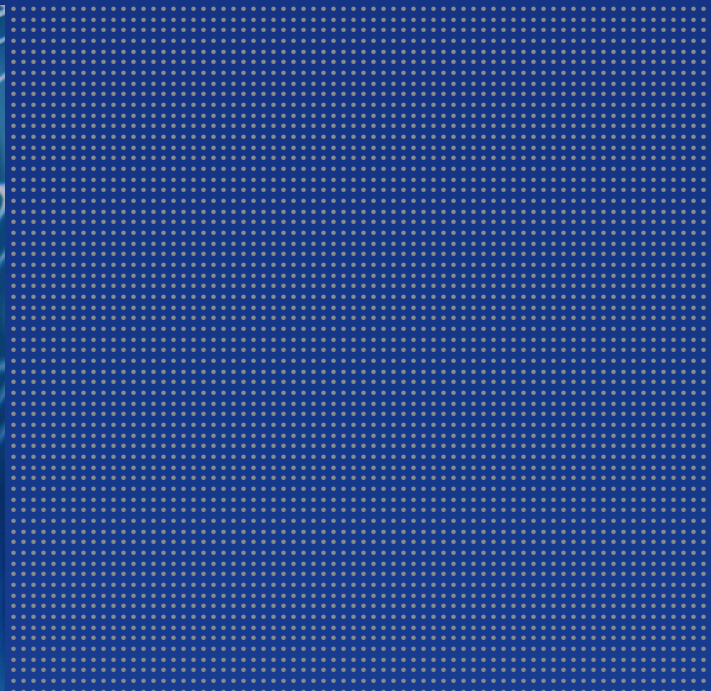


SQL Server Monitoring With Argent SQL Monitor

A R G E N T



Contents

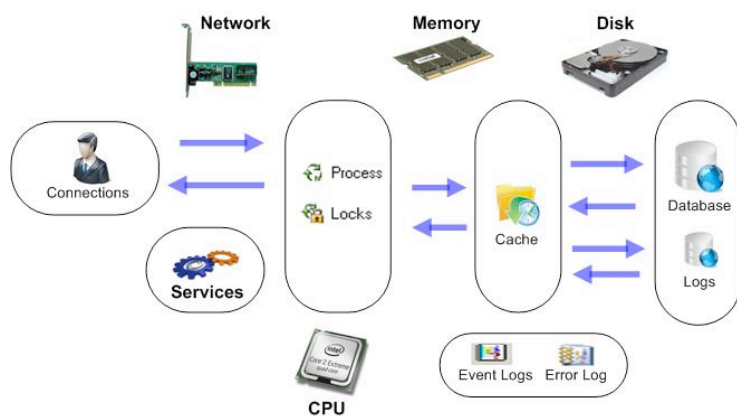
Argent SQL Monitoring - Overview	3
Create a Baseline.	3
Bottleneck Analysis	4
General Analysis	4
Argent SQL Monitoring - Server Performance Counters	5
Server Baseline Performance Counters	5
CPU Counters	5
Memory Counters	6
Disk Counters	7
Network Counters	8
Argent SQL Monitoring - SQL Performance Counters	9
SQL Baseline Performance Counters	9
SQL Data File Space	11
SQL Concurrent Users Rule	11
SQL Lock Rules	11
SQL Query Rule	12
Argent SQL Monitoring – Log Files	14
SQL Server Error Log	14
Windows Application Event Log	15
Security - Connection Auditing	16
C2 Auditing	18
Argent SQL Monitoring - Job Status	19
Argent SQL Monitoring - Availability	20
Argent SQL Monitoring - Mirroring	21
SQL Server Performance – What are DMVs and DMFs?	25
Dynamic Management Views and Dynamic Management Functions	25
Monitoring Connections (Example DMV Usage)	26
Appendix A – SQL Server Performance Objects	28

Argent SQL Monitoring - Overview

Create a Baseline

SQL Server database systems are more than ever before being chosen as the preferred backend database solution for business's critical systems. As a result of this, more users than ever before are stopped in their daily activity when the database is not available, or in a bad shape. It's your responsibility as a DBA to know the health of your SQL Server so you can take proactive action to minimize chances anything bad happens to your system.

If you have the right monitoring in place, your system will tell you how it feels and will warn you well before it reaches the panic phase. Through its performance behavior, you can spot potential problems not yet visible to the end-user. There are a number of critical areas and components that make up a SQL server and these are tightly coupled with the server's resources they are hosted on, the following diagram gives a high level view of the primary areas that need to be considered when monitoring a SQL Server Instance.



This document will suggest performance and health metrics to include in your baseline configuration. Because you know your application better than anyone, the appropriate performance counters that apply to your particular environment will be up to you – however we will outline the main areas that most DBA use and areas that are recommended by Microsoft SQL Specialists.

Argent SQL Monitor provides a myriad of methods to capture and alert all of the critical areas shown above and will allow the baseline to be tracked, this document will provide best practice approaches to achieving this.

One of the important aspects of monitoring is the when interval. Use of the performance counters should not add any overhead to your SQL Server (except for the disk I/O required to record them). If you find that your baseline data is taking up too much disk space, you can use a larger interval between data samples. Be warned, however, that the larger the interval, the less accurate your graph.

Because of these performance concerns, the use of performance counters needs to be properly implemented. You will need to test the number of counters and frequency of collection that best suites your environment. For the initial baseline, however, it is recommended that as many counters as desired be used with the highest frequency available.

SQL Server monitoring typically provides metrics to tracking common Performance problems such as:

- High CPU Utilization
- Disk I/O Bottlenecks
- Memory Bottlenecks
- Slow Running Queries
- Blocking and Deadlocking Issues

Bottleneck Analysis

Why is my system running slow?

With the five performance counters listed below, you can quickly get an overall impression of how healthy a system is - and where the problems might exist. The idea here is to pick counters that will be at low or zero values when the system is healthy and at high values when something is overloaded. A 'perfectly healthy' system would show all counters flat lined at zero.

(Perfection is unattainable, so you'll probably never see all of these counters flat lined at zero in real life. The CPU will almost always have a few items in queue.)

- **Processor utilization**

- o **System: Processor Queue Length** - number of threads queued and waiting for CPU time.

- **Memory utilization**

- o **Memory: Pages Input/Sec** - The best indicator of whether you are memory-bound, this counter shows the rate at which pages are read from disk to resolve hard page faults. In other words, the number of times the system was forced to retrieve something from disk that should have been in RAM. Occasional spikes are fine, but this should generally flat line at zero.

- **Disk Utilization**

- o **Physical Disk: Current Disk Queue Length** - this is probably the single most valuable counter to watch. It shows how many read or write requests are waiting to execute to the disk. For single disks, it should idle at 2-3 or lower, with occasional spikes being okay. For RAID arrays, divide by the number of active spindles in the array; again try for 2-3 or lower. Because a shortage of RAM will tend to beat on the disk, look closely at the *Memory\Pages Input/Sec* counter if disk queue lengths are high.

- **Network Utilization**

- o **Network Interface: Output Queue Length** - is the number of packets in queue waiting to be sent. If there is a sustained average of more than two packets in queue, you should be looking to resolve a network bottleneck.
- o **Network Interface: Packets Received Errors** - packet errors that kept the TCP/IP stack from delivering packets to higher layers. This value should stay low.

General Analysis

How hard is the system working?

The following counters are a good overview of general activity of the system.

- **Processor Utilisation**

- o **Processor: % Processor Time** - just a handy idea of how 'loaded' the CPU is at any given time. Don't confuse 100% processor utilization with a slow system though - processor queue length, mentioned above, is much better at determining this.

- **Memory Utilisation**

- o **Process: Working Set\Total** (or per specific process) - this basically shows how much memory is in the *working set*, or currently allocated RAM.
- o **Memory: Available MBytes** - amount of free RAM available to be used by new processes.

- **Disk Utilisation**

- o **PhysicalDisk: Bytes/sec\Total** (or per process) - shows the number of bytes per second being written to or read from the disk.

- **Network Utilisation**

- o **Network Interface\Bytes Total/Sec** - Measures the number of bytes sent or received.

Argent SQL Monitoring - Server Performance Counters

Server Baseline Performance Counters

The following counters should be included in your Server baseline:

CPU Counters

Processor - % Processor Time - Defined as the percentage of time that the processor is executing a non-Idle thread. A consistent state of 80 to 90 percent may indicate the need for a CPU upgrade or the addition of more processors

- Less than 60% consumed = Healthy
- 51% – 90% consumed = Monitor or Caution
- 91% – 100% consumed = Critical or Out of Spec

System: Processor Queue Length - This counter corresponds to the number of threads waiting for processor time. A processor bottleneck develops when threads of a process require more processor cycles than are available. If more than a few processes are trying to utilize the processor's time, you might need to install a faster processor or an additional processor if you are using a multiprocessor system.

- If each processor has 10 or more threads waiting - If this threshold is broken, then the processor(s) may be at capacity
- If each processor has 20 or more threads waiting - If this threshold is broken, then the processor(s) are beyond capacity

Note: If the CPU is very busy (90 percent and higher utilization) and the PQL average is consistently higher than 2 per processor, you may have a processor bottleneck that could benefit from additional CPUs. Or, you could reduce the number of threads and queue more at the application level. This will cause less context switching, and less context switching is good for reducing CPU load. The common reason for a PQL of 2 or higher with low CPU utilization is that requests for processor time arrive randomly and threads demand irregular amounts of time from the processor. This means that the processor is not a bottleneck but that it is your threading logic that needs to be improved)

Processor: % Privileged Time - This counter corresponds to the percentage of time the processor is spending executing Windows kernel commands such as processing SQL Server I/O requests. If this counter is consistently high when the **Physical Disk** counters is high, consider a faster or more efficient disk subsystem.

- Consistently over 75 percent indicates a bottleneck.

Note: Different disk controllers and drivers use different amounts of kernel processing time. Efficient controllers and drivers use less privileged time, leaving more processing time available for user applications, increasing overall throughput.

System: Context Switches/sec - Context switching happens when a higher priority thread preempts a lower priority thread that is currently running or when a high priority thread blocks. High levels of context switching can occur when many threads share the same priority level. This often indicates that there are too many threads competing for the processors on the system. If you do not see much processor utilization and you see very low levels of context switching, it could indicate that threads are blocked.

- High context switches/sec – more than 5000 context switches per second
- Very high context switches/sec – more than 15,000 context switches per second

Memory Counters

To monitor for a low-memory condition, start with the following object counters:

Memory: Available Bytes - indicates how many bytes of memory are currently available for use by processes. Low values for the **Available Bytes** counter can indicate that there is an overall shortage of memory on the computer or that an application is not releasing memory

- Low on available memory – less than 10% available
- Very low on available memory – less than 5% available
- Decreasing trend of 10MB's per hour. This could indicate a memory leak.

Memory: Pages/sec - indicates the number of pages that either were retrieved from disk due to hard page faults or written to disk to free space in the working set due to page faults. A high rate for the **Pages/sec** counter could indicate excessive paging.

- **High pages/sec – greater than 1000** (If it's higher than 1000, the system is could be beginning to run out of memory. Consider reviewing the processes to see which processes are taking up the most memory or consider adding more memory)
- **Very high average pages/sec – greater than 2500** (If this is greater than 2500, the system could be experiencing system-wide delays due to insufficient memory. Consider reviewing the processes to see which processes are taking up the most memory or consider adding more memory)
- **Critically high average pages/sec – greater than 5000** (If this is greater than 5000. If so, the system is most likely experiencing delays due to insufficient memory. Consider reviewing the processes to see which processes are taking up the most memory or consider adding more memory)

Note: 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.

Memory: Free System Page Table Entries - (Free System Page Table Entries is the number of page table entries not currently in used by the system. This analysis determines if the system is running out of free system page table entries (PTEs) by checking if there is less than 5,000 free PTE's with a Warning if there is less than 10,000 free PTE's. Lack of enough PTEs can result in system wide hangs)

- Running low on PTE's – less than 10,000 (If the free PTEs are under 10,000 the system is close to a system wide hang)
- Critically low on PTE's – less than 5000 (If the free PTEs are under 5000 the system is close to a system wide hang)

Memory: Page Faults/sec - make sure that the disk activity is not caused by paging. Paging can be caused by the following:

- Processes configured to use too much memory.
- File system activity.

If you have more than one logical partition on the same hard disk, use the **Logical Disk** counters instead of the **Physical Disk** counters. Looking at the logical disk counters will help you determine which files are heavily accessed. After you have found the disks with high levels of read/write activity, look at the read-specific and write-specific counters (for example, **Logical Disk: Disk Write Bytes/sec**) for the type of disk activity that is causing the load on each logical volume.

Paging File % Usage - this shows the percentage of the page file being utilized. If you see more than 70 percent of your page file being utilized, look into more RAM for your server.

Disk Counters

PhysicalDisk: % Disk Time - the percentage of time that the disk is busy with read/write activity.

PhysicalDisk: Avg. Disk Queue Length - how many system requests are waiting for disk access.

Note: If the **PhysicalDisk: % Disk Time** counter is high (more than 90 percent), check the **Physical Disk: Current Disk Queue Length** counter to see how many system requests are waiting for disk access. The number of waiting I/O requests should be sustained at no more than 1.5 to 2 times the number of spindles making up the physical disk. Most disks have one spindle, although redundant array of inexpensive disks (RAID) devices usually have more.

Use the values of the **Current Disk Queue Length** and **% Disk Time** counters to detect bottlenecks within the disk subsystem. If **Current Disk Queue Length** and **% Disk Time** counter values are consistently high, consider:

- Using a faster disk drive.
- Moving some files to an additional disk or server.
- Adding additional disks to a RAID array, if one is being used.

If you are using a RAID device, the **% Disk Time** counter can indicate a value greater than 100 percent. If it does, use the **PhysicalDisk: Avg. Disk Queue Length** counter to determine how many system requests, on average, are waiting for disk access.

Applications and systems that are I/O-bound may keep the disk constantly active (Check **Memory: Page Faults/sec** to make sure that the disk activity is not caused by paging).

Average Disk/sec Read & Average Disk/sec Write - Measure of disk latency. Lower values are better but this can vary and is dependent on the size and nature of I/Os being issued. Numbers also vary across different storage configurations (cache size/utilization can impact this greatly). On well-tuned I/O subsystems, ideal values would be:

- 1–5 ms for Log (ideally 1 ms on arrays with cache)
- 4–20 ms for Data on OLTP systems (ideally 10 ms or less)
- 30 ms or less on DSS (decision support system) type. Latencies here can vary significantly depending on the number of simultaneous queries being issued against the system. Sustained values of more than this when the total throughput is less than expected should be investigated.
- 50ms or Greater indicates a serious I/O Bottleneck

Consider these in combination with what is normal for your particular system.

Network Counters

Network Interface: Bytes Sent/sec - This is how many bytes of data are sent to the NIC. This is a raw measure of throughput for the network interface. We are really measuring the information sent to the interface which is the lowest point we can measure. If you have multiple NIC, you will see multiple instances of this particular counter.

Network Interface: Bytes Received/sec - This, of course, is how many bytes you get from the NIC. This is a measure of the inbound traffic in measuring the bytes; NT isn't too particular at this level. So, no matter what the byte is, it is counted. This will include the framing bytes as opposed to just the data.

Network Interface: Bytes Total/sec - This is simply a combination of the other two counters. This will tell you overall how much information is going in and out of the interface. Typically, you can use this to get a general feel, but will want to look at the Bytes Sent/sec and the Bytes Received/sec for a more exact detail of the type of traffic.

- High average network utilization – more than 50%
- Very high average network utilization – more than 80%

Network Interface\Output Queue Length\ (NIC Instance) - is the number of packets in queue waiting to be sent. If there is a sustained average of more than two packets in queue, you should be looking to resolve a network bottleneck.

- **High Network I/O** – more than 1 thread waiting on network I/O (If the output queue length is greater than 1. If so, this system's network is nearing capacity. Consider analyzing network traffic to determine why network I/O is nearing capacity such as chatty* network services and/or large data transfers)
- **Very high network I/O** – more than 2 threads waiting on network I/O (if the output queue length is greater than 2. If so, this system's network is over capacity. Consider analyzing network traffic to determine why network I/O is nearing capacity such as *chatty* network services and/or large data transfers)

The following are some possible causes of high queue utilisation.

- A network interface card cannot support the maximum bandwidth supported by your network infrastructure.
- There is too much load on the server.
- Viruses or SPAM are causing sudden elevated network usage patterns.

Argent SQL Monitoring - SQL Performance Counters

SQL Baseline Performance Counters

SQLServer: Access Methods - Full Scans/sec - Defined as the number of unrestricted full scans. These can either be base table or full index scans. Value greater than 1 or 2 indicates that we are having table / Index page scans.

SQLServer: Buffer Manager – Buffer Cache Hit Ratio - Defined 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).

Note: The Buffer Cache Hit Ratio counter is application specific; however, a rate of 90 percent or higher is desirable. Add more memory until the value is consistently greater than 90 percent, indicating that more than 90 percent of all requests for data were satisfied from the data cache.

SQLServer: Databases Application Database - Transactions/sec (use your application database) - Defined as the number of transactions started for the database. This number is depended on your application baseline and to help you troubleshoot issues.

Note: For example, if you normally show 120 transactions per second as your baseline and you come to work one Monday and see your server at 5,000 transactions per second, you will want to question the activity on your server.

SQLServer: Latches – Average Latch Wait Time - the average latch wait time (in milliseconds) for latch requests that had to wait. If this number is high, your server may be facing contention for its resources.

SQLServer: Locks – Average Wait Time - the average amount of wait time (milliseconds) for each lock request that resulted in a wait.

SQLServer: Locks – Lock Waits/sec - the number of lock requests that could not be satisfied immediately and required the caller to wait before being granted the lock.

SQLServer: Locks - Number of Deadlocks/sec - the number of lock requests that resulted in a deadlock. If you see anything above 0, your users and applications will experience problems. Their queries will abort and the applications may fail.

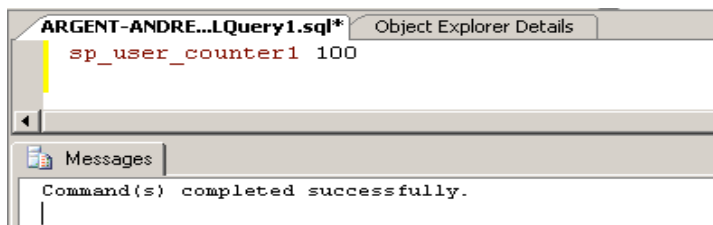
SQLServer: Memory Manager - Memory Grants Pending - the current number of processes waiting for a workspace memory grant.

Process: Working set – sqlservr - the amount of memory used by the default SQL Server process. If this number is consistently below the amount of memory SQL Server is configured to use (set by the **min server memory** and **max server memory** server options), SQL Server is configured for more memory than it needs. Otherwise, fix the size of the working set using the **set working set size server** option.

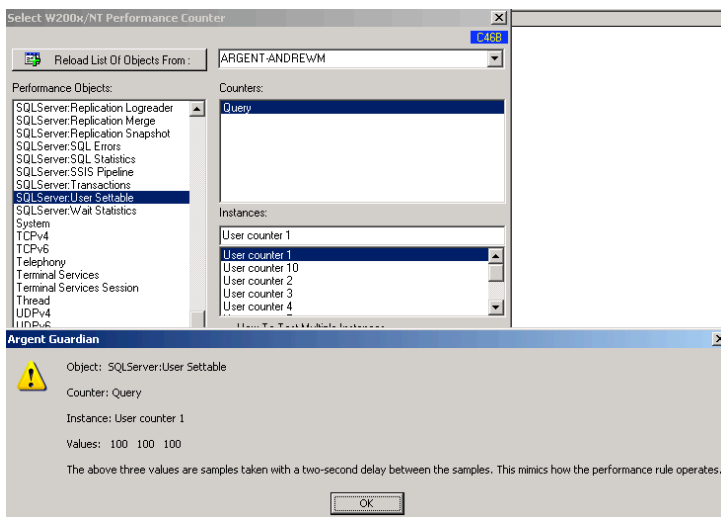
SQLServer: SQL Statistics: Batch Requests/Sec - This counter measures the number of batch requests that SQL Server receives per second, and generally follows in step to how busy your server's CPUs are. Generally speaking, over 1000 batch requests per second indicates a very busy SQL Server, and could mean that if you are not already experiencing a CPU bottleneck, that you may very well soon. Of course, this is a relative number, and the bigger your hardware, the more batch requests per second SQL Server can handle.

SQLServer: User Settable – Query – this is an application-specific counter. The value displayed by this counter is set by your application. To set this value, your application needs to call **sp_user_counter1** and provide a numeric value.

Determine an inexpensive way to check the health of your application with a single SQL statement (perhaps count the number of items that were sold in the last hour). The health check must not degrade the performance of your application in any way.



Write a stored procedure that first checks the health of your application, putting the result into a variable. The stored procedure should then call **sp_user_counter1** and provide that health check variable as the input parameter. Set up a scheduled job that runs this stored procedure every 15 minutes.



SQL Data File Space

SQL DB Space Rules – this is run against your application database. This rule tracks the percentage of space used within the database data file.

Rule Is Broken If Used Database Space Is More Than

Check SQL Database

☐ Default (Specified In Licensed Server Manager)

☒ This Specific Database (Wildcard '*' And '?' Are Allowed)

☐ The Whole SQL Server

SQL DB Space Rules – this is run against your application database. This rule tracks the bytes (KB, MB or GB) of space used within the database data file.

Rule Is Broken If Used Database Space Is More Than

Check SQL Database

☐ Default (Specified In Licensed Server Manager)

☒ This Specific Database (Wildcard '*' And '?' Are Allowed)

☐ The Whole SQL Server

Note: Both of these Argent SQL Rules can be used against the Transaction Log file also.

The following Performance counters provide the exact disk space used by the Data and Log data files.

- **SQLServer: Databases – Log File(s) Size (KB)** - (use your application database) this provides the exact size of the Log File on Disk.
- **SQLServer: Databases – Data File(s) Size (KB)** - (use your application database) this provides the exact size of the Data File on Disk.

SQL Concurrent Users Rule

SQL Concurrent User Rule - the number of user connections to the system. Dramatic shifts in this value should be researched.

Rule Is Broken If Concurrent User Connections Are More Than

Check SQL Database

☐ Default (Specified In Licensed Server Manager)

☒ This Specific Database (Wildcard '*' And '?' Are Allowed)

☐ The Whole SQL Server

SQL Lock Rules

SQL Lock Rule - the number of SQL Server locks - resources using different lock modes that determine how the resources can be accessed by concurrent transactions.

Rule Is Broken If System Has Locks More Than

Check SQL Database

☐ Default (Specified In Licensed Server Manager)

☒ This Specific Database (Wildcard '*' And '?' Are Allowed)

☐ The Whole SQL Server

☒ Get Exclusive Locks

Shared locks are used for operations that do not change or update data, such as a SELECT statement.

Update locks are used when SQL Server intends to modify a page, and later promotes the update page lock to an exclusive page lock before actually making the changes.

Exclusive locks are used for the data modification operations, such as UPDATE, INSERT, or DELETE.

SQL Query Rule

These Rules execute a single customizable SQL statement, stored procedure, or a T-SQL block on a specified SQL Server instance/database. In other words, the Argent Query Rules are essentially a product-within-a-product - you can use these Rules to do whatever you want to monitor whatever you want...

The execution of configured SQL is expected to return a recordset. It is checked according to configured "Check Option":

- Check all rows (i.e. all rows must satisfy the condition to pass the check) - further described below
- Rule is not broken if any row satisfies the condition - further described below
- Rule is not broken if any row is returned
- Rule is broken if any row is returned

With the first two check options, additional configuration is necessary. This is done in the area marked below:

Multiple conditions can be specified, optionally combined with "OR" clause. If there is no "OR" specified, an "AND" is implied.

The returned data can be saved to the Argent Predictor. To configure this option, use the Argent Predictor tab. The Argent Predictor is Argent's integrated trend analysis and capacity planning tool.

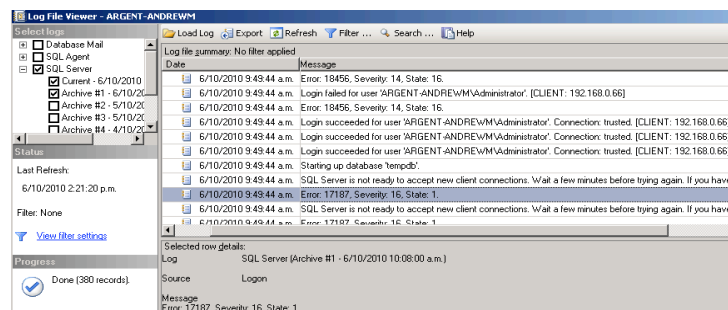
The screenshot shows a configuration window for the Argent Predictor. It includes several options and input fields:

- ☒ Save Performance Data Of Numeric Column: 2
- Object Name: Processes
- Counter Name: Blocking Processes
- Instance Name: Column 2
- ☐ Set Snap Time Using Data Of Date Column: 1
(Leave it unchecked to use query execution time as the snap time)
- ☐ Set Machine Name Using Data Of Text Column: 1
(Leave it unchecked to use SQL server name as the machine name)
- ☐ Only Save The First Row Of Query Results

Argent SQL Monitoring – Log Files

SQL Server Error Log

Another critical component of your ongoing monitoring efforts should be the SQL Server Log, also referred to as the SQL Error Log (which is found in Enterprise Manager). The SQL Server Log is an excellent resource for mining information about the health of your application.



The SQL Server Log is written to from the moment of start up, until the service is terminated. And endless arcane messages are written to it over that time span.

Error Severity Levels

The severity levels provide the best means of creating a general-purpose alerting system, supplemented by special alerts for common problems, such as TempDB running out of space.

- **Severity level 10 - 16** - are generally generated through mistakes by users, problems in the TSQL scripts and stored procedures executed by users. A number of programming errors and input problems can cause this sort of error.
- **Severity levels from 17 through 19** - require the attention of the DBA, who can then get more information by executing DBCC CHECKDB (database) to find out more about the extent of the damage.

- Severity level 17 (Insufficient Resources)
- Severity level 18 (Nonfatal Internal Error Detected)

Are generated by resource or system errors; the user's session is not interrupted.

- Severity level 19 (SQL Server Error in Resource) will stop the current batch.

Severity levels from 20 through 25 - these are considered fatal errors and terminate the client connection. Errors messages in this range may affect all of the processes in the database, and may indicate that a database or object is damaged. These errors are fatal to the process running at the time. The process will record diagnostic information, and then terminate.

Severity level 20 (SQL Server Fatal Error in Current Process)

Severity level 21 (SQL Server Fatal Error in Database dbid Processes)

Severity level 22 (SQL Server Fatal Error Table Integrity Suspect),

Severity level 23 (SQL Server Fatal Error: Database Integrity Suspect),

Severity level 24 (Hardware Error)

Severity level 25 Indicate system problems

The SQL Server Log should be checked for errors that have a severity level of 19–25.

The monitoring of the SQL error log can be achieved using the Argent Data Consolidator and an ASCII File Rule with the following configuration

BASIC Tab

Log File Path: C:\Program Files\Microsoft SQL Server\MSSQL\LOG\errorlog

Bias Date/Time Variables Used In File Path By: + 0 Hours

☐ Use As Regular Expression

Scanning Option: The Latest File Only

Time Stamp Format: yyyy-MM-dd HH:mm:ss.nn [Verify And Explain](#)

☐ Convert UTC Time To Local Time

Log Parsing Specification: [Load Sample Data](#)

```
# 6 severity
# 7 description

Delimiter = " "
IgnoreDuplicateDelimiter = Yes
Field("time") = (0, 0)
Field("machine") = (1, 1)
Field("description") = (2, 9999)
```

Time	Machine	Description
2010-10-06 09:49:44.55	Logon	Login succeeded for user 'ARGENT-ANDREW.M\Adm
2010-10-06 09:49:44.55	Logon	Login succeeded for user 'ARGENT-ANDREW.M\Adm
2010-10-06 09:49:44.55	Logon	Login succeeded for user 'ARGENT-ANDREW.M\Adm
2010-10-06 09:49:44.71	Logon	Error: 18456, Severity: 14, State: 16.
2010-10-06 09:49:44.71	Logon	Login failed for user 'ARGENT-ANDREW.M\Adm

Point at the location of the SQL error log file

Alert TAB (This test uses Numeric Greater than or Equal to 14)

☒ Post Event To Argent Console If Following Criteria Are Met

Optional Criteria:

description Contains String "Severity: " (Regular Expression) (Numeric Greater than or Equal 14)

A regular expression is used to check the value following the keyword Severity.

Note: The SQL Server Log should be checked for errors that have a severity level of 19–25.

When the SQL Error log is parse thru a production relator and a match is found the results as shown below can be found in the Argent Console when this test matches.

Event Detail	Correlated Argent Predictor Chart
Event time: 6 Oct 2010 09:49 Time recorded: 6 Oct 2010 14:45 Event from the node ARGENT-ANDREW.M answered: No Alert was fired successfully at 14:45:33 on 6 Oct 2010 System alarm alert ALARM_DEMO is executed successfully. (REL_DEMO_TEST_1 / LOG_SQL_DIAGNOSTICS_LOG) Memo note regarding the status and resolution of the event: None. Event Description: Found on log 'errorlog.1 (1069168623, 291891532)' of server ARGENT-ANDREW.M at following times: Wed Oct 06 09:49:44 Wed Oct 06 09:49:44 2010-10-06 09:49:44:37 Logon Error: 17187, Severity: 16, State: 1.	

Windows Application Event Log

The Event Viewer allows users to monitor events recorded in the Application log. These logs are separate from the SQL Server Log and SQL Agent Log, and thus provide additional information when it comes to your SQL application.

SQL Server messages can be identified as those messages with a source of "MSSQLSERVER" or "SQLSERVERAGENT."

This can be achieved using the Argent Data Consolidator Windows Event Log Rule see example below.

Event Log

☐ System Log
 ☐ Directory Service Log

☒ Application Log
 ☐ DNS Server Log

☐ Security Log
 ☐ File Replication Service

Custom Event Log Names (Separated By Commas):

Event Severity

☒ Error
 ☐ Audit Success

☒ Warning
 ☐ Audit Failure

☐ Informational

☒ Post Event To Argent Console If Following Criteria Are Met

Display Options

☐ No Filter And All Records Alerted
 ☒ Basic
 ☐ Advanced

Optional Event Log Criteria (1):

Include Events Contains String "MSSQLSERVER" From Event Fields: Source OR

Include Events Contains String "SQLSERVERAGENT" From Event Fields: Source

These events can be saved into the database for future reporting and or for alerting on matched criteria.

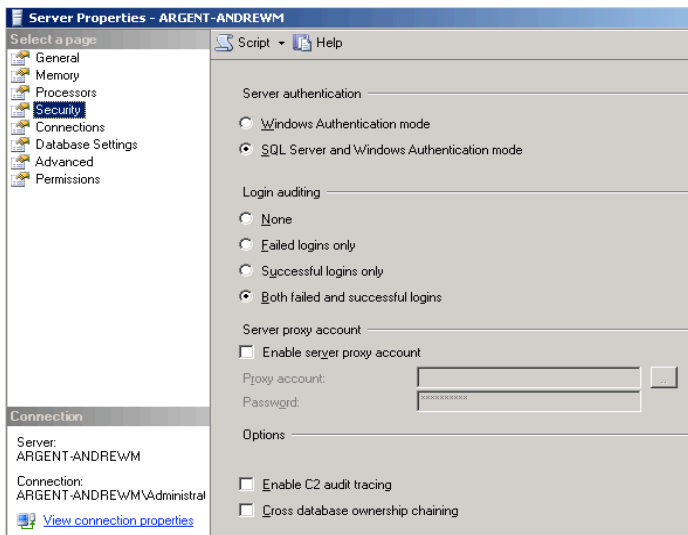
Security - Connection Auditing

MS SQL Server's first form of auditing is a subsystem that can be used to record failed and successful login attempts on the server. Recording connection attempts is useful in being able to discover:

- Who is attempting to connect to the database?
- When an attack is taking place
- If an attack was successful

This form of auditing has little negative side effects, as the amount of activity being audited is minimal. Auditing connection attempts does not result in a significant performance impact on the database, and rarely creates an excessive amount of data written to the log.

Because of the importance of knowing what logins are connecting to the database, coupled with the minimal impact recording a login causes, it is rarely a bad decision to audit connection attempts with this method.



The possible settings for login auditing are:

None - logs no auditing information

Success - causes only successful logins to be logged

Failure - causes only failed logins to be logged












All - causes successful and failed logins to be logged

The auditing information is written to the SQL Server error logs and to the Windows event log. To enable auditing of logins, perform the following actions:

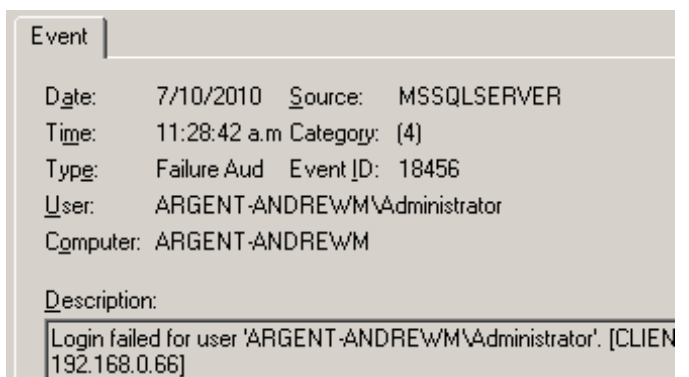
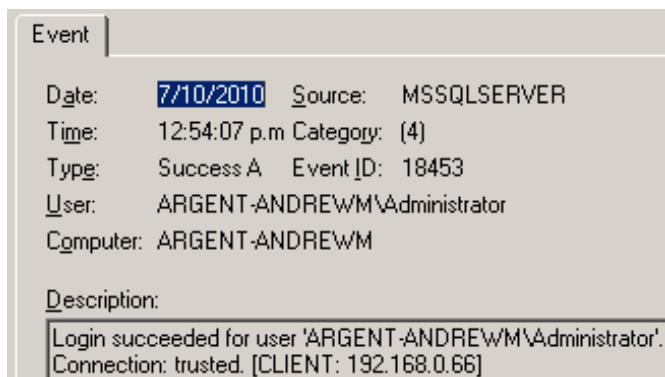
- 1) Open Enterprise Manager and connect to the database.
- 2) Click the right-mouse button on the instance and select Properties from the popup menu.
- 3) Open the 'Security' tab.
- 4) Under 'Audit Level' choose appropriate level

Every user who logs on to the SQL Server Instance is now logged as per the Login Auditing setting.

This SQL Auditing information is logged to the Windows Application Log as shown below.

	Information	6/10/2010	1:50:54 p.m.	MSSQLSERVER	(2)	17137	N/A
	Failure Audit	6/10/2010	1:50:54 p.m.	MSSQLSERVER	(4)	18456	Administrator
	Success Audit	6/10/2010	1:50:54 p.m.	MSSQLSERVER	(4)	18453	Administrator
	Information	6/10/2010	1:50:46 p.m.	MSSQLSERVER	(2)	17137	N/A
	Information	6/10/2010	1:50:44 p.m.	MSSQLSERVER	(2)	17137	N/A
	Information	6/10/2010	1:50:41 p.m.	MSSQLSERVER	(2)	17137	N/A
	Information	6/10/2010	1:50:37 p.m.	MSSQLSERVER	(2)	17137	N/A
	Information	6/10/2010	1:50:37 p.m.	MSSQLSERVER	(2)	17137	N/A
	Information	6/10/2010	1:50:32 p.m.	MSSQLSERVER	(2)	17137	N/A
	Failure Audit	6/10/2010	1:50:28 p.m.	MSSQLSERVER	(4)	18456	Administrator
	Success Audit	6/10/2010	1:50:28 p.m.	MSSQLSERVER	(4)	18453	Administrator

The following images show Success Audit and Failure Audit event information



This information can be captured using the Argent Data Consolidator for alerting and /or reporting purposes.

If you are concerned about SQL Server security and want to ensure that individuals with expired login accounts or bogus credentials are kept out of your SQL Server, you can also monitor the Security Log for failed login attempts at the Domain Level. This information can be useful for spotting patterns of fraudulent entry. It can also alert you to applications that are using bad credentials and need to be properly configured.

C2 Auditing

Microsoft SQL Server also provides an option to audit successful and failed attempts to access statements and objects. This option was designed to meet the standards of the C2 Security Evaluation criteria. This feature provides a valuable means of monitoring for system misuse. By default, it is not enabled.

When C2 auditing is enabled, SQL Server will track C2 audit events and record them to a file in the \mssql\data directory. A drawback of this approach is that the C2 auditing feature will cause the database to stop when the server is unable to write to the audit file for any reason. This is in accordance with the requirements for C2 security. Care should be taken when enabling C2 auditing since excessive use of audit counters could have a significant performance impact on the server. C2 auditing is generally considered ineffective because the amount of information being record is simply overwhelming. Enabling this feature can result in running out of disk space quickly and can harm performance on an already taxed system.

Argent SQL Monitoring – Job Status

Monitoring Maintenance Jobs

It is likely that your application includes one or more maintenance jobs, such as jobs for backing up the database, processing files, or clearing out staging tables. It is also very likely that the health of these jobs directly impacts the health of your application.

The screenshot shows a configuration window for SQL Job Rules. At the top, there is a text box labeled "SQL Job Names:" containing an asterisk (*). Below it, a note states "(Wildcard '*' And '?' Are Allowed)". Underneath, a label "Rule Is Broken If Following Condition Is Met By" is followed by a dropdown menu currently set to "Any Job". A section titled "Last Run Status Of SQL Job Is One Of Selected" contains five checkboxes: "Failed" (checked), "Succeeded" (unchecked), "Cancelled" (unchecked), "In Progress" (unchecked), and "Unknown" (unchecked). At the bottom, there is a checkbox labeled "Ignore Job With Last Run Time" which is checked, followed by a time selection area with a dropdown set to "Before", a numeric spinner set to "24", and a dropdown set to "Hours".

Argent SQL Monitor provides an excellent way to monitor the health of your maintenance jobs. Associated with each scheduled job is a history of job execution. This history should be checked on a regular basis by using the SQL Job Rules as shown above.

Argent SQL Monitoring – Availability

Monitoring Maintenance Jobs

Services - This can be done by monitoring related services such as SQL Server and SQL Agent fail using Argent Guardian Service Rules and can automatically restart them if required.

Service(s) To Monitor	Add		Delete		Manual	
Rule Will Fail If Service Or Any Service In Macro Is	NOT Running	Running	NOT Responding	Not Installed		
SQL Server (MSSQLSERVER)	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
SQL Server Agent (MSSQLSERVER)	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>		

SQL Logon - This can be done by logging into the SQL Server Instance using the Argent SQL Monitor connectivity rules.

How To Determine If The Nodes In The Monitoring Group Are Running:

☐ Use Windows NetRemoteTOD API
☐ Check Cluster (Node, Group, Network, Network Interface Or Resource)
 UNIX And Other Non-NT/W200X
☐ Open This Existing File On The Node
☐ TCP/IP Ping
☐ Scan Specific TCP/IP Port Number
☒ Logon To SQL Server Of Instance

Server - This can be done by using the connectivity rules shown above and typically to test if a server can serve user connections we would use either the Windows NetRemoteTOD API or Open a file on the Node.

Server Down Time Report

This is a sample report installed by setup program of Argent Extended Technology.
 It is the report for the list of time period when server is offline.

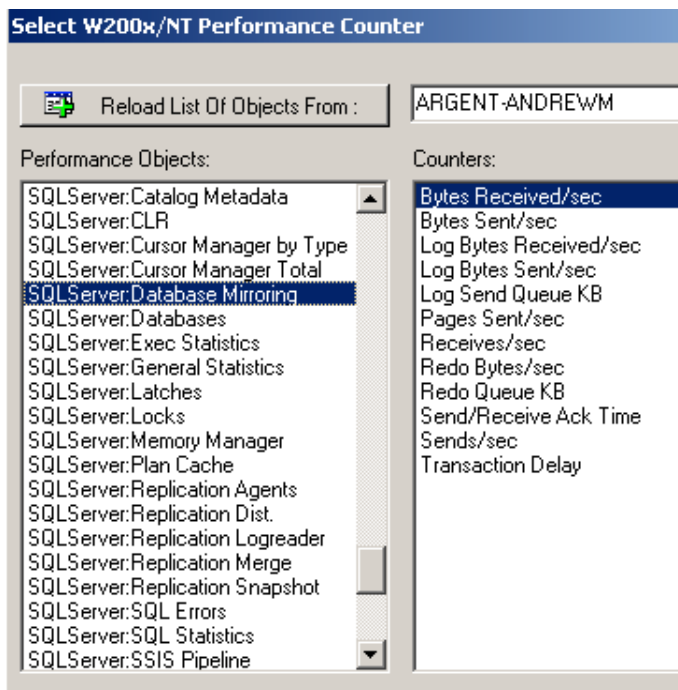
Device	Detected Down	Detected Up	Total OutageTime (hh:mm:ss)
ARGENT-ANDREWM			
Total Up Time:100.00%		Total DownTime:00:00:00	
ARGENT-ANDREWM (MSSQLServer)			
Total Up Time:100.00%		Total DownTime:00:00:00	
ARGENT-ANDREWM (SQLServerAgent)			
Thu, 7 Oct 2010 02:08:19 PM		Thu, 7 Oct 2010 02:14:04 PM	00:05:45
Total Up Time:100.00%		Total DownTime:00:05:45	

All of the above availability metrics can be reported against in a Server down Time report as shown above.

Argent SQL Monitoring – Mirroring

Monitoring Database Mirroring

To monitor the performance of database mirroring, SQL Server provides a System Monitor performance object (**SQLServer:Database Mirroring**) on each partner (principal and mirror). Following shows the counters that are available to monitor:



The following Stored procedure can be used to provide detailed information on the status on mirroring for a particular database.

***sp_dbmmonitorresults database_name, rows_
to_return, update_status***

database_name – Specifies the Mirrored database name to monitor

rows_to_return - Specifies the quantity of rows returned:

- 0 = LAST ROW
- 1 = ROWS LAST TWO HOURS
- 2 = ROWS LAST FOUR HOURS
- 3 = ROWS LAST EIGHT HOURS
- 4 = ROWS LAST DAY
- 5 = ROWS LAST TWO DAYS
- 6 = LAST 100 ROWS
- 7 = LAST 500 ROWS
- 8 = LAST 1,000 ROWS
- 9 = LAST 1,000,000 ROWS

update_status - Specifies that before returning results the procedure:

0 = Does not update the status for the database. The results are computed using just the last two rows, the age of which depends on when the status table was refreshed.\

1 = Updates the status for the database by calling **sp_dbmmonitorupdate** before computing the results. However, if the status table has been updated within the previous 15 seconds, or the user is not a member of the **sysadmin** fixed server role, **sp_dbmmonitorresults** runs without updating the status.

Argent SQL Monitor has the capability of utilising a SQL Statement or Stored Procedure within a SQL Query Rule and checks the data in the return columns

The screenshot shows the configuration window for a SQL Query Rule. The 'Check SQL Database' section has 'This Specific Database' set to 'MSDB'. The 'Run Following SQL Query Or Stored Procedure' section has 'SQL Statement' selected. The query text is 'sp_dbmmonitorresults argentxt, 0, 1'. The 'Instance Name' is empty. 'SQL Timeout' is 30 seconds. 'Check Option' is 'Check ALL Rows'. 'Comparison Column' is 0. The 'Rule Is NOT Broken If The Following Conditions Are True:' section contains the condition 'Numeric(\$3) Equal 4'. Red arrows point from text annotations to these specific fields.

Check SQL Database

☐ Default (Specified In Licensed Server Manager)

☒ This Specific Database (Wildcard '*' And '?' Are Allowed)

Run Following SQL Query Or Stored Procedure

☒ SQL Statement ☐ Stored Procedure ☐ Transact-SQL

`sp_dbmmonitorresults argentxt, 0, 1`

Instance Name:

SQL Timeout: Seconds (Value 0 for default timeout)

Check Option:

Comparison Column: (Used When Firing Events Separately)

Rule Is NOT Broken If The Following Conditions Are True:

`Numeric($3) Equal 4`

If Column 3 is Broken

This query must be run against msdb database

The stored procedure and variables passed to it

Argent will check all rows

equal to Numeric Value 4 then the rule is NOT

This will check that the Mirrored Database is OK
(4 = SYNCHRONIZED)

Argent SQL Monitor has the capability of utilising a SQL Statement or Stored Procedure within a SQL Query Rule and checks the data in the return columns

The screenshot shows the configuration window for a SQL Query Rule. It has two main sections: 'Check SQL Database' and 'Run Following SQL Query Or Stored Procedure'.

Check SQL Database: The 'Default (Specified In Licensed Server Manager)' radio button is unselected, and the 'This Specific Database' radio button is selected. The text box next to it contains 'MSDB'. A note in parentheses says '(Wildcard '*' And '?' Are Allowed)'.

Run Following SQL Query Or Stored Procedure: The 'SQL Statement' radio button is selected, while 'Stored Procedure' and 'Transact-SQL' are unselected.

SQL Statement: The text area contains the query: `sp_dbmmonitorresults argentxt, 0, 1|`

Instance Name: An empty text box.

SQL Timeout: A spinner box set to '30' with the label 'Seconds (Value 0 for default timeout)'.

Check Option: A dropdown menu set to 'Check ALL Rows'.

Comparison Column: A spinner box set to '0' with the label '(Used When Firing Events Separately)'.

Rule Is NOT Broken If The Following Conditions Are True: A text area containing the condition: `Numeric($4) Equal 1`

This will check that the Witness is OK (1 = CONNECTED)

Database Mirroring Status Table

Database mirroring status is stored in an internal, undocumented database mirroring status table in the **msdb** database. This status table is automatically created the first time the mirroring status is updated.

The status table may be updated either automatically or manually by a system administrator, with a minimum update interval of 15 seconds. The 15 second minimum prevents server instances from being overloaded with status requests.

The first time **sp_dbmmonitorupdate** runs, it creates the **database mirroring status** table and the **dbm_monitor** fixed database role in the **msdb** database. **sp_dbmmonitorupdate** usually updates the mirroring status by inserting a new row into the status table for every mirrored database on the server instance.

The following table shows the column names and the description of the data they contain

Column name	Description
database_name	Name of a mirrored database.
role	Current mirroring role of the server instance: 1 = Principal 2 = Mirror
mirroring_state	State of the database: 0 = Suspended 1 = Disconnected 2 = Synchronizing 3 = Pending Failover 4 = Synchronized
witness_status	Connection status of the witness in the database mirroring session of the database, can be: 0 = Unknown 1 = Connected 2 = Disconnected
log_generation_rate	Amount of log generated since preceding update of the mirroring status of this database in kilobytes/sec.
unsent_log	Size of the unsent log in the send queue on the principal in kilobytes.
send_rate	Send rate of log from the principal to the mirror in kilobytes/sec.
unrestored_log	Size of the redo queue on the mirror in kilobytes.
recovery_rate	Redo rate on the mirror in kilobytes/sec.
transaction_delay	Total delay for all transactions in milliseconds.
transactions_per_sec	Number of transactions that are occurring per second on the principal server instance.
average_delay	Average delay on the principal server instance for each transaction because of database mirroring. In high-performance mode (that is, when the SAFETY property is set to OFF), this value is generally 0.
time_recorded	Time at which the row was recorded by the database mirroring monitor. This is the system clock time of the principal.
time_behind	Approximate system-clock time of the principal to which the mirror database is currently caught up. This value is meaningful only on the principal server instance.
local_time	System clock time on the local server instance when this row was updated.

SQL Server Performance – What are DMVs and DMFs?

Dynamic Management Views and Dynamic Management Functions

Microsoft introduced dynamic management views (DMV) and functions (DMF) with SQL Server 2005. DMVs and DMFs are a mechanism to allow you to look at the internal workings of SQL Server using TSQL. They allow you an easy method for monitoring what SQL Server is doing and how it is performing. They replace the need to query the system tables or using other awkward methods of retrieving system information that you had to use with SQL Server 2000.

- Some DMV's/DMF's are used to retrieve server wide information and are said to have server scope.
- Other DMV's/DMF's are used to obtain database information and therefore have database scope.

All DMV's/DMF's are stored in the master database, under the sys schema. Any object in the sys schema whose name starts with "dm_" is known as a dynamic object. The DMV/DMF's have been organized into the following different groups:

- Common Language Runtime related
- Database Mirroring related
- Execution related
- Full-Text Search related
- Index related
- I/O related
- Query Notifications related
- Replication related
- Service Broker related
- SQL Server Operation system
- Transaction related

DMV return server metric information that can be used to monitor the health of an instance, diagnose problems, and performance tune. They provide insight into the following areas.

- Views on memory structures in the Engine
- Monitor activity inside the Engine
- Takes the black box approach out of tuning
- DMV expose server state information spanning sessions, transactions, and requests.
- DMVs and DMFs reflect what's going on inside the server process or across sessions in the server.
- Use for diagnostics, memory and process tuning, and monitoring across all sessions in the server.

Note: Books Online (BOL) documents what each DMV/DMF does, and what parameters need to be passed to the DMF's, as well as the columns returned when using these objects.

DMVs are organized into five general categories according to the area on which they report:

- **sys.dm_exec_*** – provide information about execution of .NET CLR Modules and connections. All contained here are a number of views available to help with issues related to execution of queries.
- **sys.dm_os_*** – report on memory, locks, and execution scheduling.
- **sys.dm_trans_*** – provide insight into transactions and isolation.
- **sys.dm_io_*** – monitoring disk I/O
- **sys.dm_db_*** – provide database-level data.

Monitoring Connections (Example DMV Usage)

You can use the **sys.dm_exec_connections** view to retrieve information about the connections established to a specific SQL server and the details of each connection. In addition, the **sys.dm_exec_sessions** view is helpful when retrieving information about all active user connections and internal tasks.

The following query retrieves information on the current connections:

```
SELECT
    e.connection_id,
    s.session_id,
    s.login_name,
    s.last_request_end_time,
    s.cpu_time
FROM
    sys.dm_exec_sessions s
    INNER JOIN sys.dm_exec_connections e
    ON s.session_id = e.session_id
```

Results		Messages			
	connection_id	session_id	login_name	last_request_end_time	cpu_time
1	1640968D-5AD4-460F-8073-48947304FE07	51	ARGENT-ANDREW\M\Administrator	2010-10-08 14:16:38.957	1187
2	9C8189DA-4FFB-4685-BC1E-236A0DDFE6F0	52	ARGENT-ANDREW\M\Administrator	2010-10-08 14:19:07.903	32
3	95FCD616-5265-42B5-B473-9555839918F6	53	ARGENT-ANDREW\M\Administrator	2010-10-08 14:33:24.473	2187
4	A7D06CC5-E6F3-4866-97B4-B3AC395808DE	54	ARGENT-ANDREW\M\Administrator	2010-10-08 14:21:18.237	16
5	FFDE0888-9AE5-4A33-ADCA-C2894F51021A	55	ARGENT-ANDREW\M\Administrator	2010-10-08 14:22:02.193	15

Here is another example code found in Books Online. This code uses the **sys.dm_exec_query_stats** DMV and the **sys.dm_exec_sql_text** DMF to show the average CPU time for the top 5 query execution plans in cache.

```
SELECT TOP 5 total_worker_time/execution_count 'Avg CPU Time',
    SUBSTRING(st.text, (qs.statement_start_offset/2)+1,
        ((CASE qs.statement_end_offset
            WHEN -1 THEN DATALENGTH(st.text)
            ELSE qs.statement_end_offset
            END - qs.statement_start_offset)/2) + 1)statement_text
FROM sys.dm_exec_query_stats qs
CROSS APPLY sys.dm_exec_sql_text(qs.sql_handle) st
ORDER BY total_worker_time/execution_count DESC;
```

Results Messages		
	Avg CPU Time	statement_text
1	3062546	SELECT sp.name AS [Name], 'Server[@Name=' + quot...
2	217914	SELECT creation_time, last_execution_time ,total_phy...
3	201891	SELECT creation_time,last_execution_time,total_physic...
4	132561	SELECT creation_time,last_execution_time,total_physic...
5	108713	select TABLE_QUALIFIER = convert(sysname,...

These queries could be used with Argent SQL Monitor in a SQL Query Rule to save data to the database or to trigger a threshold being reached.

Appendix A – SQL Server Performance Objects

SQL Server object	Description
SQL Server: Access Methods	Searches through and measures allocation of SQL Server database objects (for example, the number of index searches or number of pages that are allocated to indexes and data).
SQL Server: Backup Device	Provides information about backup devices used by backup and restore operations, such as the throughput of the backup device.
SQL Server: Buffer Manager	Provides information about the memory buffers used by SQL Server, such as freememory and buffer cache hit ratio .
SQL Server: Buffer Partition Object	Provides information about how frequently SQL Server requests and accesses free pages.
SQL Server: Cache Manager	Provides information about the SQL Server cache used to store objects such as stored procedures, triggers, and query plans.
SQL Server: Databases	Provides information about a SQL Server database, such as the amount of free log space available or the number of active transactions in the database. There can be multiple instances of this object.
SQL Server: General Statistics	Provides information about general server-wide activity, such as the number of users who are connected to an instance of SQL Server.
SQL Server: Latches	Provides information about the latches on internal resources, such as database pages, that are used by SQL Server.
SQL Server: Locks	Provides information about the individual lock requests made by SQL Server, such as lock time-outs and deadlocks. There can be multiple instances of this object.
SQL Server: Memory Manager	Provides information about SQL Server memory usage, such as the total number of lock structures currently allocated.
SQL Server: Replication Agents	Provides information about the SQL Server replication agents currently running.
SQL Server: Replication Dist.	Measures the number of commands and transactions read from the distribution database and delivered to the Subscriber databases by the Distribution Agent.
SQL Server: Replication Logreader	Measures the number of commands and transactions read from the published databases and delivered to the distribution database by the Log Reader Agent.
SQL Server: Replication Merge	Provides information about SQL Server merge replication, such as errors generated or the number of replicated rows that are merged from the Subscriber to the Publisher.

SQL Server object	Description
SQL Server: Replication Snapshot	Provides information about SQL Server snapshot replication, such as the number of rows that are bulk copied from the publishing database.
SQL Server: SQL Statistics	Provides information about aspects of SQL queries, such as the number of batches of Transact-SQL statements received by SQL Server.
SQL Server: User Settable Object	Performs custom monitoring. Each counter can be a custom stored procedure or any Transact-SQL statement that returns a value to be monitored.

Note: ArgSoft Intellectual Property Holdings Limited has created this White Paper for informational purposes only. ArgSoft Intellectual Property Holdings Limited makes no warranties, express or implied, in this document. The information contained in this document is subject to change without notice. ArgSoft Intellectual Property Holdings Limited shall not be liable for any technical or editorial errors, or omissions contained in this document, nor for incidental, indirect or consequential damages resulting from the furnishing, performance, or use of the material contained in this document, or the document itself. All views expressed are opinions of ArgSoft Intellectual Property Holdings Limited. All trademarks are the property of their respective owners.