can indicate if additional memory is needed on the physical server.

Process Modules Loaded (K2 Server)

The number of process modules loaded by the K2 Server correlates to the number of processes deployed. This counter also includes process versions with instances that are active. Long running processes based on older versions will cause additional process modules to be loaded by the K2 Server, thereby increasing the memory used by the K2 Server. Cleaning up process instances running on old versions will help decrease the number of modules loaded into memory, and free up resources for the current version and increase performance.

SQL SERVER MONITORING

The following are the recommended counters for monitoring the SQL server. A rationale for each of these counters is provided in the below table:

Table 2: SQL Server Monitoring Counters

Counter	Description
% Processor Time	The % Processor Time is the standard SQL Server counter and provides a good gauge for looking at a server's usefulness. % Processor Time is defined as the amount of time that the server is executing a non-idle thread, and is calculated by looking at the percentage of time that the idle thread executes and subtracting this from 100.
Processor Queue Length	Processor Queue Length represents how many threads are waiting for processor time. The count includes all threads that are "ready" and not waiting on some other thing like IO to

	complete.
Pages/Sec	This counter looks at the number of memory pages read or written to disk, and the number should be the sum of both these.
Available Mbytes	Available Mbytes should be the amount of MB of memory on the computer (from the OS perspective) that is available to be allocated to processes. It is calculated as the sum of Free, Standby, and Zeroed memory pages.
Avg. Disk Queue Length	This counter measures whether I/O requests are being held by the disk drive while they catch up on requests.
% Idle Time	Measures the percent time that your hard disk is idle during the measurement interval. If you see this counter fall below 20% then you've likely got read/write requests queuing up for your disk which is unable to service these requests in a timely fashion.
Bytes Total/sec	This counter provides the number of bytes being transmitted per second on the NIC. The counter will display one instance per NIC on the machine, including one for the loopback adapter.
Full Scans/Sec	This counter shows how often indexes are not being used.
Transactions/Sec	This number indicates how active your SQL Server system is. A higher value indicates more activity is occurring.
Cache Hit Ratio	This counter shows the percentage of pages that are found in the buffer as opposed to disk. Read ahead threads are not included in this counter.
User Connections	The User Connections counter measures how many connections there are to SQL Server on a spot basis. This counter is not an average; rather it is just the current number of connections.
Average Wait Time	The Average Wait Time counter measures the amount of time in milliseconds that a user is waiting for a lock and is given as an average that is compiled over the performance monitor interval.

The key performance counters are described in more detail below. The sections are named by the performance counter, followed by the type of object. This type is used to categorize counters in PerfMon.

% Processor Time (Process Object)

This counter provides an indication of a first bottleneck. If the CPU is pegged, then the server is probably fully tasked, and end users will notice performance degradation. It is recommended to keep this number under 40% unless a large query is being processed. It is not recommended to set an alert or have the monitoring software send a notification immediately as this will result in being constantly notified. Servers will go to 100% CPU at

times; however, unless this is sustained for a period of time, such as more than 15 minutes, it should not be a concern. The threshold could be lowered for some systems if necessary.

Over time, such as a month, if the CPUs are averaging more than 50%, the planning process for an upgrade should be considered. If CPUs are greater than 70% on average, an upgrade is strongly recommended.

Processor Queue Length (System Object)

This counter should be monitored over a sustained period to determine if the CPU is a bottleneck. As a general rule of thumb, this value should be less than 2x the number of CPUs in the system. If the number is growing and the CPU % (see above) is above 80%, more or faster processors are indicated.

Pages/Sec (Memory Object)

The SQL 2000 Operations Guide defines Pages/Sec as "The number of pages read from or written to disk to resolve hard page faults. (Hard page faults occur when a process requires code or data that is not in its working set or elsewhere in physical memory, and must be retrieved from disk). This counter was designed as a primary indicator of the kinds of faults that cause system-wide delays. It is the sum of Memory: Pages Input/sec and Memory: Pages Output/sec. It is counted in numbers of pages, so it can be compared to other counts of pages, such as Memory: Page Faults/sec, without conversion. It includes pages retrieved to satisfy faults in the file system cache (usually requested by applications) non-cached mapped memory files. This counter displays the difference between the values observed in the last two samples, divided by the duration of the sample interval."

This counter is a relative counter and cannot be measured independent of a system. It is recommended to establish a baseline for this counter and then monitor the counter over time and as changes are made to the system. In general, adding memory (or allocating more to SQL) should lower this counter; however, if too much memory is allocated to SQL and the Operating System (OS) is starved for memory, this counter could go up.

Available Mbytes (Memory Object)

This counter should be used to monitor that the average amount of memory (in MB) is fairly consistent from a baseline perspective. A drop in this counter may indicate a change, such as processes or services being added to the server.

Avg. Disk Queue Length (System Object)

Average Disk Queue Length can indicate a bottleneck in performance for a server in that if it grows large or is consistently above an 8 for a particular disk, then that disk is spending a good amount of time stacking requests up instead of servicing them.

% Idle Time (Physical Disk Object)

Monitoring the % Idle Time counter eliminates problems with the % Disk Time monitor due to how the data for the counters are collected. Subtracting the % Idle Time from 100 provides an idea of how hard the disks are working.

When using this counter it is important that the counter is included for each disk instance and not the whole group.

Bytes Total/Sec (Network Interface Object)

The SQL 2000 Operations Guide defines Bytes Total/Sec as "the number of bytes travelling over the network interface per second." If this rate begins to drop, you should investigate whether or not network problems are interfering with your application.

This counter should be roughly 60% of the theoretical max for the NIC. If the value is lower, begin investigating the network to ensure the right type of card is plugged into an appropriate port. Overall, this counter basically provides a double check, and is good to have as part of the baseline for monitoring.

Full Scans/Sec (SQL Server Access Methods Object)

The SQL 2000 Operations Guide defines Full Scans/Sec as "the number of unrestricted full scans." These can either be base table or full index scans.

This counter is another relative counter and requires a baseline measurement for monitoring. Once a baseline is established providing an average number, tuning efforts for raising or lowering this number can be monitored.

Transactions/Sec (SQL Server Databases Methods Object)

The Transactions/Sec counter helps to determine the average utilization of the server since transactions are the basis of everything in SQL Server; however, this counter only shows transactions that change data and can therefore show the long term expected number of transactions. This counter is extremely helpful in determining if the load has substantially increased. For example, having a long term average of approximately 12 or so transactions/sec and seeing a spot rate of 200/sec shows that perhaps the application is ok but that the hardware may not be sufficient.

Cache Hit Ratio (SQL Server Buffer Manager Object)

The SQL 2000 Operations Guide defines Cache Hit Ratio as "the percentage of pages that were found in the buffer pool without having to incur a read from disk." When this percentage is high your server is operating at optimal efficiency (as far as disk I/O is concerned). Generally this counter should stay above 95%. If the counter drops below 95% it should be investigated.

User Connections (SQL Server General Statistics Object)

The number of connections depends on how the servers are connected. Tracking this counter on a baseline basis helps to correlate the performance of the system with an average number of users. If the users increase, check other counters such as CPU, memory, and lock wait time to see if there is a corresponding change with the larger user load. As with many counters, this counter should be used as a broad instrument to measure the performance of the system, not a finely honed one.

Average Wait Time (SQL Server Locks Object)

The SQL 2000 Operations Guide defines Average Wait Time as "the average amount of wait time (milliseconds) for each lock request that resulted in a wait."

This counter should be monitored for deviations from the average or baseline over time. While troubleshooting performance issues the spot values should be compared to the baseline. On a broad, system level basis, this counter indicates if the system in general is experiencing problems or just one user.

OVERALL SERVER MONITORING

Regardless of the role of the server in the K2 environment, there are some common performance counters that can be monitored to ensure the server is healthy. These counters are described in the below table. These counters should be monitored on all servers in addition to the role-specific counters described in later sections.

The following list of counters includes some of the key counters that can be used to monitor the general health of a server. In addition, the counters that are listed below the key counters are useful for investigating or troubleshooting server issues.

Table 3: Server Monitoring Counters

Counter	Description
System\System Up Time	The System\System Up Time counter tells how many seconds it has been since the server last rebooted.
Process(instance)\Elapsed Time	Using the Process(instance)\Elapsed Time counter provides information on how long a particular process has been running on a server.
Processor(_Total)\%ProcessorTime	Processor(_Total)\% Processor Time measures the total utilization of the processor by all running processes to provide data on how busy a server is. Note that on multiprocessor machines, Processor(_Total)\% Processor Time actually measures the average processor utilization of the machine (i.e., utilization averaged over all processors). Related counters: > Process(instance)\%Processor Time > Processor(_Total)\% Privileged Time > Processor(_Total)\% User Time
System\Processor Queue Length	The System\Processor Queue Length counter gives an indication of how many threads are waiting for execution.
System\Context Switches/sec	System\Context Switches/sec measures how frequently the processor has to switch from user- to kernel-mode to handle a request from a thread running in user mode, which helps to monitor how the machine's hardware devices are functioning. Related counters: > Processor(_Total)\Interrupts/sec
Memory\Pages/sec	The Memory\Pages/sec counter indicates the number of paging operations to disk during the measuring interval.
Memory\Available Bytes	Indicates how much physical memory is remaining after the working sets of running processes and the cache have been served. Related counters: > Process(instance)\Working Set

	> Memory\Cache Bytes
Memory\Transition Faults/sec	Memory\Transition Faults/sec measures how often recently trimmed pages on the standby list are re-referenced and can be an indicator of insufficient RAM.
Physical Disk (instance)\Disk Transfers/sec	The average number of read and write operations that have occurred on a disk per second. Related counters: > Physical Disk(instance)\% Idle Time

Each of these counters and how they are effective in monitoring server performance are described below.

System\System Up Time

Tracking this counter provides data around server availability. The easiest way to view this counter is in report view, which shows the actual numerical value of the counter in elapsed seconds. A better way to track this counter is through the use of a performance counter log where this counter can be tracked in the background and reviewed periodically, such as at month end.

Process(instance)\Elapsed Time

Using the **Process(instance)\Elapsed Time** counter provides information on how long a particular process has been running on a server. For example, **Process(winlogon)\Elapsed Time** tells how long it has been since the Winlogon process started running on the machine, and should normally be a few seconds less than **System\System Up Time** since Winlogon starts running during the boot process. The **Process(instance)\Elapsed Time** counter can also be used to monitor processes associated with specific applications and services to monitor the availability of these applications and services. Exercise caution when using it in this scenario as some services are designed to start and stop under certain conditions, while other services are embedded in service host processes (svchost.exe).

Processor(_Total)\%Processor Time

If the Processor(_Total)\%Processor Time counter is running at or near 100% for extended periods of time, use the **Process(instance)\%Processor Time** counter for various process instances on the machine to drill down to the process level. For example, on an IIS web server **Process(inetinfo)\% Processor Time** might be tracked, while on an Exchange server a good counter to watch is **Process(store)\% Processor Time**. High processor utilization isn't always a sign of a problem. Typically a server running at around 70% or 80% processor utilization is normally a good sign and means the machine is handling its load effectively and is not under-utilized. Average processor utilization of around 20% or 30% on the other hand suggests the machine is under-utilized and may be a good candidate for server consolidation using virtualization software.

Another counter that may be useful for investigating high processor utilization is to drill down into **Processor(_Total)\% Privileged Time** and **Processor(_Total)\% User Time**, which respectively show processor utilization for kernel- and user-mode processes on the machine. If kernel mode utilization is high, the machine is likely underpowered as it's too busy handling basic OS housekeeping functions to be able to effectively run other

applications. And if user mode utilization is high, it may be that the server is running too many specific roles and the hardware should be improved by adding another processor or migrating an application or role to another box.

System\Processor Queue Length

If this counter is consistently higher than around 5 when processor utilization approaches 100%, then it is a good indication that there is more work (active threads) available (ready for execution) than the machine's processors are able to handle. Note that this is not always a hard and fast indicator as some services, like IIS 6, pool and manage their own worker threads. In this situation, look at other counters like **ASP\Requests Queued** or **ASP.NET\Requests Queued** to determine server utilization. Additionally, the larger the number of active services and applications running on the server, the busier the processor queue will normally be, so on a multi-role server running near 100% utilization content may only be a significant factor once **System\Processor Queue Length** exceeds something like 10 instead of 5 as previously mentioned.

System\Context Switches/sec

The heavier the workload running on the machine, the higher this counter will generally be, but over long term the value of this counter should remain fairly constant. However, if this counter suddenly starts increasing, it may indicate a malfunctioning device, especially if a similar jump is seen in the **Processor(_Total)\Interrupts/sec** counter. Checking the **Processor(_Total)\% Privileged Time** counter for a similar unexplained increase, may indicate problems with a device driver that is causing an additional hit on kernel mode processor utilization. Use the **Process(instance)\% Processor Time** counter to drill down and examine each process instance running on the machine to determine which calling application is indirectly causing the problem and to help troubleshoot the issue further. If **Processor(_Total)\Interrupts/sec** does not correlate well with **System\Context Switches/sec**, the sudden jump in context switches may instead mean that the application is hitting its scalability limit on the particular machine and the application may need to be scaled out (for example by clustering) or possibly redesigning how the application handles user mode requests. Ideally **System\Context Switches/sec** should be monitored over a period of time to establish a baseline and then create a PerfMon alert that will trigger when this counter deviates significantly from its observed mean value.

Memory\Pages/sec

The **Memory\Pages/sec** counter is the primary counter to watch for indication of possible insufficient RAM to meet the server's needs. A good idea here is to configure a PerfMon alert that triggers when the number of pages per second exceeds 50 per paging disk on the system.

Memory\Available Bytes

If the **Memory\Available Bytes** counter is greater than 10% of the actual RAM in your machine then you probably have more than enough RAM and don't need to worry. When monitoring the **Memory\Available Bytes** counter, create a performance log and monitor it regularly to see if any downward trend develops, and set an alert to trigger if it drops below 2% of the installed RAM. If a downward trend does develop, monitor **Process(instance)\Working Set** for each process instance to determine which process is consuming larger and larger amounts of RAM. **Process(instance)\Working Set** measures the size of the working set for each process, which indicates the number of allocated pages the process can address without generating a page fault. A related counter is **Memory\Cache Bytes,** which measures the working set for the system, i.e., the number of allocated pages kernel threads can address without generating a page fault.

Memory\Transition Faults/sec

If this counter slowly starts to rise over time then it could also indicate reaching a point where the server no longer has enough RAM to function well.

Physical Disk (instance)\Disk Transfers/sec

Monitor the **Physical Disk (instance)\Disk Transfers/sec** counter for each physical disk to identify any bottlenecks from the disk. If this counter goes above 25 disk I/Os per second then the disk has poor response time. Investigate poor response time further by tracking **Physical Disk(instance)\% Idle Time**, which measures the percent time that the hard disk is idle during the measurement interval. If this counter falls below 20% then it is likely read/write requests are queuing up for the disk which is unable to service these requests in a timely fashion. In this case it's time to upgrade the hardware to use faster disks or scale out the application to better handle the load.

WEB SERVER MONITORING

Web Server monitoring focuses on the performance of the application using Internet Information Services (IIS) and ASP.NET platforms.

When monitoring the Web server, the following areas should be focused on:

- Web Service Performance
- Web Service Cache Performance
- Internet Information Services Global Performance
- Active Server Pages Performance
- ASP.NET and ASP.NET Applications Performance

The key counters for each of these areas are described below. The counters described in this section are specific to IIS 6.

Web Service Performance

Web Service counters provide information on how well the World Wide Web Publishing Service (WWW Service) is processing requests. The WWW Service is a user-mode service. The counters used for monitoring IIS also reflect the processing that occurs in the kernel-mode driver, HTTP.sys.

Counters for monitoring web service performance include the following factors:

- Bytes Sent, Received and Transferred
- Files Sent, Received and Transferred
- Anonymous and Non-Anonymous Users
- Connections and Attempts
- Requests
- Errors
- CGI Requests and ISAPI Extension Requests

The counters can be configured to monitor performance for individual Web sites or for all sites on a server. To configure for all sites on a server, use the _Total instance.

Key counters for monitoring web service performance are provided below. The full list of counters can be found in Appendix C.

Table 4: Web Service Performance Monitoring Counters

Files (Sent, Received, and Transferred) Counters	
Counter	**Description**
Files/sec	The rate, in seconds, at which files have been sent and received by the WWW service.
Anonymous Users and NonAnonymous Users Counters	
Counter	**Description**
Anonymous Users/sec	The rate, in seconds, at which users have made anonymous requests to the WWW service. This is a good counter when monitoring for security and/or attacks.
Connections and Attempts Counters	
Counter	**Description**
Current Connections	The number of active connections to the WWW service.
Maximum Connections	The maximum number of simultaneous connections made to the WWW service since the service started.
Total Logon Attempts	The number of attempts to log on to the WWW service that have occurred since the service started.
Errors (Not Found and Locked) Counters	
Counter	**Description**
Total Not Found Errors	The number of requests that have been made since the service started that were not satisfied by the server because the requested document was not found. Usually reported as HTTP error 404.

Web Service Cache Performance

The WWW service and FTP service do not share a common cache. Instead, the caches are split into two separate performance objects: one for FTP service and one for the WWW service. WWW service cache counters are designed to monitor server performance only; therefore, you cannot configure them to monitor individual sites. For information about FTP service counters, see Internet Information Services Global Counters below.

Counters for monitoring web service cache performance include the following factors:

- File

- URI
- Metadata
- Kernel (Kernel counters reflect all HTTP.sys activity, not just IIS activity.)

Key counters for monitoring web service cache performance are listed below. The full list of counters can be found in Appendix D.

Table 5: Web Service Cache Performance Monitoring Counters – File Counters

Counter	Description
Current File Cache Memory Usage	The number of bytes currently used for the user-mode file cache.

Internet Information Services Global Performance

The Internet Information Services Global counters described below should be used to monitor FTP, SMTP, and NNTP services as a whole. These counters cannot be configured to monitor individual sites. If the service that you want to monitor (FTP, SMTP, or NNTP) is not installed or is not running, the counters return a zero value.

Counters for monitoring Internet Information Services Global performance include the following factors:

- File Cache
- Flushed and URI Cached
- Binary Large Objects (BLOBs)

Key counters for monitoring IIS Global performance are listed below. The full list of counters can be found in Appendix E.

Table 6: IIS Performance Monitoring Counters – File Cache Counters

Counter	Description
Current Files Cached	The number of files whose content is currently in the cache.
Current File Cache Memory Usage	The current number of bytes used for the file cache.

Active Server Pages Performance

The Active Server Pages (ASP) counters help determine how well the server or site is responding to ASP requests. The ASP counters are designed to monitor server performance. Individual ASP applications cannot be monitored using these counters because ASP counters collect global data across the entire WWW service.

Counters for monitoring Active Server Pages performance include the following factors:

- ASP Debugging and Errors

- Requests
- Script Engines, Sessions, Templates and Transactions

Key counters for monitoring ASP performance are listed below. The full list of counters can be found in Appendix F.

Table 7: ASP Performance Monitoring Counters

ASP Debugging and Errors Counters	
Counter	**Description**
Errors During Script Runtime	The number of requests that failed because run-time errors occurred.
Errors/sec	The average number of errors that occurred per second.
Requests Counters for ASP Pages	
Counter	**Description**
Requests Disconnected	The number of requests that were disconnected because communication failed.
Requests Executing	The number of requests that are currently executing.
Requests Failed Total	The number of requests that failed due to errors, authorization failure, and rejections.
Requests Not Authorized	The number of requests that failed because access rights were insufficient.
Requests Not Found	The number of requests that were made for files that were not found.
Requests Queued	The number of requests that are waiting in the queue for service.
Requests Rejected	The number of requests that were not executed because there were insufficient resources to process them.
Requests Succeeded	The number of requests that executed successfully.
Requests Timed Out	The number of requests that timed out.
Requests/sec	The average number of requests that were executed per second.
Script Engines, Sessions, Templates, and Transactions Counters	
Counter	**Description**
Session Duration	The length of time that the most recent session lasted, in milliseconds.

Sessions Current	The number of sessions currently being serviced.
Sessions Timed Out	The number of sessions that have timed out.
Sessions Total	The number of sessions that have run since the service was started.
Transactions Aborted	The number of transactions that have been aborted.
Transactions Pending	The number of transactions that are in progress.

ASP.NET and ASP.NET Application Performance

As K2 blackpearl is an ASP.NET application, monitoring ASP.NET performance is vital. ASP.NET includes a number of performance counters that are useful in tracking the execution of Web applications. Two performance objects are defined for ASP.NET, supporting two types of performance counters:

- The ASP.NET performance object contains system counters that monitor events on the state server. These global counters are not bound to a particular application instance.
- The ASP.NET Applications performance object contains application counters.

For example, a significant difference exists between the State Server Sessions counters for the ASP.NET performance object, which apply only to the computer on which the state server is running, and the Sessions counters for the ASP.NET Applications performance object, which apply only to user sessions that occur in process. Most of the ASP.NET performance counters are exposed per application.

Use System Monitor to view the counters for ASP.NET. If the Web server serves multiple applications, a particular application instance must be specified when selecting a counter to monitor. In addition, System Monitor provides a **__Total__** application instance, which aggregates the counter values for all applications on a server. During a steady-state running analysis, it is recommended that the first request and also any one-time initialization costs for objects are ignored.

Below are recommended counters for monitoring and troubleshooting ASP.NET and ASP.NET Applications. The full list of counters available for monitoring are included in Appendix G.

Note: The value of each ASP.NET performance counter is updated every 400milliseconds.

Table 8: ASP.NET Performance Monitoring Counters

Object\Counter	Description
ASP.NET\Requests Queued	The number of ASP.NET requests waiting to be processed.

ASP.NET\Worker Process Restarts	The number of times that a worker process has restarted on the computer.
ASP.NET Applications\Errors Total	The total number of errors that have occurred in ASP.NET applications.

Performance Counters for Troubleshooting ASP.NET Applications*	
Object\Counter	**Description**
.NET CLR Exceptions\# of Exceps Thrown	The total number of exceptions thrown since the start of the application. Collects data both on .NET exceptions and on unmanaged exceptions that are converted to .NET exceptions. For example, a null pointer reference exception in unmanaged code is rethrown in managed code as a .NET SystemNullReferenceException. Exceptions that are rethrown are recounted. This counter collects data from both handled and unhandled exceptions.
System\Context Switches/sec	The rate at which thread contexts are switched by all CPUs in the Web server. A high number usually indicates either high contention for locks or many switches between user and kernel mode by the thread.

***Note: Some code paths rely on exceptions for their proper functioning. For example, the HttpResponse.Redirect method always throws an exception, ThreadAbortException. Therefore, it can be more useful to track the number of exceptions thrown by using the Errors Total counter to see if the exception generated an error for the application.**

SHAREPOINT SERVER MONITORING

Microsoft Office SharePoint Server (MOSS) monitoring has front end and back end performance counters. The front end counters look at performance of the front end web servers that end users interact with. The back end counters look at SQL server performance in relation to SharePoint.

In the counters below, replace the server name $fe1$ with your front end server name, or $be1$ with your back end server name.

Table 9: SharePoint Performance Monitoring Counters

Front End Performance Counters*	
Counter	**Description**
\\$fe1$\Processor(_Total)\% Processor Time	The % Processor Time counter measures the percentage of elapsed time that the processor spends to execute a non-Idle thread.

\\$fe1$\Process(LSASS)\% Processor Time	Local Security Authority Subsystem Service (LSASS) is a process in Microsoft Windows operating systems that is responsible for enforcing the security policy on the system. This counter shows the percentage of elapsed time that this service used the processor to execute instructions.
\\$fe1$\Process(w3wp)\% Processor Time	Shows the percentage of elapsed time that the web service (IIS) used the processor to execute instructions.
\\$fe1$\Process(OWSTIMER)\% Processor Time	Shows the percentage of elapsed time that the SharePoint timer service used the processor to execute instructions.
\\$fe1$\Memory\Pages/sec	Measures the rate at which pages are read from or written to disk to resolve hard page faults.
\\$fe1$\Memory\Available MBytes	Available MBytes counter to measure the amount of physical memory in MB immediately available for allocation to a process or for system use.
\\$fe1$\System\Context Switches/sec	Indicates that the kernel has switched the thread it is running on a processor. A context switch occurs each time a new thread runs, and each time one thread takes over from another. A large number of threads are likely to increase the number of context switches. Context switches allow multiple threads to share time slices on the processors, but they also interrupt the processor and might reduce overall system performance, especially on multiprocessor computers. You should also observe the patterns of context switches over time.
\\$fe1$\Process(w3wp)\Working Set	The Working Set is the set of memory pages recently touched by the threads in the process.
\\$fe1$\Process(w3wp)\Private Bytes	Measures the current size, in bytes, of memory that this process has allocated and that cannot be shared with other processes.
\\$fe1$\Process(w3wp)\Page Faults/sec	Use the counter Page Faults/sec to measure the average number of pages faulted per second.
\\$fe1$\Process(w3wp)\Working Set Peak	The maximum size, in bytes, in the working set of this process at any point in time. If free memory in the computer is above a certain threshold, pages are left in the working set of a process even if they are not in use. When free memory falls below a certain threshold, pages are trimmed from working sets. If they are needed, they are then soft-faulted back into

	the working set before they leave main memory.
\\$fe1$\Process(w3wp)\Virtual Bytes	The current size, in bytes, of the virtual address space the process is using. Use of virtual address space does not necessarily imply corresponding use of either disk or main memory pages. Virtual space is finite, and by using too much, the process can limit its ability to load libraries.
\\$fe1$\Process(w3wp)\Virtual Bytes Peak	The maximum size, in bytes, of virtual address space the process has used at any one time. Use of virtual address space does not necessarily imply corresponding use of either disk or main memory pages. However, virtual space is finite, and the process might limit its ability to load libraries by using too much.
\\$fe1$\Process(w3wp)\Private Bytes	The Private Bytes counter reports memory allocated exclusively to the process.
\\$fe1$\Process(w3wp)\Page File Bytes	A small paging file limits what can be stored and might exhaust your virtual memory for applications. If you are short on RAM, more paging occurs, which generates extra activity for your disks and slows response times for the system.
\\$fe1$\Process(w3wp)\Page File Bytes Peak	The maximum size, in bytes, of the page files bytes.
\\$fe1$\ASP.NET\Request Execution Time	The number of milliseconds that the most recent ASP request took to execute. This value can be somewhat misleading because it is not an average.
\\$fe1$\ASP.NET\Request Wait Time	The number of milliseconds that the most recent ASP request was waiting in the queue.
\\$fe1$\ASP.NET\Requests Queued	The number of queued ASP requests that are waiting to be processed. The maximum number for this counter is determined by the metabase property AspRequestQueueMax.
\\$fe1$\ASP.NET\Requests Rejected	The value of this counter is the number of rejected requests. Requests are rejected when one of the queue limits is exceeded. Back-end latency, such as that caused by a slow computer running SQL Server, is often preceded by a sudden increase in the number of pipeline instances and a decrease in % Processor Time and Requests/second. A server might be overwhelmed during times of heavy load due to processor or memory constraints that ultimately result in the rejection of requests.

\\$fe1$\ASP.NET\Worker Process Restarts	Measures Worker Process Restarts
\\$fe1$\ASP.NET\Application Restarts	There are many causes to restart an application, the most common are: 1) Unhandled Exception. 2) Any write to application bin directory. 3) Any change to web.config. 4) Anti-virus program that touches files. 5) One of the ProcessModel attributes in the machine.config file that causes application recycling.
Back End Performance Counters	
Counter	**Description**
\\$be1$\\Processor(_Total)\% Processor Time	Shows the percentage of elapsed time that the total threads used the processor to execute instructions.
\\$be1$\Memory\Pages/sec	Measures the rate at which pages are read from or written to disk to resolve hard page faults.
\\$be1$\System\Context Switches/sec	Indicates that the kernel has switched the thread it is running on a processor. A context switch occurs each time a new thread runs, and each time one thread takes over from another. A large number of threads are likely to increase the number of context switches. Context switches allow multiple threads to share time slices on the processors, but they also interrupt the processor and might reduce overall system performance, especially on multiprocessor computers. You should also observe the patterns of context switches over time.
\\$be1$\Process(sqlservr)\% Processor Time	Shows the percentage of elapsed time that the SQL Server used the processor to execute instructions.
\\$be1$\Process(sqlservr)\Working Set	The set of memory pages (areas of memory allocated to a process) recently used by the threads in a process. If available memory on the server is above a specified threshold, pages remain in the Working Set of a process even if they are not in use. When available memory falls below a specified threshold, pages are removed from the Working Set. If these pages are needed, they will be returned back to the Working Set before they leave main memory and are made available for other processes to use.
\\$be1$\Process(sqlservr)\Private Bytes	Displays the current number of bytes this process has

	allocated that cannot be shared with other processes.
\\$be1$\SQLServer:General Statistics\User Connections	Displays the number of users currently connected to the server. Its maximum value is 255. An increase in the value of the counter causes performance problems and affects throughput.
\\$be1$\SQLServer:Databases\Transactions/sec	This number indicates how active your SQL Server system is. A higher value indicates more activity is occurring.
\\$be1$\SQLServer:Locks(_Total)\Number of Deadlocks/sec	Number of lock requests per second that resulted in a deadlock.
\\$be1$\SQLServer:Locks(_Total)\Lock Waits/sec	Number of lock requests per second that required the caller to wait.
\\$be1$\SQLServer:Locks(_Total)\Lock Wait Time (ms)	Total wait time (in milliseconds) for locks in the last second.
\\$be1$\SQLServer:SQL Statistics\Batch Requests/sec	Number of Transact-SQL command batches received per second. This statistic is affected by all constraints (such as I/O, number of users, cache size, complexity of requests, and so on). High batch requests mean good throughput.
\\$be1$\PhysicalDisk(_Total)\Current Disk Queue Length	Indicates the number of disk requests that are currently waiting as well as requests currently being serviced. Subject to wide variations unless the workload has achieved a steady state and you have collected a sufficient number of samples to establish a pattern. An instantaneous value or snapshot of the current queue length, unlike Avg. Disk Queue Length, Avg. Disk Read Queue Length, and Avg. Disk Write Queue Length, that reports averages.
\\$be1$\PhysicalDisk(_Total)\Disk Read Bytes/sec	Measures the rate of read operations from the disk
\\$be1$\PhysicalDisk(_Total)\Disk Write Bytes/sec	Measures the rate of write operations from the disk

***Note: Replace $fe1$ with your front end server name.**

****Note: Replace $be1$ with your back end server name.**

ADDITIONAL MONITORING

While not part of the K2 environment per se, there are additional items in your environment that can affect the performance of K2 blackpearl. These include:

- Active Directory Performance
- Network Performance

Active Directory (AD) performance will likely not be a major concern in a small network; however, as the number of domain controllers grows and more users and other objects are added to the directory, performance begins to become a more important consideration. The table below contains a list of the key counters that can be used in monitoring Active Directory and network performance. A full list of counters can be found in Appendix H.

Table 10: Active Directory and Network Performance Monitoring Counters

Network Transmission Counters	
Counter	**Description**
Bytes Total/sec	The rate at which bytes are sent and received on the interface. A higher number indicates better performance. Track the performance of each network interface to identify high utilization per interface and determine whether you need to use switches to segment the network or increase bandwidth.
% Network Utilization	This counter provides a good indication of the bandwidth utilization for the local segment and enables you to evaluate the impact of certain network events—such as replication—on network bandwidth. Consider 30 percent utilization a maximum for unswitched Ethernet. Adjust your acceptable benchmark based on your network topology.
Server and Domain Controller (NTDS Object) Counters	
Counter	**Description**
DRA Pending Replication Synchronizations	This counter indicates the replication backlog on the server. This value should be low, with a higher value indicating that the hardware is not adequately servicing replication.
DS Threads in Use	This counter shows the number of threads in use by Active Directory, with a lack of activity typically pointing to network problems that are preventing client requests from succeeding.
Kerberos Authentications/sec	This counter shows the number of Kerberos authentications on the server per second. A lack of activity can indicate network problems that are preventing authentication requests from succeeding.

LDAP Bind Time	This counter shows the time required for completion of the last LDAP binding, with a higher value pointing to either hardware or network performance problems.
LDAP Client Sessions	This counter shows the number of connected LDAP client sessions, with a lack of activity pointing to network problems.
NTLM Authentications	This counter shows the number of NTLM authentications per second handled by the domain controller (from Windows 98 and Windows NT clients). A lack of activity points to network problems.

Server and Domain Controller (Database Object) Counters

Counter	**Description**
Cache % Hit	This counter shows the percentage of database page requests handled by the cache, thereby not causing a file I/O. A lack of activity can indicate that the server has insufficient physical memory.
Cache Page Fault Stalls/sec	This counter shows the number of page faults per second that go unserviced due to lack of available pages in the database cache. A value other than zero indicates insufficient physical memory in the server.
Cache Page Faults/sec	This counter shows the number of page requests per second that cause the database cache to allocate new pages from the cache. This value should be low, with a higher value indicating insufficient physical memory in the server.
File Operations Pending	This counter shows the number of file operations for the database file(s) currently pending by the operating system. The value should be low, with a higher value indicating insufficient physical memory and/or inadequate CPU availability or performance.
File Operations/sec	This counter shows the number of file operations per second generated by the database cache manager against the database files. The value should be low, with a higher value indicating inadequate physical memory in the server.

One factor that naturally has a significant effect on Active Directory performance is network performance. LDAP queries, replication, and other directory functions take place across the network, so network performance bandwidth and performance can have an effect on Active Directory performance. The reverse is also true: Active Directory can impose additional load on the network, affecting other network traffic for file and print sharing, streaming audio or video, and other functions. Establishing a baseline and monitoring network performance to help you evaluate Active Directory's impact on the network and vice versa is beneficial.

By default, the System Monitor includes a performance object named Network Interface that you can use to monitor network transmission. If Network Monitor is installed on the system, the Network Segment object provides additional useful performance counters.

In addition to monitoring network performance, you also need to monitor server and domain controller performance through System Monitor. First, consider monitoring CPU utilization. In the Performance Monitor, select the Processor object and, as a minimum, monitor the % Processor Time counter. On multiprocessor systems the total processor utilization or individual CPUs can be monitored. If the CPU utilization is high, it's a good indication that it's time for an upgrade, either through replacing the server or adding CPUs. Available disk space for the volumes containing the directory database files, log files, and SYSVOL folder should also be monitored. Use the LogicalDisk object and monitor the Free Megabytes counter to keep tabs on free space in the target volumes. Domain controller performance issues should also be monitored. System Monitor provides two objects that enable you to monitor a broad range of counters for Active Directory – the NTDS object and the Database object. Steps for adding the Database object, if not present in System Monitor, are found in Appendix H.

Monitoring general server performance, network performance, and NTDS/Database performance provides a good indication of domain controller and network health. Monitoring replication will also help to identify potential problems, such as network congestion, that can affect directory replication. The Microsoft Windows 2000 Resource Kit includes a handful of tools to help you monitor replication:

- **Netdiag.exe**: This tool performs a wide range of tests to check network connectivity and DNS consistency. The tool has been updated to include additional tests and also to add functionality to existing tests. Netdiag.exe is a console-based command, and its syntax and options can be viewed by executing Netdiag.exe /? at a console prompt. The help/syntax information is relatively lengthy; therefore, redirect the output to a text file so it can be viewed in Notepad.
- **Repadmin.exe**: This tool permits viewing of replication topology and enables forcing replication events between domain controllers. Use the /showreps switch to display the domain controller's replication partners, when the last replication was attempted, and whether or not it was successful. Use the /showconn switch to view connection objects on the domain controller to determine whether the domain controller is configured to replicate with the appropriate servers.
- **Dcdiag.exe**: This tool performs several tests to check the status and health of a domain controller. These tests verify connectivity, replication, topology integrity, domain controller roles, and other aspects of the domain controller's function.
- **Replmon.exe**: Unlike the previous three tools, Replmon is a Windows-based application. Use Replmon to view the status and performance of directory replication, force synchronization between domain controllers, and view replication topology graphically, as well as generate status reports that include a wide variety of configuration and performance data on the monitored server.

MONITORING APPLICATION PERFORMANCE – AN EXAMPLE

The following section provides a scenario of how performance monitoring can be used to address application performance. This example is not specific to K2 blackpearl but can be used as a foundation for monitoring K2 blackpearl performance.

The key steps in monitoring include creating a performance plan, setting up and executing the test, reviewing the test results, and retesting after changes have been made.

PERFORMANCE PLAN

A very important part of application performance monitoring is planning. The three main decisions to be made during the planning process are:

1. What counters to track
2. When the testing will occur
3. How often to perform the testing

Addressing these three decisions is necessary so that changes are not made to an application without being absolutely sure that those changes are based on accurate performance data.

Determining which counters to track depends on the required performance data. In this example we will be monitoring the cache performance, request performance, and error performance. Therefore the counters for cache, requests, and errors will be included. Several counters may be included; however, the results should be viewed one at a time. One approach to help decide what aspects to monitor is to simply include all of the ASP.NET performance counters and then view them one at a time in the performance monitor. When a specific area is identified as having performance problems, just those counters can be monitored on subsequent tests.

To get the most accurate sampling of data, the applications should be monitored at different times of the day, or for very long periods of time. It is also important to take into consideration when an application will be used the most, during a certain time of the day, or a certain day of the week. Knowing this information drives the decision for the timeframe that will be used to monitor the application. Generally it is recommended to monitor an application for a 24-hour period on a 10-second interval, as it provides an accurate view of the performance for the entire day. In this example, we will be using 15-minute intervals for the sake of convenience.

The third decision for the performance plan is how many times the test will be performed. A couple of tests should be performed before making any major changes to an application to make sure the test results are accurate.

Here is the test plan for this example:

Time: (duration of our test)

> 15 Minutes

Counters: (which counters we will we be monitoring)

> Cache Hit Ratio
> Cache Turnover Rate
> Requests/Sec
> Requests Failed
> Errors Total
> Errors Unhandled

Tests: (the number of tests we will run)

> 1

When: (when we will we run the tests)

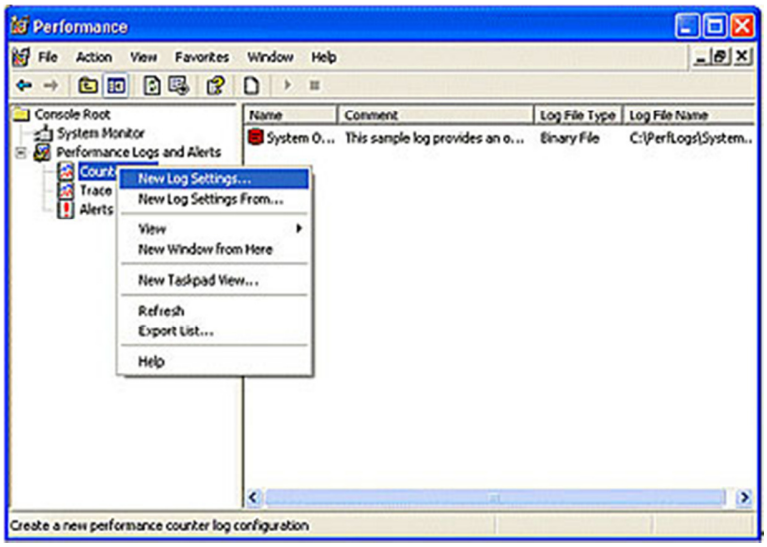
> Test 1 will be run on a weekend day

Note: Only a single test is being used in this example, as it makes it easier to understand. In a real life testing scenario it is a good idea to run multiple tests at different times or on different days.

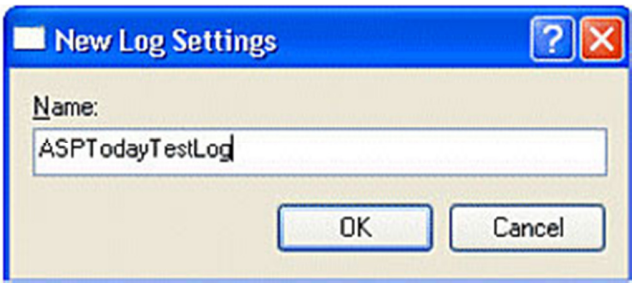
TEST SETUP

Prior to performing the test, the Performance Monitor needs to be set up to record a log file. In order to do this, complete the following steps:

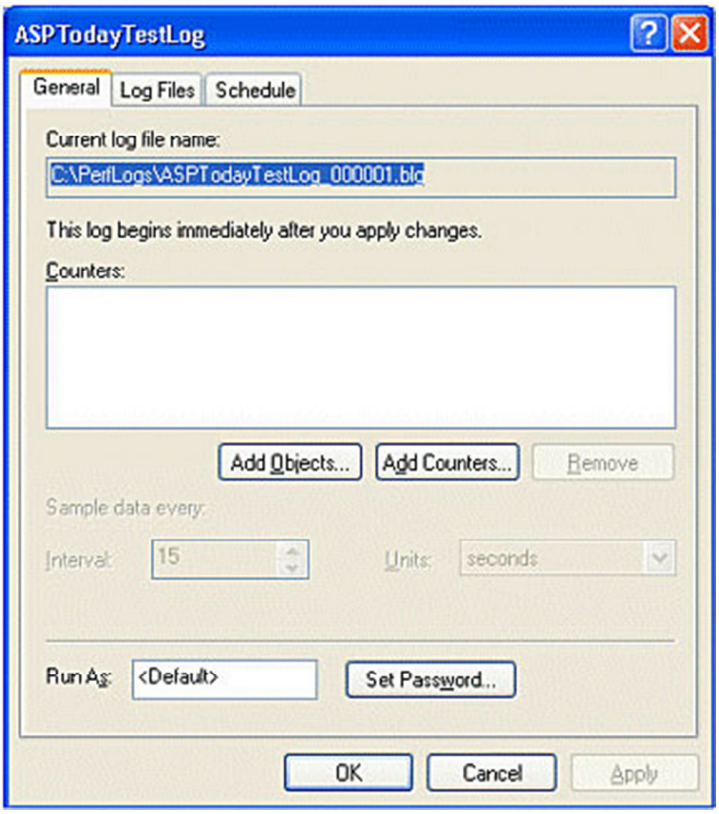
1. Open Performance Monitor and create a new log.



2. Enter a name for the log and click **OK**. In this example, we will call it **ASPTodayTestLog**:



3. The configuration window appears for the Log file. Click the **Add Counters** button.



4. The Add Counters window appears. Use this window to add all counters that should be recorded for the test. Use the options on the window to locate the counters you have elected to monitor. Once you have selected all the monitors, click the **Close** button.

Note: When adding the counters, be sure to select the counter for the application that you want to measure, and not the Total counter, which tracks all the applications on the system.

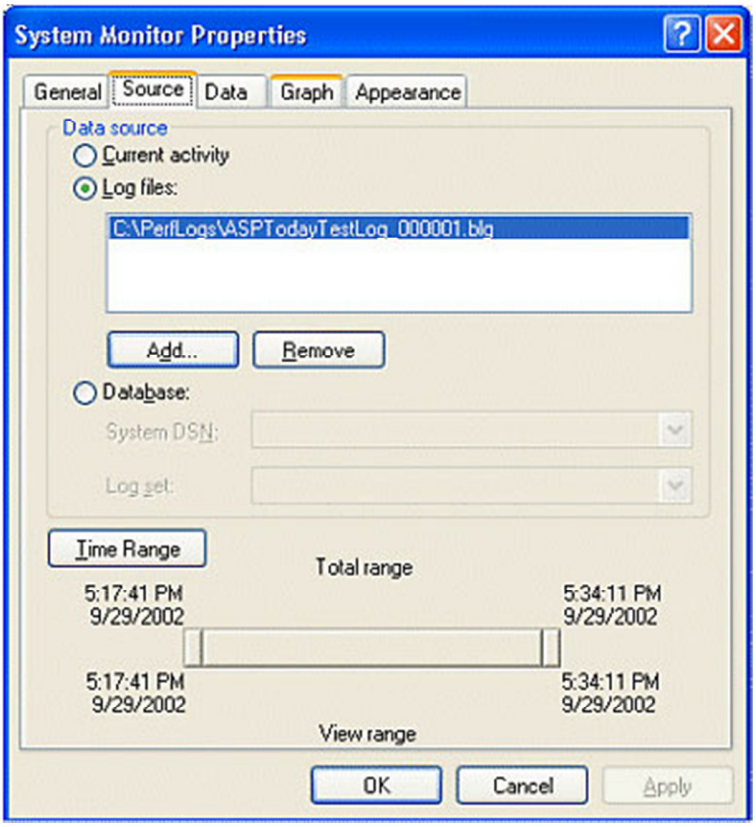
Note: When viewing the test, you can choose what counters to view at that moment.

5. Click the Schedule tab to set the schedule of the log to stop after 15 minutes.

Now the log file is ready to be started for the test. To start the log, first right-click on the log and click **Start**.

REVIEWING THE TEST RESULTS

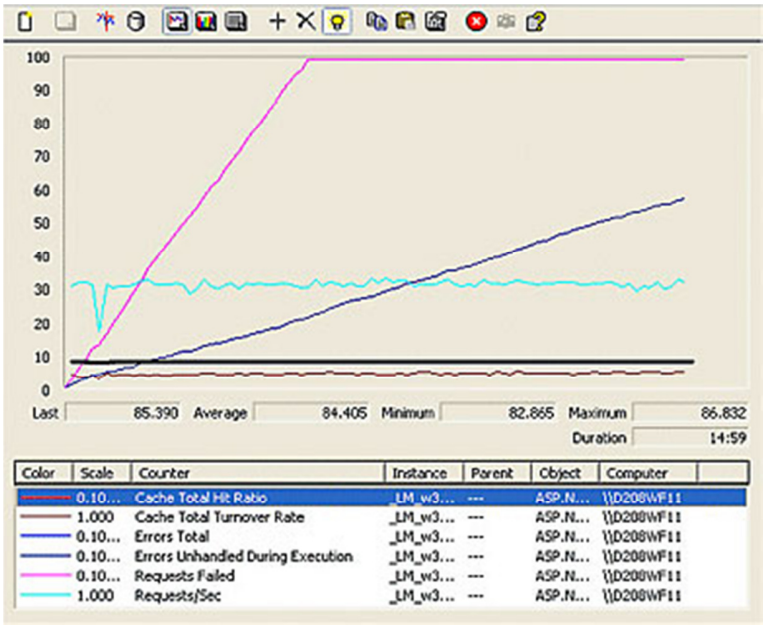
After 15 minutes, the log file can be loaded into the Performance Monitor by right-clicking on where the performance counter data is viewed and choosing Properties:



In this window, set the Source of the monitor to the log file and be sure that the entire time range of the log is being used.

Note: Additional settings can be selected on the other tabs, including setting the different colors of the counters under Appearance and setting the range of the graph under Graph; however, for this example no other changes need to be made.

To start reviewing the results in the Performance Monitor look at the graph:



The graph shows the counters that were added to the log and how they reacted during the test period. The first thing the graph shows is that our application is definitely having some issues with failed requests and errors.

The next sections walk through each counter, what it means, what values should be expected, and how to improve the performance of these areas.

CACHE MONITORING

Caching applications has been made much easier with ASP.NET, and there are many ways to incorporate caching to improve the performance of an application. The following describe the counters that were included in the test to monitor caching and the performance improvements that are gaining from it.

Cache Total Hit Ratio

The Cache Total Hit Ratio is the ratio of hits to misses when trying to access a page from cache. The test results show the Cache Total Hit Ratio close to 84, which means that for every time a page was not found in cache there were 84 times where a page was found. The kinds of readings seen from this counter depend on what duration settings are set for the cache and how many pages use caching. To improve this reading, the time that the busiest pages stay in cache should be increased, and the time that the least used pages stay in cache should be decreased. To determine which pages should have the time increased and which should have the time decreased, examine the IIS logs, or other statistics tools.

Cache Total Turnover Rate

This counter represents the rate at which objects are being added and removed from cache each second. The Cache Total Turnover Rate in this example hovered around 4, which is very good, meaning that a large number of objects were not being added and removed each second. If this counter was too high, then it would mean that too many pages or objects were being added to the cache that did not need to be there. When determining if this

value is too high, consider how many pages are being cached and then see how many are being added or removed each second. If the number is over 10% then there is definitely some room for improvement.

Optimizing Cache

Based on the results for these monitors we know that we could optimize our cache. The first step is to determine what pages are accessed the most either using the IIS log or a statistics program, and then to increase the time that those pages are maintained in cache. For this example, around 5 pages had relatively low duration times set. To change the duration time to the 50 millisecond range, the duration value in the OutputCache declaration at the top of each of those**.aspx** pages needs to be changed. At the end of this example we will re-run the same test and see what impact our improvements have had.

MONITORING ERRORS

Errors in an application can be a major performance issue as well as a user issue. A site that has constant errors will quickly send a user to a competitor's site or to your help desk.

Errors Total

The Errors Total keeps track of the total number of errors that occurs on the site since the service or server was last restarted. This counter includes any parser, compilation, or run-time errors and will count all errors that occur on our site. This is a very important counter to watch since errors should be avoided at any cost.

Errors Unhandled During Execution

This counter tracks the total number of errors that occur during the execution of the page and are handled by the default ASP.NET error handler. That means these errors result in the generic ASP.NET error page being displayed to users. In our example these two counters turned out to be the same, as we most likely had a single error happening somewhere that was not handled by our own code.

Fixing Errors

To find the pages that are causing the errors, use the IIS log of the web server. Open the IIS log and look for any requests that resulted in a 500 http code, which is the code used for server errors. In this example, numerous entries in the log looked like this:

```
22:19:01 192.168.0.101 GET /asptodaysite/WebForm5.aspx 500
```

No other entries resulted in a 500 code, so this page is the logical place to start. When navigating to the page, an error occurred when trying to connect to the SQL server. The problem was with the connection string. After correcting the problem navigate to the page again to test the SQL Server connection.

MONITORING REQUESTS

Monitoring requests counters is just as important as error counters as they directly correlate to users who are being disappointed and inconvenienced. The two counters included in this example monitor the number of requests and the number of requests that might be failing.

Requests/Sec

The Requests/Sec counter tracks the number of HTTP requests that the application is handling. This counter provides the capability to compare levels of traffic when comparing multiple tests.

Requests Failed

This counter tracks the number of requests that have failed since the server or service was last restarted. It includes any requests that fail due to the page not being found, the user not being authorized, or the request timing out. Any of these failed requests result in an ugly message being sent to the users, which should be avoided.

Fixing Request Failed

To identify the failed requests, use the IIS log to find the pages that are causing this failure. When looking through the log, look for any requests that resulted in code of 404, 414 for not found, and 401 for not authorized. Requests Timed out could show up as 500 server errors or 408. In this example, we had a large number of requests that were failing. After scanning through the IIS log, we find numerous 404 error codes.

A couple of different actions can be taken to correct 404 issues:

- The broadest solution is to create a custom 404 page handler and to offer the user suggestions as to where they can find the page they might be looking for.
- Another solution is to find the site that is linking to incorrect pages and fix those links.
- A third solution is to create a dummy page to redirect them to the new page

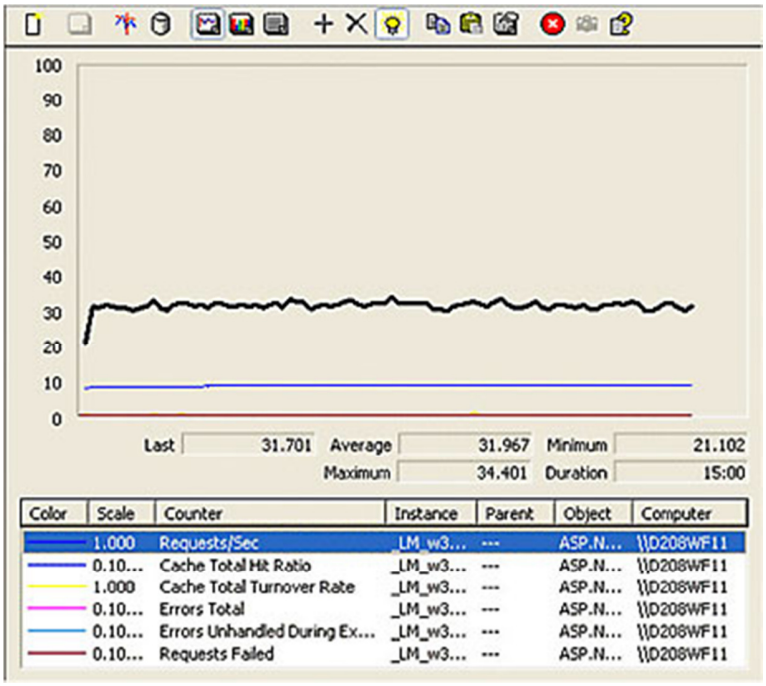
In this example, we will create a simple page to sit where the page was that was not being found. This page will simply notify the user that this page no longer exists, and suggests a page that they are most likely looking for.

RE-TESTING

After making adjustments to the application, it is important to run another identical test to see what type of improvements have been made to the performance of the site. As a review, here are the changes that were made in this example:

- Increased the amount of time that pages are stored in cache
- Found an error caused by a bad SQL Server connection string and fixed the error by entering the correct connection string
- Found some pages causing 404 page not found errors and created "dummy" pages to suggest what page the user might be looking for

Run the test using the exact same setup as the first test. This time the log should automatically save the results as **log2**. Finally, review the results of the second test:



Major improvements have resulted from our small modifications. The Errors and Requests Failed counters did not move at all, meaning we found all of the pages causing errors. Improvements made to the caching of the pages increased the Cache Hit Ratio up into the 90's and even lowered the Turnover Rate as the pages were not being added and removed as often.

CONCLUSION

Performance Monitoring is a complex subject and very dependent on environmental factors. This paper contained some common performance counters for the various architectural pieces of a K2 blackpearl environment. You will notice that there were no hard and fast guidelines for performance metrics. It is important to analyze your environment periodically to keep an eye on the performance. These performance counters can be useful tools for troubleshooting slow performance and justifying scaling out tiers.

Please note that all of the best practices included in this document were sourced from freely accessible support, blog and technical sites, as well as best practices. As there is a very wide scope in system counters to choose for variants of systems and applications, this document is aimed at pointing out the most useful counters for monitoring a typical K2 blackpearl installation.

ADDITIONAL RESOURCES

The following resources were used in compiling the information for this white paper. Please utilize these sites for additional information:

Microsoft TechNet (http://www.microsoft.com/technet/prodtechnol/WindowsServer2003/Library/IIS/8f7a1448-84f0-49d0-ae43-ad9970e2ba66.mspx?mfr=true):

- Web Service Counters
- Web Service Cache Counters
- Internet Information Services Global Counters
- Active Server Pages Performance Counters
- ASP.NET Performance Counters

Microsoft Technet: ASP.NET Performance Monitoring (http://technet2.microsoft.com/WindowsServer/en/library/0b792d7c-30ca-4349-af4c-e000643aa6f31033.mspx?mfr=true)

Apress (formerly ASPToday) (http://microsoft.apress.com/asptodayarchive)

WindowsNetworking.com (http://www.windowsnetworking.com/articles_tutorials/Key-Performance-Monitor-Counters.html)

SQLServerCentral.com (http://www.windowsnetworking.com/articles_tutorials/Key-Performance-Monitor-Counters.html)

APPENDIX A: K2 SERVER COUNTERS

Counter	Description
K2 Processes Started	The number of K2 Process Instances started by the K2 Server across all process definitions, based on when the K2 Host Server service was last started. This is a running count based on up time which can help describe how performant your K2 server is. Combined with the other user counters, such as K2 Worklists Opens and K2 Worklist Items Finished, you can monitor how much load on the K2 Server is created by the end users.
K2 Processes Started Per Second	The number of K2 Process instances started per second by the K2 Server
K2 Processes Total Active	The number of K2 Process instances that are currently in an active state, not based on uptime. This describes the load of the machine based on active process instances, not just the end user impact.
K2 Processes Total Completed	The number of K2 Process instances that are currently in a completed state.
K2 Processes Total Deleted	The number of K2 Process instances that are currently in a deleted state
K2 Processes Total Error	The number of K2 Process instances that are currently in an error state. This counter can be monitored for trends, if the number of process instances in error increases dramatically, then there could be an underlying issue that needs to be investigated.
K2 Processes Total Escalated	The number of K2 Process instances that reached an Escalation. This metric can be used to see if people are performing their tasks on time, which can be used for cultural change or process redesign initiatives.
K2 Processes Total Running	The number of K2 Process instances that are currently in a running state
K2 Processes Total Stopped	The number of K2 Process instances that are currently in a stopped state
K2 Worklist Items Finished	The number of K2 Worklist items that was finished by a user, also based on uptime. This is a running count of how many times an end user actioned a Worklist item.
K2 Worklist Items Finished per second	The number of K2 Worklist items that was finished by a user per

	second, also based on uptime.
K2 Worklists Opens	The number of worklists that have been opened by the K2 Server, either through the API, K2 Worklist Web Part, or on the Workspace. Large numbers of worklist items on a user's worklist can impact performance as it is a memory intensive process to open large worklists.
K2 Worklists Opens per second	The number of worklists that has been opened by the K2 Server per second
Process Memory Usage	The memory usage (in bytes) for the K2 Server Windows process. This counter is not related to process instances or process definitions, but for the K2 Host Server.
Process Modules Loaded	The number of modules loaded by the K2 Server, this number correlates to the number of processes deployed, including process versions with instances that are active.
Process Thread Count	The number of threads running in the K2 Server Windows process, which is a good indicator of how busy the system is.
TCP Bytes Received Per Second	The number of Transmission Control Protocol (TCP) Bytes received by the K2 Server per second, can indicate bandwidth bottlenecks. In a distributed environment, the network connectivity between K2 components can impact performance of the environment. Monitoring this counter can look at the network performance and indicate issues.
TCP Bytes Received Total	The number of TCP Bytes received for the duration of uptime of the K2 Server service.
TCP Bytes Sent Per Second	The number of Transmission Control Protocol (TCP) Bytes sent by the K2 Server per second, can indicate bandwidth bottlenecks. In a distributed environment, the network connectivity between K2 components can impact performance of the environment. Monitoring this counter can look at the network performance and indicate issues.
TCP Bytes Sent Total	The number of TCP Bytes sent for the duration of uptime of the K2 Server service.
TCP Concurrent Connections	The number of Concurrent Client Connections for the duration of the K2 Server running instance, currently shows the same number as the TCP Connections Opened counter due to a known issue.

TCP Connections Opened	The number of TCP connections opened for the duration of the K2 Server running instance, from client to server. This includes connections opened via custom ASP.NET forms, the API, Workspace, and Reports. If this number is high and therefore impacts performance, you can look at introducing connection pooling.
TCP Connections Opened Per Second	The number of TCP connections opened for the duration of the K2 Server running instance per second

APPENDIX B: SERVER COUNTERS

Counter	Description
Processor\ % Processor Time	The Processor\% Processor Time counter determines the percentage of time the processor is busy by measuring the percentage of time the thread of the Idle process is running and then subtracting that from 100 percent. This measurement is the amount of processor utilization. Although you might sometimes see high values for the Processor\% Processor Time counter (70 percent or greater depending on your workload and environment), it might not indicate a problem; you need more data to understand this activity. For example, high processor-time values typically occur when you are starting a new process and should not cause concern.
System\ % Total Processor Time	The Processor\ % Processor Time counter is most often used as a general measure of processor activity on both single-processor and multiprocessor computers. System\ % Total Processor Time is included for monitoring system-wide processor use on multiprocessor computers. On single-processor computers, System\ % Total Processor Time always equals Processor\ % Processor Time. On multiprocessor computers, System\ % Total Processor Time represents the active time of all processors divided by the number of processors.
System\ Processor Queue Length	The System\Processor Queue Length counter gives an indication of how many threads are waiting for execution.
Process\ % Privileged Time	The percentage of non-idle processor time spent in privileged mode. Privileged mode is a processing mode designed for operating system components and hardware-manipulating drivers. It allows direct access to hardware and all memory.
Process\ % Processor Time	Peak utilization can be 100 percent, but utilization should not be sustained at high level (for example, greater than 90 percent). Processor bottlenecks are characterized by a high value for Process\% Processor Time while the network adapter card remains well below capacity.
Process\ % User Time	The percentage of non-idle processor time spent in user mode. User mode is a restricted processing mode designed for applications, environment subsystems, and integral subsystems.
Process\ Priority Base	

Thread\ % Privileged Time	The percentage of non-idle thread time spent in privileged mode. Privileged mode is a processing mode designed for operating system components and hardware-manipulating drivers. It allows direct access to hardware and all memory.
Thread\ % Processor Time	The amount of processor time for a thread.
Thread\ % User Time	The percentage of non-idle thread time spent in user mode. User mode is a restricted processing mode designed for applications, environment subsystems, and integral subsystems.
Thread\ Context Switches/sec	The Thread\Context Switches/sec counter in System Monitor provides another perspective on how the operating system schedules threads to run on the processor. A context switch occurs when the kernel switches the processor from one thread to another. A context switch might also occur when a thread with a higher priority than the running thread becomes ready or when a running thread must wait for some reason (such as an I/O operation). The Thread\Context Switches/sec counter value increases when the thread gets or loses the time of the processor.
Thread\ Priority Base	Threads are scheduled to run based on their scheduling priority. Each thread is assigned a scheduling priority. The priority levels range from zero (lowest priority) to 31 (highest priority). Only the zero-page thread can have a priority of zero. (The zero-page thread is a system thread responsible for zeroing any free pages when there are no other threads that need to run.) Related counters: > Thread\Priority Current
Thread\ Priority Current	The current priority set on the thread. Related counters: > Thread\Priority Base
Thread\ Thread State	Thread State defines a set of all possible execution states for threads. Once a thread is created, it is in at least one of the states until it terminates. Threads created within the common language runtime are initially in the Unstarted state, while external threads that come into the runtime are already in the

	Running state.
Network Interface\ Bytes Total/sec	This counter reflects the number of bytes that are sent and received through the interface, including all framing characters. This counter measures all net traffic that moves through the NIC and includes overhead from media access protocol and transport protocol.
Network Interface\ Bytes Sent/sec	The number of bytes sent per second.
Network Interface\ Bytes Received/sec	The number of bytes received per second.
Protocol_layer_object \ Segments Received/sec	The rate at which segments are received by using the protocol (e.g., TCP).
Protocol_layer_object \ Segments Sent/sec	The rate at which segments are sent by using the protocol (e.g., TCP).
Protocol_layer_object \ Frames Sent/sec	Shows the rate at which data bytes are sent by the computer. This counter shows only the bytes in frames that carry data.
Protocol_layer_object \ Frames Received/sec	Shows the rate at which data bytes are received by the computer. This counter shows only the bytes in frames that carry data.
Server\ Bytes Total/sec	Specifies the total number of bytes the server has sent and received over the network. The values this object monitors gives a good idea of how busy a server is. If this number is consistently at or above 50 percent of the network capacity, you may need to break the server off onto a faster switch or hub.
Server\ Bytes Received/sec	The number of bytes the server has received over the network per second.
Server\ Bytes Sent/sec	The number of bytes the server has sent over the network per second.
Network Segment\ % Network Utilization	Network utilization is the ratio of current network traffic to the maximum traffic that the port can handle. It indicates the bandwidth use in the network. While high network utilization indicates the network is busy, low network utilization indicates the network is idle. When network utilization exceeds the threshold under normal condition, it will cause low transmission speed, intermittence, request delay and so on.

System\System Up Time	The System\System Up Time counter tells how many seconds it has been since the server last rebooted. Related counters: > Process(instance)\Elapsed Time
Process(instance)\Elapsed Time	Using the Process(instance)\Elapsed Time counter provides information on how long a particular process has been running on a server.
Processor (_Total)\%Processor Time	Processor(_Total)\% Processor Time measures the total utilization of the processor by all running processes to provide data on how busy a server is. Note that on multiprocessor machines, Processor(_Total)\% Processor Time actually measures the average processor utilization of the machine (i.e., utilization averaged over all processors). Related counters: > Process(instance)\%Processor Time > Processor(_Total)\% Privileged Time > Processor(_Total)\% User Time
System\Processor Queue Length	The System\Processor Queue Length counter gives an indication of how many threads are waiting for execution.
System\Context Switches/sec	System\Context Switches/sec measures how frequently the processor has to switch from user- to kernel-mode to handle a request from a thread running in user mode, which helps to monitor how the machine's hardware devices are functioning. Related counters: > Processor(_Total)\Interrupts/sec
Memory\Pages/sec	The Memory\Pages/sec counter indicates the number of paging operations to disk during the measuring interval.
Memory\Available Bytes	Indicates how much physical memory is remaining after the working sets of running processes and the cache have been served. Related counters: > Process(instance)\Working Set

	> Memory\Cache Bytes
Memory\Transition Faults/sec	Memory\Transition Faults/sec measures how often recently trimmed pages on the standby list are re-referenced and can be an indicator of insufficient RAM.
Physical Disk (instance)\Disk Transfers/sec	The average number of read and write operations that have occurred on a disk per second. Related counters: > Physical Disk(instance)\% Idle Time

For NWLink performance objects, frame-related counters report only zeroes. Use datagram-based counters for these objects.

APPENDIX C: WEB SERVICE PERFORMANCE COUNTERS

Counter	Description
Bytes (Sent, Received and Transferred) Counters	
Total Bytes Sent	The number of data bytes that have been sent by the WWW service since the service started. This counter is new in IIS 6.0.
Bytes Sent/sec	The rate, in seconds, at which data bytes have been sent by the WWW service.
Total Bytes Received	The total bytes of data that have been received by the WWW service since the service started. This counter is new in IIS 6.0.
Bytes Received/sec	The rate, in seconds, at which data bytes have been received by the WWW service.
Total Bytes Transferred	The total number of bytes of data that have been sent and received by the WWW service since the service started. This counter is new in IIS 6.0.
Bytes Total/sec	The sum of Bytes Sent/sec and Bytes Received/sec.
Files (Sent, Received, and Transferred) Counters	
Total Files Sent	The number of user-mode files that have been sent by the WWW service since the service started. This counter does not include cache hits. Note that this counter does not increment when files are being served from the kernel-mode cache. For more information, see Kernel: URI Cache Hits.
Files Sent/sec	The rate, in seconds, at which files have been sent.
Total Files Received	The number of files that have been received by the WWW service since the service started.
Files Received/sec	The rate, in seconds, at which files have been received by the WWW service.
Total Files Transferred	The sum of Total Files Sent and Total Files Received by the WWW service since the service started. Note that this counter does not increment when files are being served from the kernel-mode cache. For more information, see Kernel: URI Cache Hits in Table D.13.
Files/sec	The rate, in seconds, at which files have been sent and received by the WWW service.
Anonymous Users and NonAnonymous Users Counters	
Current Anonymous Users	The number of users who currently have an anonymous request pending with the WWW service. In IIS 6.0, Current Users (Anonymous or NonAnonymous) is the number of requests currently being worked on by the server.
Current	The number of users who currently have a nonanonymous request pending with the

NonAnonymous Users	WWW service. In IIS 6.0, Current Users (Anonymous or NonAnonymous) is the number of requests currently being worked on by the server.
Total Anonymous Users	The number of users who have established an anonymous request since the WWW service started. This counter does not increment when files are being served from the kernel cache. For more information, see Kernel: URI Cache Hits in Table D.13.
Anonymous Users/sec	The rate, in seconds, at which users have made anonymous requests to the WWW service.
Total NonAnonymous Users	The number of users who have made nonanonymous requests to the WWW service since the service started. For more information, see Kernel: URI Cache Hits in Table D.13.
NonAnonymous Users/sec	The rate, in seconds, at which users have made nonanonymous requests to the WWW service.
Maximum NonAnonymous Users	The maximum number of users who have made concurrent nonanonymous requests to the WWW service since the service started.
Connections and Attempts Counters	
Current Connections	The number of active connections to the WWW service.
Maximum Connections	The maximum number of simultaneous connections made to the WWW service since the service started.
Total Connection Attempts	The number of connections to the WWW service that have been attempted since the service started.
Connection Attempts/sec	The rate, in seconds, at which connections to the WWW service have been attempted since the service started.
Total Logon Attempts	The number of attempts to log on to the WWW service that have occurred since the service started.
Logon Attempts/sec	The rate, in seconds, at which attempts to log on to the WWW service have occurred.
Requests Counters	
Total Options Requests	The number of HTTP requests that have used the OPTIONS method since the WWW service started.
Options Requests/sec	The rate, in seconds, at which HTTP requests that use the OPTIONS method have been made.

Total Get Requests	The number of HTTP requests that have used the GET method since the WWW service started.
Get Requests/sec	The rate, in seconds, at which HTTP requests that use the GET method have been made to the WWW service.
Total Post Requests	The number of HTTP requests that have used the POST method since the WWW service started.
Post Requests/sec	The rate, in seconds, at which requests that use the POST method have been made to the WWW service.
Total Head Requests	The number of HTTP requests that have used the HEAD method since the WWW service started.
Head Requests/sec	The rate, in seconds, at which HTTP requests that use the HEAD method have been made to the WWW service.
Total Put Requests	The number of HTTP requests that have used the PUT method since the WWW service started.
Put Requests/sec	The rate, in seconds, at which HTTP requests that use the PUT method have been made to the WWW service.
Total Delete Requests	The number of HTTP requests that have used the DELETE method since the WWW service started.
Delete Requests/sec	The rate, in seconds, at which HTTP requests that use the DELETE method have been made to the WWW service.
Total Trace Requests	The number of HTTP requests that have used the TRACE method since the WWW service started.
Trace Requests/sec	The rate, in seconds, at which HTTP requests that use the TRACE method have been made to the WWW service.
Total Move Requests	The number of HTTP requests that have used the MOVE method since the WWW service started.
Move Requests/sec	The rate, in seconds, at which HTTP requests that use the MOVE method have been made to the WWW service.
Total Copy Requests	The number of HTTP requests that have used the COPY method since the WWW service started.
Copy Requests/sec	The rate, in seconds, at which HTTP requests that use the COPY method have been

	made to the WWW service.
Total Mkcol Requests	The number of HTTP requests that have used the MKCOL method since the WWW service started.
Mkcol Requests/sec	The rate, in seconds, at which HTTP requests that use the MKCOL method have been made to the WWW service.
Total Propfind Requests	The number of HTTP requests that have used the PROPFIND method since the WWW service started.
Propfind Requests/sec	The rate, in seconds, at which HTTP requests that use the PROPFIND method have been made to the WWW service.
Total Proppatch Requests	The number of HTTP requests that have used the PROPPATCH method since the WWW service started.
Proppatch Requests/sec	The rate, in seconds, at which HTTP requests that use the PROPPATCH method have been made to the WWW service.
Total Search Requests	The number of HTTP requests that have used the SEARCH method since the WWW service started.
Search Requests/sec	The rate, in seconds, at which HTTP requests that use the SEARCH method have been made to the WWW service.
Total Lock Requests	The number of HTTP requests that have used the LOCK method since the WWW service started.
Lock Requests/sec	The rate, in seconds, at which HTTP requests that use the LOCK method have been made to the WWW service.
Total Unlock Requests	The number of HTTP requests that have used the UNLOCK method since the WWW service started.
Unlock Requests/sec	The rate, in seconds, at which HTTP requests that use the UNLOCK method have been made to the WWW service.
Total Other Request Methods	The number of HTTP requests that did not use the OPTIONS, GET, HEAD, POST, PUT, DELETE, TRACE, MOVE, COPY, MKCOL, PROPFIND, PROPPATCH, SEARCH, LOCK, or UNLOCK methods since the WWW service started. Can include LINK or other methods supported by gateway applications.
Other Request Methods/sec	The rate, in seconds, at which HTTP requests that do not use the methods listed for the Total Other Requests Methods counter have been made to the WWW service.

Total Method Requests	The number of HTTP requests that have been made since the WWW service started.
Total Method Requests/sec	The rate, in seconds, at which all HTTP requests have been received.
Errors (Not Found and Locked) Counters	
Total Not Found Errors	The number of requests that have been made since the service started that were not satisfied by the server because the requested document was not found. Usually reported as HTTP error 404.
Not Found Errors/sec	The rate, in seconds, at which requests were not satisfied by the server because the requested document was not found.
Total Locked Errors	The number of requests that have been made since the service started that could not be satisfied by the server because the requested document was locked. Usually reported as HTTP error 423.
Locked Errors/sec	The rate, in seconds, at which requests were not satisfied because the requested document was locked.
CGI Requests and ISAPI Extension Requests Counters	
Current CGI Requests	The number of CGI requests that are being processed simultaneously by the WWW service.
Total CGI Requests	The number of all CGI requests that have been made since the WWW service started.
CGI Requests/sec	The rate, in seconds, at which CGI requests are being processed simultaneously by the WWW service.
Maximum CGI Requests	The maximum number of CGI requests that have been processed simultaneously by the WWW service since the service started.
Current ISAPI Extension Requests	The number of ISAPI extension requests that are being processed simultaneously by the WWW service.
Total ISAPI Extension Requests	The number of ISAPI extension requests that have been made since the WWW service started.
ISAPI Extension Requests/sec	The rate, in seconds, at which ISAPI extension requests are being processed by the WWW service.
Maximum ISAPI Extension Requests	The maximum number of ISAPI extension requests that were processed simultaneously by the WWW service.

OBSOLETE COUNTERS

The table below lists the CAL (client access license) Count counters for the Web Service performance object, which are not valid in IIS 6.0. The value of these counters is always zero.

CAL Count Counters

Counter	Description
Current CAL count for authenticated users	This counter is no longer valid; value is always zero.
Maximum CAL count for authenticated users	This counter is no longer valid; value is always zero.
Total count of failed CAL requests for authenticated users	This counter is no longer valid; value is always zero.
Current CAL count for SSL connections	This counter is no longer valid; value is always zero.
Maximum CAL count for SSL connections	This counter is no longer valid; value is always zero.
Total count of failed CAL requests for SSL connections	This counter is no longer valid; value is always zero.

Other miscellaneous counters, some of which are no longer valid, for the Web Service performance object are included in the table below.

Async I/O Requests, Bandwidth Bytes, and Miscellaneous Counters

Counter	Description
Total Allowed Async I/O Requests	This counter is no longer valid; its value is always zero.
Total Blocked Async I/O Requests	This counter is no longer valid; its value is always zero.
Total Rejected Async I/O Requests	This counter is no longer valid; its value is always zero.
Current Blocked Async I/O Requests	This counter is no longer valid; its value is always zero.
Measured Async I/O Bandwidth Usage	This counter is no longer valid; its value is always zero.
Total Blocked Bandwidth Bytes	This counter is no longer valid; its value is always zero.
Current Blocked Bandwidth Bytes	This counter is no longer valid; its value is always zero.
Service Uptime	The uptime for the WWW service or a Web site.
MaxCalSsl	This counter is no longer valid; its value is always zero.

APPENDIX D: WEB SERVICE CACHE PERFORMANCE COUNTERS

Counter	Description
File Counters for the Web Service Cache Object	
Active Flushed Entries	The number of user-mode cache entries that have been flushed, though memory is still allocated for these entries. The allocated memory will be released after all current transfers complete.
Current File Cache Memory Usage	The number of bytes currently used for the user-mode file cache.
Current Files Cached	The number of files whose content is currently in the user-mode cache.
Total Files Cached	The number of files whose content has been added to the user-mode cache since the WWW service started.
File Cache Hits	The number of successful lookups in the user-mode file cache that have occurred since the WWW service started.
File Cache Hits %	The ratio of user-mode file cache hits to total cache requests that have been made since the WWW service started up.
File Cache Misses	The number of unsuccessful lookups in the user-mode file cache that have been made since the WWW service started.
File Cache Flushes	The number of files that have been removed from the user-mode cache since the WWW service started.
Maximum File Cache Memory Usage	The maximum number of bytes that have been used for the user-mode file cache since the WWW service started.
Total Flushed Files	The number of file handles that have been removed from the user-mode cache since the WWW service started.
URI Counters for the Web Service Cache Object	
Current URIs Cached	The number of URI information blocks that are currently stored in the user-mode cache.
Total URIs Cached	The number of URI information blocks that have been added to the user-mode cache since the WWW service started.
URI Cache Hits	The number of successful lookups that have been made in the user-mode URI cache since the WWW service started.

URI Cache Misses	The number of unsuccessful lookups that have been made in the user-mode URI cache since the WWW service started.
URI Cache Hits %	The ratio of URI Cache Hits to total cache requests that have occurred since the WWW service started.
URI Cache Flushes	The total number of URI cache flushes that have occurred since the WWW service started.
Total Flushed URIs	The number of URI information blocks that have been removed from the user-mode cache since the WWW service started.
Metadata Counters for the Web Service Cache Object	
Current Metadata Cached	The current number of metadata information blocks in the user-mode cache.
Total Metadata Cached	The number of metadata information blocks that have been added to the user-mode cache since the WWW service started.
Metadata Cache Hits	The number of successful lookups in the user-mode metadata cache that have occurred since the WWW service started.
Metadata Cache Misses	The number of unsuccessful lookups in the user-mode metadata cache that have occurred since the WWW service started.
Metadata Cache Hits %	The ratio of successful lookups to total metadata cache requests.
Metadata Cache Flushes	The number of user-mode metadata cache flushes that have occurred since the WWW service started.
Total Flushed Metadata	The number of metadata information blocks that have been removed from the user-mode cache since the WWW service started.
Kernel Counters for the Web Service Cache Object	
Kernel: Current URIs Cached	The number of URI information blocks currently cached by the kernel.
Kernel: Total URIs Cached	The number of URI information blocks that have been added to the kernel URI cache since the WWW service started.
Kernel: URI Cache Hits	The number of successful lookups in the kernel URI cache that have occurred since the WWW service started.
Kernel: URI Cache Hits%	The ratio of Kernel: URI Cache Hits to total cache requests since the WWW service started.

Kernel: URI Cache Hits/sec	The average number of kernel URI cache hits that are being made per second.
Kernel URI Cache Misses	The number of unsuccessful lookups in the kernel URI cache that have occurred since the WWW service started.
Kernel: URI Cache Flushes	The number of kernel URI cache flushes that have occurred since the WWW service started.
Kernel: Total Flushed URIs	The number of URI information blocks that have been removed from the kernel cache since the WWW service started.

APPENDIX E: IIS COUNTERS

The table below lists the Async I/O counters that are no longer valid for the Internet Information Services Global performance object. The value of these counters is always zero.

Async I/O Counters	
Counter	**Description**
Measured Async I/O Bandwidth Usage	This counter is no longer valid; value is always zero.
Total Blocked Async I/O Requests	This counter is no longer valid; value is always zero.
Total Rejected Async I/O Requests	This counter is no longer valid; value is always zero.
Current Blocked Async I/O Requests	This counter is no longer valid; value is always zero.
Measured Async I/O Bandwidth Usage	This counter is no longer valid; value is always zero.

The table below provides the full list of counters for the Internet Information Services Global performance object.

Counter	**Description**
File Cache Counters	
Current Files Cached	The number of files whose content is currently in the cache.
Total Files Cached	The number of files whose content has been added to the cache since the service started.
File Cache Hits	The number of successful lookups in the file cache.
File Cache Misses	The number of unsuccessful lookups in the file cache.
File Cache Hits %	The ratio of File Cache Hits to the total number of cache requests.
File Cache Flushes	The number of file cache flushes that have occurred since the service started.
Current File Cache Memory Usage	The current number of bytes used for the file cache.
Maximum File Cache Memory Usage	The maximum number of bytes used for the file cache.
Flushed and URI Cached Counters	

Active Flushed Entries	The number of cached file handles that will close when all current transfers are complete.
Total Flushed Files	The number of file handles that have been removed from the cache since the service started.
Current URIs Cached	The number of URI information blocks that are currently in the cache.
Total URIs Cached	The number of URI information blocks that have been added to the cache.
URI Cache Hits	The number of successful lookups in the URI cache.
Uri Cache Misses	The number of unsuccessful lookups in the URI cache.
URI Cache Hits %	The ratio of URI Cache Hits to the total number of cache requests.
URI Cache Flushes	The number of URI cache flushes that have occurred since the server started.
Total Flushed URIs	The number of URI information blocks that have been removed from the cache since the service started.
Binary Large Object (BLOB) Counters	
Current BLOBs Cached	The BLOB information blocks currently in the cache.
Total BLOBs Cached	The number of BLOB information blocks that have been added to the cache.
BLOB Cache Hits	The number of successful lookups in the BLOB cache.
BLOB Cache Misses	The number of unsuccessful lookups in the BLOB cache.
BLOB Cache Hit %	The ratio of BLOB Cache Hits to the total number of cache requests.
BLOB Cache Flushes	The number of BLOB cache flushes that have occurred since the service started.
Total Flushed BLOBs	The number of BLOB information blocks that have been removed from the cache since the service started.

APPENDIX F: ASP PERFORMANCE COUNTERS

Counter	Description
ASP Debugging and Errors Counters	
ASP Debugging Requests	The number of requests for debugging documents that have been made since the WWW service started.
Errors During Script Runtime	The number of requests that failed because run-time errors occurred.
Errors From ASP Preprocessor	The number of requests that failed because preprocessor errors occurred.
Errors From Script Compilers	The number of requests that failed because script compilation errors occurred.
Errors/sec	The average number of errors that occurred per second.
Requests Counters for ASP Pages	
Request Bytes In Total	The total size, in bytes, of all requests.
Request Bytes Out Total	The total size, in bytes, of responses sent to clients. This total does not include standard HTTP response headers.
Request Execution Time	The number of milliseconds that it took to execute the most recent request.
Request Wait Time	The number of milliseconds that the most recent request waited in the queue.
Requests Disconnected	The number of requests that were disconnected because communication failed.
Requests Executing	The number of requests that are currently executing.
Requests Failed Total	The number of requests that failed due to errors, authorization failure, and rejections.
Requests Not Authorized	The number of requests that failed because access rights were insufficient.
Requests Not Found	The number of requests that were made for files that were not found.
Requests Queued	The number of requests that are waiting in the queue for service.
Requests Rejected	The number of requests that were not executed because there were insufficient resources to process them.

Requests Succeeded	The number of requests that executed successfully.
Requests Timed Out	The number of requests that timed out.
Requests Total	The number of requests that have been made since the service started.
Requests/sec	The average number of requests that were executed per second.
Script Engines, Sessions, Templates, and Transactions Counters	
Script Engines Cached	The number of script engines in the cache.
Script Engine Cache Hit Rate	The percentage of requests that were found in the script engine cache.
Engine Flush Notifications	The number of engines invalidated in the cache because change notification occurred.
Session Duration	The length of time that the most recent session lasted, in milliseconds.
Sessions Current	The number of sessions currently being serviced.
Sessions Timed Out	The number of sessions that have timed out.
Sessions Total	The number of sessions that have run since the service was started.
Templates Cached	The number of templates that are currently cached.
Template Cache Hit Rate	The percentage of requests that have been found in the template cache.
Template Notifications	The number of templates that have been invalidated in the cache because change notification occurred.
In Memory Templates Cached	The number of compiled templates that are cached in memory.
In Memory Template Cache Hit Rate	The percentage of requests that have been found in the memory cache.
Transactions Aborted	The number of transactions that have been aborted.
Transactions Committed	The number of transactions that have been committed. This counter increments after page execution if the transaction does not abort.
Transactions Pending	The number of transactions that are in progress.
Transactions Total	The number of transactions that have occurred since the service was started.
Transactions/sec	The average number of transactions that have been started, per second.

APPENDIX G: ASP.NET PERFORMANCE COUNTERS

ASP.NET supports the following ASP.NET system performance counters, which aggregate information for all ASP.NET applications on a Web server computer, or, alternatively, apply generally to a system of ASP.NET servers running the same applications.

Counter	Description
Application Restarts and Applications Running Counters	
Application Restarts	The number of times that an application has been restarted since the Web service started. Application restarts are incremented with each Application_OnEnd event. An application restart can occur because changes were made to the Web.config file or to assemblies stored in the application's \Bin directory, or because too many changes occurred in Web Forms pages. Sudden increases in this counter can mean that your Web application is shutting down. If an unexpected increase occurs, be sure to investigate it promptly. This value resets every time IIS is restarted.
Applications Running	The number of applications that are running on the server computer.
Requests Counters	
Requests Disconnected	The number of requests that were disconnected because a communication failure occurred.
Requests Queued	The number of requests in the queue waiting to be serviced. If this number increases as the number of client requests increases, the Web server has reached the limit of concurrent requests that it can process. The default maximum for this counter is 5,000 requests. You can change this setting in the computer's Machine.config file.
Requests Rejected	The total number of requests that were not executed because insufficient server resources existed to process them. This counter represents the number of requests that return a 503 HTTP status code, which indicates that the server is too busy.
Request Wait Time	The number of milliseconds that the most recent request waited in the queue for processing.
State Server Sessions Counters	
State Server Sessions Abandoned	The number of user sessions that were explicitly abandoned. These are sessions that have been ended by specific user actions, such as closing the browser or navigating to another site.
State Server Sessions Active	The number of active user sessions.

State Server Sessions Timed Out	The number of user sessions that are now inactive. In this case, the user is inactive, not the server.
State Server Sessions Total	The number of sessions created during the lifetime of the process. This counter represents the cumulative value of State Server Sessions Active, State Server Sessions Abandoned, and State Server Sessions Timed Out counters.
Worker Process Counters	
Worker Process Restarts	The number of times that a worker process restarted on the server computer. A worker process can be restarted if it fails unexpectedly or when it is intentionally recycled. If worker process restarts increase unexpectedly, investigate immediately.
Worker Processes Running	The number of worker processes that are running on the server computer.

ASP.NET APPLICATIONS PERFORMANCE COUNTERS

The following application performance counters, can be used to monitor the performance of a single instance of an ASP.NET application. A unique instance of these counters, named __Total__, aggregates counters for all applications on a Web server. The __Total__ instance is always available. When no applications are running on the server, the counters display zero.

Counter	Description
Anonymous Requests Counters	
Anonymous Requests	The number of requests that use anonymous authentication.
Anonymous Requests/sec	The average number of requests that have been made per second that use anonymous authentication.
Cache Total Counters	
Cache Total Entries	The total number of entries in the cache. This counter includes both internal use of the cache by the ASP.NET framework and external use of the cache through exposed APIs.
Cache Total Hits	The total number of responses served from the cache. This counter includes both internal use of the cache by the ASP.NET framework and external use of the cache through exposed APIs.
Cache Total Misses	The number of failed cache requests. This counter includes both internal use of the cache by ASP.NET and external use of the cache through exposed APIs.

Cache Total Hit Ratio	The ratio of cache hits to cache misses. This counter includes both internal use of the cache by ASP.NET and external use of the cache through exposed APIs.
Cache Total Turnover Rate	The number of additions to and removals from the cache per second. Use this counter to help determine how efficiently the cache is being used. If the turnover rate is high, the cache is not being used efficiently.
Cache API Counters	
Cache API Entries	The total number of entries in the application cache.
Cache API Hits	The total number of requests served from the cache when it was accessed only through the external cache APIs. This counter does not track use of the cache internally by ASP.NET.
Cache API Misses	The total number of requests to the cache that failed when the cache was accessed through the external cache APIs. This counter does not track use of the cache internally by ASP.NET.
Cache API Hit Ratio	The cache hit-to-miss ratio when the cache was accessed through external cache APIs. This counter does not track use of the cache internally by ASP.NET.
Cache API Turnover Rate	The number of additions to and removals from the cache per second, when it is used through the external APIs (excluding internal use by the ASP.NET framework). This counter is useful for determining how effectively the cache is being used. If the turnover is great, then the cache is not being used effectively.
Errors Counters	
Errors During Preprocessing	The number of errors that occurred during parsing. Excludes compilation and run-time errors.
Errors During Compilation	The number of errors that occurred during dynamic compilation. Excludes parser and run-time errors.
Errors During Execution	The total number of errors that occurred during the execution of an HTTP request. Excludes parser and compilation errors.
Errors Unhandled During Execution	The total number of unhandled errors that occurred during the execution of HTTP requests. An unhandled error is any uncaught run-time exception that escapes user code on the page and enters the ASP.NET internal error-

	handling logic. Exceptions occur in the following circumstances: > When custom errors are enabled, an error page is defined, or both. > When the Page_Error event is defined in user code and either the error is cleared (by using the HttpServerUtility.ClearError method) or a redirect is performed.
Errors Unhandled During Execution/sec	The number of unhandled exceptions that occurred per second during the execution of HTTP requests.
Errors Total	The total number of errors that occurred during the execution of HTTP requests. Includes parser, compilation, or run-time errors. This counter represents the sum of the Errors During Compilation, Errors During Preprocessing, and Errors During Execution counters. A well-functioning Web server should not generate errors.
Errors Total/sec	The average number of errors that occurred per second during the execution of HTTP requests. Includes any parser, compilation, or run-time errors.
Output Cache Counters	
Output Cache Entries	The total number of entries in the output cache.
Output Cache Hits	The total number of requests serviced from the output cache.
Output Cache Misses	The number of output-cache requests that failed per application.
Output Cache Hit Ratio	The percentage of total requests that were serviced from the output cache.
Output Cache Turnover Rate	The average number of additions to and removals from the output cache per second. If the turnover is great, the cache is not being used effectively.
Request Bytes Counters	
Request Bytes In Total	The total size, in bytes, of all requests.
Request Bytes Out Total	The total size, in bytes, of responses sent to a client. This does not include standard HTTP response headers.
Requests Counters	
Requests Executing	The number of requests that are currently executing.

Requests Failed	The total number of failed requests. All status codes greater than or equal to 400 increment this counter. Note: Requests that cause a 401 status code increment this counter and the Requests Not Authorized counter. Requests that cause a 404 or 414 status code increment this counter and the Requests Not Found counter. Requests that cause a 500 status code increment this counter and the Requests Timed Out counter.
Requests In Application Queue	The number of requests in the application request queue.
Requests Not Found	The number of requests that failed because resources were not found (status code 404, 414).
Requests Not Authorized	The number of requests that failed because of lack of authorization (status code 401).
Requests Succeeded	The number of requests that executed successfully (status code 200).
Requests Timed Out	The number of requests that timed out (status code 500).
Requests Total	The total number of requests that have been made since the service started.
Requests/sec	The average number of requests that have been executed per second. This counter represents the current throughput of the application.
Session State and SQL Server Connections Counters	
Session State Server Connections Total	The total number of session-state connections that were made to a computer on which out-of-process session-state data is stored.
Session SQL Server Connections Total	The total number of session-state connections that were made to the Microsoft® SQL Server™ database in which session-state data is stored.
Sessions Counters	
Sessions Active	The number of sessions that are active.
Sessions Abandoned	The number of sessions that have been explicitly abandoned.
Sessions Timed Out	The number of sessions that timed out.

Sessions Total	The total number of sessions.
Transactions Counters	
Transactions Aborted	The number of transactions that were aborted.
Transactions Committed	The number of transactions that were committed. This counter increments after page execution if the transaction does not abort.
Transactions Pending	The number of transactions that are in progress.
Transactions Total	The total number of transactions that have occurred since the service was started.
Transactions/sec	The average number of transactions that were started per second.
Miscellaneous Counters for ASP.NET Applications	
Compilations Total	The total number of times that the Web server process dynamically compiled requests for files with .aspx, .asmx, .ascx, or .ashx extensions (or a code-behind source file). Note: This number initially climbs to a peak value as requests are made to all parts of an application. After compilation occurs, however, the resulting binary compilation is saved on disk, where it is reused until its source file changes. This means that, even when a process restarts, the counter can remain at zero (be inactive) until the application is modified or redeployed.
Debugging Requests	The number of requests that occurred while debugging was enabled.
Pipeline Instance Count	The number of active request pipeline instances for the specified ASP.NET application. Because only one execution thread can run within a pipeline instance, this number represents the maximum number of concurrent requests that are being processed for a specific application. In most circumstances, it is better for this number to be low when the server is busy, because this means that the CPU is well used.

APPENDIX H: ACTIVE DIRECTORY AND NETWORK PERFORMANCE COUNTERS

The following counters are available for monitoring Active Directory and network performance.

Network Transmission Counters

Counter	Description
Packets Outbound Discarded	The length of the outbound packet queue, by number of packets waiting in the queue. A queue with a few items indicates acceptable performance, but longer queues indicate the NIC is waiting for the network and is not keeping pace with the server, indicating a bottleneck.
Bytes Total/sec	The rate at which bytes are sent and received on the interface. A higher number indicates better performance. Track the performance of each network interface to identify high utilization per interface and determine whether you need to use switches to segment the network or increase bandwidth.
Broadcast Frames Received/sec	This counter lets you define a baseline over time against which to evaluate variations in network traffic.
% Network Utilization	This counter provides a good indication of the bandwidth utilization for the local segment and enables you to evaluate the impact of certain network events—such as replication—on network bandwidth. Consider 30 percent utilization a maximum for unswitched Ethernet. Adjust your acceptable benchmark based on your network topology.
Total Frames Received/sec	Use this counter to monitor network-wide traffic and determine when switches and routers are becoming saturated, indicating a need for additional segmenting.

Server and Domain Controller (NTDS Object) Counters

Counter	Description
DRA Inbound Bytes Total/sec	This counter shows total bytes received through replication per second. Lack of activity indicates that the network is slowing down replication.
DRA Inbound Object Updates Remaining in Packet	This counter shows the number of object updates received for replication that have not yet been applied to the local server. The value should be low, with a higher value indicating that the hardware is incapable of adequately servicing replication (warranting a server upgrade).
DRA Outbound Bytes Total/sec	This counter shows the total bytes sent per second. Lack of activity indicates that the hardware or network is slowing down replication.

DRA Pending Replication Synchronizations	This counter indicates the replication backlog on the server. This value should be low, with a higher value indicating that the hardware is not adequately servicing replication.
DS Threads in Use	This counter shows the number of threads in use by Active Directory, with a lack of activity typically pointing to network problems that are preventing client requests from succeeding.
Kerberos Authentications/sec	This counter shows the number of Kerberos authentications on the server per second. A lack of activity can indicate network problems that are preventing authentication requests from succeeding.
LDAP Bind Time	This counter shows the time required for completion of the last LDAP binding, with a higher value pointing to either hardware or network performance problems.
LDAP Client Sessions	This counter shows the number of connected LDAP client sessions, with a lack of activity pointing to network problems.
LDAP Searches/sec	This counter shows the number of LDAP searches per second performed by clients in the directory. A lack of activity points to network problems.
LDAP Successful Binds/sec	This counter shows the number of successful LDAP binds per second, with a lack of activity pointing to network problems.
NTLM Authentications	This counter shows the number of NTLM authentications per second handled by the domain controller (from Windows 98 and Windows NT clients). A lack of activity points to network problems.

Server and Domain Controller (Database Object) Counters

Counter	Description
Cache % Hit	This counter shows the percentage of database page requests handled by the cache, thereby not causing a file I/O. A lack of activity can indicate that the server has insufficient physical memory.
Cache Page Fault Stalls/sec	This counter shows the number of page faults per second that go unserviced due to lack of available pages in the database cache. A value other than zero indicates insufficient physical memory in the server.
Cache Page Faults/sec	This counter shows the number of page requests per second that cause the database cache to allocate new pages from the cache. This value should be low, with a higher value indicating insufficient physical memory in the server.

File Operations Pending	This counter shows the number of file operations for the database file(s) currently pending by the operating system. The value should be low, with a higher value indicating insufficient physical memory and/or inadequate CPU availability or performance.
File Operations/sec	This counter shows the number of file operations per second generated by the database cache manager against the database files. The value should be low, with a higher value indicating inadequate physical memory in the server.
Log Record Stalls/sec	This counter shows the number of log records per second that could not be added to the log buffers because the buffers were full. The value should be zero or close to zero, with a higher value indicating inadequate physical memory in the server.
Log Threads Waiting	This counter indicates the number of threads waiting on pending log writes. The value should be low, with a higher value indicating insufficient physical memory, poor disk performance, or poor disk structuring.
Table Open Cache Hits/sec	This counter shows the number of directory database tables open per second from the cache. A high value indicates better caching, with a lower value typically indicating inadequate physical memory in the server.

If the Database object is not present in the System Monitor, use the following steps to add it:

1. Create a new folder to contain the Database object's DLL. In this example, assume you create the folder C:\dataperf.
2. Copy the file %systemroot%\System32\Esentprf.dll to the \dataperf folder.
3. Create the following registry keys, if they do not already exist:
   HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\ESENT
   HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\ESENT\Performance
4. Create the following values under the Performance key:
   Open : REG_SZ : OpenPerformanceData
   Collect : REG_SZ : CollectPerformanceData
   Close : REG_SZ : ClosePerformanceData
   Library : REG_SZ : C:\Performance\esentprf.dll
5. Open a command console in %systemroot%\System32 and execute the following command:
   Lodctr.exe Esentperf.ini