Monitoring ASP .Net Servers

eG Enterprise v6

Restricted Rights Legend

The information contained in this document is confidential and subject to change without notice. No part of this document may be reproduced or disclosed to others without the prior permission of eG Innovations Inc. eG Innovations Inc. makes no warranty of any kind with regard to the software and documentation, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose.

Trademarks

Microsoft Windows, Windows NT, Windows 2003, and Windows 2000 are either registered trademarks or trademarks of Microsoft Corporation in United States and/or other countries.

The names of actual companies and products mentioned herein may be the trademarks of their respective owners.

Copyright

Table of Contents

Table of Figures

Chapter 1

Monitoring ASP .Net Servers

ASP .Net is a programming framework built on the common language runtime (CLR) that can be used on a server to build powerful web applications, dynamic web sites, and mission-critical web services. To ensure the stability of these web services, the ASP .Net framework should perform without a glitch. This is why continuous monitoring of ASP .Net is important.

eG Enterprise has specially designed an *ASP .Net* monitoring model (see Figure 1.1) , which closely monitors the performance of the ASP .Net framework from its core worker processes, to the language (i.e., CLR) on which it has been built, to applications deployed on them, and accurately pin-points bottlenecks to optimal performance.

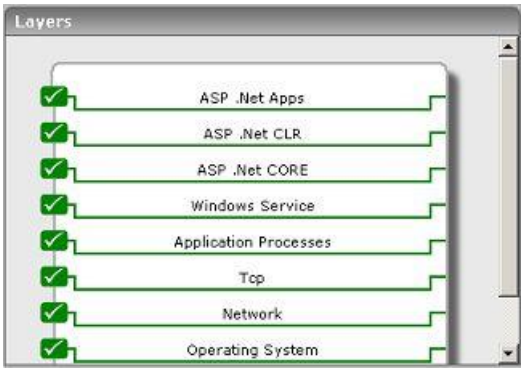

Figure 1.1: The layer model of an ASP .Net server

The sections to come will only discuss the top 3 layers of Figure 1.1, as the rest of the layers have already been extensively discussed in the *Monitoring Unix and Windows Servers* document.

1.1 The ASP .Net CORE Layer

The test mapped to this layer (see Figure 1.2) monitors the performance of the worker process of the ASP .Net server.

Figure 1.2: The tests associated with the ASP .Net CORE Layer

1.1.1 ASP .Net Workers Test

The AspNetWorkerTest reports statistics pertaining to the performance of the worker process of the ASP .Net server.

Purpose	Reports statistics pertaining to the performance of the worker process of the ASP .Net servers		
Target of the test	The ASP .Net server		
Agent deploying the test	An internal agent		
Configurable parameters for the test	1. **TEST PERIOD** - How often should the test be executed 2. **HOST** - The host for which the test is to be configured 3. **PORT** - The port at which the specified **HOST** listens		
Outputs of the test	One set of results for the ASP .Net server being monitored		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Application restarts: The number of application restarts.	Number	In a perfect world, the application domain will and should survive for the life of the process. Even if a single restart occurs, it is a cause for concern because proactive and reactive restarts cause automatic recycling of the worker process. Moreover, restarts warrant recreation of the application domain and recompilation of the pages, both of which consume a lot of time. To investigate the reasons for a restart, check the values set in the processModel configuration.
	Applications running: The number of applications currently running.	Number	

	Requests current: The number of requests currently handled by the ASP.NET ISAPI. This includes those that are queued , executing, or waiting to be written to the client.	Number	
	Request execution time: The number of seconds taken to execute the last request.	Number	In version 1.0 of the framework, the execution time begins when the worker process receives the request, and stop when the ASP.NET ISAPI sends HSE_REQ_DONE_WITH_SESSION to IIS. In version 1.1 of the framework, execution begins when the HttpContext for the request is created, and stop before the response is sent to IIS. The value of this measure should be stable. Any sudden change from the previous recorded values should be notified.
	Requests queued: The number of requests currently queued.	Number	When running on IIS 5.0, there is a queue between inetinfo and aspnet_wp, and there is one queue for each virtual directory. When running on IIS 6.0, there is a queue where requests are posted to the managed ThreadPool from native code, and a queue for each virtual directory. This counter includes requests in all queues. The queue between inetinfo and aspnet_wp is a named pipe through which the request is sent from one process to the other. The number of requests in this queue increases if there is a shortage of available I/O threads in the aspnet_wp process. On IIS 6.0 it increases when there are incoming requests and a shortage of worker threads.
	Requests rejected: The number of rejected requests	Number	Requests are rejected when one of the queue limits is exceeded. An excessive value of this measure hence indicates that the worker process is unable to process the requests due to overwhelming load or low memory in the processor.

	Requests wait time: The number of seconds that the most recent request spent waiting in the queue, or named pipe that exists between inetinfo and aspnet_wp. This does not include any time spent waiting in the application queues.	Secs	
	Worker processes running: The current number of aspnet_wp worker processes	Number	Every application executing on the .NET server corresponds to a worker process. Sometimes, during active or proactive recycling, a new worker process and the worker process that is being replaced may coexist. Under such circumstances, a single application might have multiple worker processes executing for it. Therefore, if the value of this measure is not the same as that of *Applications running*, then it calls for closer examination of the reasons behind the occurence.
	Worker process restarts: The number of aspnet_wp process restarts in the machine	Number	Process restarts are expensive and undesirable. The values of this metric are dependent upon the process model configuration settings, as well as unforeseen access violations, memory leaks, and deadlocks.

1.2 The ASP .Net CLR Layer

The tests associated with this layer (see Figure 1.3) monitor the following:

- Managed locks and threads
- Exceptions that occur in the CLR
- Garbage collection activity
- The locking activity
- the security system activity
- JIT compilation

Figure 1.3: The tests associated with the ASP .Net CLR layer

1.2.1 ASP Lock Threads Test

This test provides information about managed locks and threads that an application uses.

Purpose	Provides information about managed locks and threads that an application uses		
Target of the test	An ASP .Net server		
Agent deploying the test	An internal agent		
Configurable parameters for the test	1. **TEST PERIOD** - How often should the test be executed 2. **HOST** - The host for which the test is to be configured 3. **PORT** - The port at which the specified **HOST** listens		
Outputs of the test	One set of results for the ASP .Net server being monitored		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Current logical threads: The number of current managed thread objects in the application. This measure maintains the count of both running and stopped threads.	Number	

	Current physical threads: The number of native operating system threads created and owned by the common language runtime to act as underlying threads for managed thread objects. This measure does not include the threads used by the runtime in its internal operations.	Number	
	Current recognized threads: The number of threads that are currently recognized by the runtime. These threads are associated with a corresponding managed thread object.	Number	
	Contention rate: The rate at which threads in the runtime attempt to acquire a managed lock unsuccessfully.	Rate/Sec	
	Current queue length: The total number of threads that are currently waiting to acquire a managed lock in the application.	Number	

1.2.2 ASP .Net CLR ExceptionsTest

This test reports statistics related to the exceptions that occur in the CLR due to managed and unmanaged exceptions.

Purpose	Reports statistics related to the exceptions that occur in the CLR due to managed and unmanaged exceptions
Target of the test	An ASP .Net server
Agent deploying the test	An internal agent

Configurable parameters for the test	1. **TEST PERIOD** - How often should the test be executed 2. **HOST** - The host for which the test is to be configured 3. **PORT** - The port at which the specified **HOST** listens		
Outputs of the test	One set of results for every worker process on the ASP .Net server		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Clr exceptions: The total number of managed exceptions thrown per second.	Exceptions/Sec	Exceptions are very costly and can severely degrade your application performance. A high value of this measure is therefore an indicator of potential performance issues.

1.2.3 ASP .Net CLR GC Test

This test monitors the memory allocation activity of the ASP .Net server, in terms of heaps when objects are created and managed.

Purpose	Monitors the memory allocation activity of the ASP .Net server		
Target of the test	An ASP .Net server		
Agent deploying the test	An internal agent		
Configurable parameters for the test	1. **TEST PERIOD** - How often should the test be executed 2. **HOST** - The host for which the test is to be configured 3. **PORT** - The port at which the specified **HOST** listens		
Outputs of the test	One set of results for every worker process on the ASP .Net server		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Heap mem usage: The number of bytes committed by managed objects. This is the sum of the large object heap and the generation 0, 1, and 2 heaps.	MB	

	Gen 0 collections: The rate at which the generation 0 objects (youngest; most recently allocated) are garbage collected (Gen 0 GC) since the start of the application.	Collections/Sec	
	Gen 1 collections: The rate at which the generation 1 objects have been garbage collected since the start of the application. Objects that survive are promoted to generation 2.	Collections/Sec	
	Gen 2 collections: The number of seconds taken to execute the last request.	Number	The number of times generation 2 objects have been garbage collected since the start of the application. Generation 2 is the highest, thus objects that survive collection remain in generation 2. Gen 2 collections can be very expensive, especially if the size of the Gen 2 heap is huge.
	Time in gc: % Time in GC is the percentage of elapsed time that was spent in performing a garbage collection (GC) since the last GC cycle.	Percent	This measure is usually an indicator of the work done by the Garbage Collector on behalf of the application to collect and conserve memory. This measure is updated only at the end of every GC and the measure reflects the last observed value; its not an average.

1.2.4 ASP CLR Load Test

a. This test monitors the classes and assemblies loaded on to an ASP .Net application. A class is essentially the blueprint for an object. It contains the definition for how a particular object will be instantiated at runtime, such as the properties and methods that will be exposed publicly by the object and any internal storage structures.

b. Also known as Managed DLLs, assemblies are the fundamental unit of deployment for the .NET platform. The .NET Framework itself is made up of a number of assemblies, including mscorlib.dll, among others. The assembly boundary is also where versioning and security are applied. An assembly contains Intermediate Language generated by a specific language compiler, an assembly manifest (containing information about the assembly), type metadata, and resources.

Purpose	Monitors the classes and assemblies loaded on to an ASP .Net application
Target of the test	An ASP .Net server

Agent deploying the test	An internal agent		
Configurable parameters for the test	1. **TEST PERIOD** - How often should the test be executed 2. **HOST** - The host for which the test is to be configured 3. **PORT** - The port at which the specified **HOST** listens		
Outputs of the test	One set of results for every worker process on the ASP .Net server being monitored		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Classes loaded: Indicates the cumulative number of classes loaded in all assemblies since the start of this application.	Number	
	Current classes loaded: Indicates the current number of classes loaded in all Assemblies.	Number	An unusually high value may indicate a sudden increase in classes which loaded on to this .NET application.
	Rate of assemblies: The rate at which Assemblies were loaded across all AppDomains.	Assembles/Sec	If the Assembly is loaded as domain-neutral from multiple AppDomains then this counter is incremented once only. Assemblies can be loaded as domain-neutral when their code can be shared by all AppDomains or they can be loaded as domain-specific when their code is private to the AppDomain. This counter is not an average over time; it displays the difference between the values observed in the last two samples divided by the duration of the sample interval.
	Rate of classes loaded: This rate at which the classes loaded in all Assemblies.	Classes/Sec	This counter is not an average over time; it displays the difference between the values observed in the last two samples divided by the duration of the sample interval.
	Rate of load failures: The rate of load failures on the application.	Failures/Sec	This counter is not an average over time; it displays the difference between the values observed in the last two samples divided by the duration of the sample interval. These load failures could be due to many reasons like inadequate security or illegal format.
	Current appdomains: The number of AppDomains currently loaded in this application.	Number	AppDomains (application domains) provide a secure and versatile unit of processing that the CLR can use to provide isolation between applications running in the same process.

	Current assemblies: The number of assemblies currently loaded across all AppDomains in this application.	Number	If the Assembly is loaded as domain-neutral from multiple AppDomains then this counter is incremented once only. Assemblies can be loaded as domain-neutral when their code can be shared by all AppDomains or they can be loaded as domain-specific when their code is private to the AppDomain.
	Loader heap size: The size of the memory committed by the class loader across all AppDomains.	MB	Committed memory is the physical memory for which space has been reserved on the disk paging file.
	Load failures: The number of classes that have failed to load during the last measurement period,	Number	These load failures could be due to many reasons like inadequate security or illegal format.
	Appdomains loaded: The number of AppDomains loaded during the last measurement period.	Number	
	Num assemblies: The number of assemblies loaded during the last measurement period.	Number	An assembly in ASP.NET is a collection of single-file or multiple files. The assembly that has more than one file contains either a dynamic link library (DLL) or an EXE file. The assembly also contains metadata that is known as assembly manifest. The assembly manifest contains data about the versioning requirements of the assembly, author name of the assembly, the security requirements that the assembly requires to run, and the various files that form part of the assembly. The biggest advantage of using ASP.NET Assemblies is that developers can create applications without interfering with other applications on the system.

1.2.5 Clr Lock Threads Test

This test monitors the thread locking activity on the ASP .Net server.

Purpose	Monitors the thread locking activity on the ASP .Net server
Target of the test	An ASP .Net server
Agent	An internal agent

deploying the test			
Configurable parameters for the test	1. **TEST PERIOD** - How often should the test be executed 2. **HOST** - The host for which the test is to be configured 3. **PORT** - The port at which the specified **HOST** listens		
Outputs of the test	One set of results for the ASP .Net server being monitored		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Queue length rate: Indicates the rate at which threads are waiting to acquire some lock in the application.	Threads/Sec	
	Recognized threads rate: Indicates the number of threads per second that have been recognized by the CLR.	Threads/Sec	The recognized threads have a corresponding .NET thread object associated with them. These threads are not created by the CLR; they are created outside the CLR but have since run inside the CLR at least once. Only unique threads are tracked; threads with the same thread ID re-entering the CLR or recreated after thread exit are not counted twice.
	Queue length peak: Indicates the total number of threads that waited to acquire some managed lock during the last measurement period.	Number	A high turnover rate indicates that items are being quickly added and removed, which can be expensive.
	Recognized threads: Indicates the total number of threads that have been recognized by the CLR during the last measurement period.	Number	The recognized threads have a corresponding .NET thread object associated with them. These threads are not created by the CLR; they are created outside the CLR but have since run inside the CLR at least once. Only unique threads are tracked; threads with the same thread ID re-entering the CLR or recreated after thread exit are not counted twice.
	Contention threads: Indicates the total number of times threads in the CLR have attempted to acquire a managed lock unsuccessfully.	Number	Managed locks can be acquired in many ways; by the lock statement in C# or by calling System.Monitor.Enter or by using MethodImplOptions.Synchronized custom attribute.

1.2.6 Clr Security Test

This test monitors the security system activity of the ASP .Net server.

Purpose	Monitors the security system activity of the ASP .Net server		
Target of the test	An ASP .Net server		
Agent deploying the test	An internal agent		
Configurable parameters for the test	1. **TEST PERIOD** - How often should the test be executed 2. **HOST** - The host for which the test is to be configured 3. **PORT** - The port at which the specified **HOST** listens		
Outputs of the test	One set of results for the ASP .Net server being monitored		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Time in runtime checks: Indicates the percentage of elapsed time spent in performing runtime Code Access Security (CAS) checks during the last measurement period.	Percent	If this counter is high, revisit what is being checked and how often. The application may be executing unnecessary stack walk depths. Another cause for a high percentage of time spent in runtime checks could be numerous linktime checks.
	Stack walk depth: Indicates the depth of the stack during that last measurement period.	Number	
	Link time checks: Indicates the total number of linktime Code Access Security (CAS) checks during the last measurement period.	Number	The value displayed is not indicative of serious performance issues, but it is indicative of the health of the security system activity.
	Runtime checks: Indicates the total number of runtime CAS checks performed during the last measurement period.	Number	A high number for the total runtime checks along with a high stack walk depth indicates performance overhead.

1.2.7 AspNetClrJit Test

c. The CLR (Common Language Runtime) is the execution environment for code written for the .NET Framework. The CLR manages the execution of .NET code, including memory allocation and garbage

collection (which helps avoid memory leaks), security (including applying differing trust levels to code from different sources), thread management, enforcing type-safety, and many other tasks.

d. The CLR works with every language available for the .NET Framework, so there is no need to have a separate runtime for each language. Code developed in a .NET language is compiled by the individual language compiler (such as the Visual Basic .NET compiler) into an intermediate format called Intermediate Language (IL). At runtime, this IL code generated by the compiler is just-in-time (JIT) compiled by the CLR into native code for the processor type the CLR is running on.

e. This AspNetClrJit test monitors the JIT compilation performed by the CLR. This compilation provides the flexibility of being able to develop with multiple languages and target multiple processor types while still retaining the performance of native code at execution time.

Purpose	Monitors the JIT compilation performed by the CLR		
Target of the test	An ASP .Net server		
Agent deploying the test	An internal agent		
Configurable parameters for the test	1. **TEST PERIOD** - How often should the test be executed 2. **HOST** - The host for which the test is to be configured 3. **PORT** - The port at which the specified **HOST** listens		
Outputs of the test	One set of results for the ASP .Net server being monitored		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	ASP .Net – Time in JIT: Indicates the percentage of elapsed time spent in JIT compilation; a JIT compilation phase is the phase when a method and its dependencies are being compiled..	Percent	
	ASP .Net – Data JIT rate: Indicates the rate at which IL bytes are jitted.	KB/Sec	
	ASP .Net – JIT failures: Indicates the number of methods the JIT compiler has failed to JIT during the last measurement period.	Number	An unusually high value may indicate a sudden increase in jit failures occured in the application.
	ASP .Net – Data jitted: Indicates the total IL bytes jitted during the last measurement period.	KB/Sec	

	ASP .Net – Methods jitted: Indicates the methods compiled Just-In-Time (JIT) by the CLR JIT compiler during the last measurement period.	Number	AppDomains (application domains) provide a secure and versatile unit of processing that the CLR can use to provide isolation between applications running in the same process.

1.3 The ASP .Net Apps Layer

The tests associated with this layer (see Figure 1.4) monitor the following:

- The application cache
- How well the appdomains perform during compilation
- How well the appdomains handle requests
- Performance of the applications deployed on the ASP .Net server
- Client connections to the ASP.Net server
- Sessions to the ASP .Net server
- The health of the interaction between the ASP .Net server and the MS SQL / Oracle database servers via the respective .Net Framework Data Providers

Figure 1.4: The tests associated with the ASP .Net Apps layer

1.3.1 ASP .Net App Cache Test

This test monitors the performance of the ASP.NET Application (or Application Domain) Cache.

Purpose	Monitors the performance of the ASP.NET Application (or Application Domain) Cache		
Target of the test	An ASP .Net server		
Agent deploying the test	An internal agent		
Configurable parameters for the test	1. **TEST PERIOD** - How often should the test be executed 2. **HOST** - The host for which the test is to be configured 3. **PORT** - The port at which the specified **HOST** listens		
Outputs of the test	One set of results for every ASP .Net application/application domain cache on a monitored ASP .Net server		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Cache total entries: The current number of entries in the cache (both User and Internal).	Number	
	Cache hit ratio: The current hit-to-miss ratio of all cache requests (both user and internal).	Percent	Physical I/O takes a significant amount of time, and also increases the CPU resources required. The server configuration should therefore ensure that the required information is available on the memory. A low value of this measure indicates that physical I/O is greater.
	Cache turnover rate The number of additions and removals to the cache per second (both user and internal).	Cached/Sec	A high turnover rate indicates that items are being quickly added and removed, which can be expensive.
	Cache api entries: The number of entries currently in the user cache.	Number	
	Cache user hit ratio: Total hit-to-miss ratio of user cache requests.	Percent	A high value of this measure is indicative of the good health of the server.
	Cache user turnover rate: The number of additions and removals to the user cache per second.	Cached/Sec	A high turnover rate indicates that items are being quickly added and removed, which can be expensive.

	Output cache entries: The number of entries currently in the Output Cache.	Number	
	Output cache hit ratio: The total hit-to-miss ratio of Output Cache requests	Percent	A high value of this measure is a sign of good health.
	Output cache turnover rate: The number of additions and removals to the output cache per second	Cached/Sec	Output caching allows you to store dynamic page and user control responses on any HTTP 1.1 cache-capable device in the output stream, from the originating server to the requesting browser. On subsequent requests, the page or user control code is not executed; the cached output is used to satisfy the request Sudden increases in the value of this measure are indicative of backend latency.

1.3.2 ASP .Net App Compile Test

This test reports how well the AppDomains perform during the compilation of the aspx, asmx, ascx or ashx files, loading of assemblies, and execution of assemblies to generate the page.

Purpose	Reports how well the AppDomains perform during the compilation of the aspx, asmx, ascx or ashx files, loading of assemblies, and execution of assemblies to generate the page		
Target of the test	An ASP .Net server		
Agent deploying the test	An internal agent		
Configurable parameters for the test	1. **TEST PERIOD** - How often should the test be executed 2. **HOST** - The host for which the test is to be configured 3. **PORT** - The port at which the specified **HOST** listens		
Outputs of the test	One set of results for every ASP .Net application domain on a monitored ASP .Net server		
Measurements made by the	**Measurement**	**Measurement Unit**	**Interpretation**

test	**Compilation total:** The total number of compilations that have taken place during the lifetime of the current Web server process. This occurs when a file with a .aspx, .asmx, asax,.ascx, or .ashx extension or code-behind source files are dynamically compiled on the server.	Number	
	Processing errors: The rate at which configuration and parsing errors occur.	Errors/Sec	A consistent increase in the value of this measure could prove to be fatal for the application domain.
	Compilation errors: The rate at which compilation errors occur. The response is cached, and this counter increments only once until recompilation is forced by a file change.	Errors/Sec	
	Runtime errors: The rate at which run-time errors occur.	Errors/Sec	
	Unhandled runtime errors: The rate of unhandled runtime exceptions.	Errors/Sec	A consistent increase in the value of this measure could prove to be fatal for the application domain. This measure however, does not include the following: • Errors cleared by an event handler (for example, by Page_Error or Application_Error) • Errors handled by a redirect page • Errors that occur within a try/catch block

1.3.3 ASP .Net App Requests Test

This test monitors how well the application domain handles requests.

Purpose	Monitors how well the application domain handles requests

Target of the test	An ASP .Net server		
Agent deploying the test	An internal agent		
Configurable parameters for the test	1. **TEST PERIOD** - How often should the test be executed 2. **HOST** - The host for which the test is to be configured 3. **PORT** - The port at which the specified **HOST** listens		
Outputs of the test	One set of results for every ASP .Net application domain on a monitored ASP .Net server		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Requests executing: The number of requests currently executing.	Number	This measure is incremented when the HttpRuntime begins to process the request and is decremented after the HttpRuntime finishes the request.
	Requests app queue: The number of requests currently in the application request queue.	Number	
	Requests not found: The number of requests that did not find the required resource.	Number	
	Requests not authorized: The number of request failed due to unauthorized access.	Number	Values greater than 0 indicate that proper authorization has not been provided, or invalid authors are trying to access a particular resource.
	Requests timed out: The number of requests timed out.	Number	
	Requests succeeded: The rate at which requests succeeded	Requests/Sec	

1.3.4 ASP .Net Applications Test

This test reports key statistics pertaining to applications deployed on the ASP .Net server.

Purpose	Reports key statistics pertaining to applications deployed on the ASP .Net server
Target of the test	An ASP .Net server

Agent deploying the test	An internal agent		
Configurable parameters for the test	1. **TEST PERIOD** - How often should the test be executed 2. **HOST** - The host for which the test is to be configured 3. **PORT** - The port at which the specified **HOST** listens		
Outputs of the test	One set of results for the ASP .Net server being monitored		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Request rate: Indicates the number of requests executed per second.	Number	This represents the current throughput of the application.
	Pipeline instances: Indicates the number of active pipeline instances for the ASP.NET application.	Number	Since only one execution thread can run within a pipeline instance, this number gives the maximum number of concurrent requests that are being processed for a given application. Ideally, the value of this measure should be low.
	Number of errors: Indicates the total sum of all errors that occur during the execution of HTTP requests.	Number	This measure should be kept at 0 or a very low value.

1.3.5 ASP Sql Clients Test

This test reports metrics pertaining to client connections to the ASP .Net server.

Purpose	Reports metrics pertaining to client connections to the ASP .Net server
Target of the test	An ASP .Net server
Agent deploying the test	An internal agent
Configurable parameters for the test	1. **TEST PERIOD** - How often should the test be executed 2. **HOST** - The host for which the test is to be configured 3. **PORT** - The port at which the specified **HOST** listens
Outputs of the test	One set of results for the ASP .Net server being monitored

Measurements made by the test	Measurement	Measurement Unit	Interpretation
	Connection pool size: Indicates the number of connection pools that have been created.	Number	If the connection pool maxes out while new connection requests are still coming in, you willl see connection requests refused, apparently at random. The cure in this case is simply to specify a higher value for the Max Pool Size property.
	Number of connections: Indicates the number of connections currently in the pool.	Number	
	Pooled connections: Indicates the number of connections that have been pooled.	Number	
	Pooled connections peak: Indicates the highest number of connections that have been used.	Number	If the value of this measure is at the Max Pool Size value, and the value of the *Failed connects* measure increases while the application is running, you might have to consider increasing the size of the connection pool.
	Failed connects: Indicates the number of connection attempts that have failed.	Number	If the connection pool maxes out while new connection requests are still coming in, you willl see connection requests refused, apparently at random. The cure in this case is simply to specify a higher value for the Max Pool Size property.

1.3.6 ASP .Net Sessions Test

This test monitors the sessions on the ASP .Net server.

Purpose	Monitors the sessions on the ASP .Net server
Target of the test	The ASP .Net objects
Agent deploying the test	An internal agent
Configurable parameters for the test	1. **TEST PERIOD** - How often should the test be executed 2. **HOST** - The host for which the test is to be configured 3. **PORT** - The port at which the specified **HOST** listens
Outputs of the test	One set of results for the ASP .Net server being monitored

Measurements made by the test	Measurement	Measurement Unit	Interpretation
	SQL connections: Indicates the number of connections to the SQL Server used by session state.	Number	An unusually high value may indicate a sudden increase in sessions to the SQL Server.
	State server connections: Indicates the number of connections to the StateServer used by session state.	Number	An unusually high value may indicate a sudden increase in sessions to the StateServer.
	Abandoned ASPNet application sessions: Indicates the number of sessions that have been explicitly abandoned during the last measurement period.	Number	
	Active ASPNet application sessions: Indicates the currently active sessions.	Number	
	Timedout ASPNet application sessions: Indicates the number of sessions that timed out during the last measurement period.	Number	
	ASPNet application sessions: Indicates the total number of sessions during the last measurement period.	Number	

f.

1.3.7 ASP.Net SQL Data Provider Test

A data provider in the .NET Framework serves as a bridge between an application and a data source. A .NET Framework data provider enables you to return query results from a data source, execute commands at a data source, and propagate changes in a DataSet to a data source.

The .Net Data Provider for SQL Server allows you to connect to a Microsoft SQL Server 7.0, 2000, and 2005 databases, and perform the above-mentioned operations. This test reports many useful metrics that shed light on the health of the interactions between the ASP .Net sever and the SQL server.

Purpose	Shed light on the health of the interactions between the ASP .Net sever and the SQL server		
Target of the test	The ASP .Net server		
Agent deploying the test	An internal agent		
Configurable parameters for the test	1. **TEST PERIOD** - How often should the test be executed 2. **HOST** - The host for which the test is to be configured 3. **PORT** - The port at which the specified **HOST** listens		
Outputs of the test	One set of results for the ASP .Net server being monitored		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Hard connects: Indicates the number of actual connections per second that are being made to a database server.	Connects/Sec	
	Hard disconnects: Indicates the number of actual disconnects per second that are being made to a database server.	Disconnects/Sec	
	Active connection pool groups: Indicates the number of currently active connection pool groups.	Number	The value of this measure is controlled by the number of unique connection strings that are found in the AppDomain.

	Active connection pools: Indicates the number of connection pools that are currently active.	Number	When a connection is first opened, a connection pool is created based on matching criteria that associates the pool with the connection string in the connection. Each connection pool is associated with a distinct connection string. If the connection string is not an exact match to an existing pool when a new connection is opened, a new pool is created. Connections are pooled per process, per application domain, per connection string, and, when integrated security is used, per Windows identity. When using Windows Authentication (integrated security), both the *Active connection pool groups* and *Active connection pools* measures are significant. The reason is that connection pool groups map to unique connection strings. When integrated security is used, connection pools map to connection strings and additionally create separate pools for individual Windows identities. For example, if Fred and Julie, each within the same AppDomain, both use the connection string **"Data Source=MySqlServer;Integrated Security=true"**, a connection pool group is created for the connection string, and two additional pools are created, one for Fred and one for Julie. If John and Martha use a connection string with an identical SQL Server login, **"Data Source=MySqlServer;User Id=lowPrivUser;Password=Strong?Password"**, then only a single pool is created for the **lowPrivUser** identity.
	Active connections: Indicates the number of connections that are currently in use.	Number	
	Free connections: Indicates the count of unused connections.	Number	Ideally, the value of this measure. A very low value indicates excessive connection usage.
	Inactive connection pools: Indicates the number of connection pools that have had no recent activity and are waiting to be disposed.	Number	

	Inactive connection pool groups: Indicates the number of inactive connection pool groups that were waiting to be deactivated i.e., to be pruned.	Number	
	Non-pooled connections: Indicates the number of active connections that are not using any of the connection pools.	Number	
	Pooled connections: Indicates the number of connections that are managed by the connection pooler.	Number	
	Reclaimed connections: Indicates the number of connections that have been reclaimed through garbage collection where Close or Dispose was not called by the application.	Number	Not explicitly closing or disposing connections hurts performance.
	Waiting connections: Indicates the number of connections that are currently awaiting completion of an action and are therefore unavailable for use by any other application.	Number	
	Soft connects: Indicates the rate at which connections are pulled from the connection pool.	Connects/Sec	
	Soft disconnects: Indicates the rate at which connections are returned to the connection pool.	Disconnects/Sec	

1.3.8 ASP .Net Oracle Data Provider Test

A data provider in the .NET Framework serves as a bridge between an application and a data source. A .NET Framework data provider enables you to return query results from a data source, execute commands at a data source, and propagate changes in a DataSet to a data source.

The Oracle Data Provider for .NET (ODP.NET) features optimized data access to the Oracle database from a .NET environment. ODP.NET allows developers to take advantage of advanced Oracle database functionality, including Real Application Clusters, XML DB, and advanced security. The data provider can be used from any .NET language, including C# and Visual Basic .NET.

This test reports many useful metrics that shed light on the health of the interactions between the ASP .Net sever and the Oracle database server.

Purpose	Sheds light on the health of the interactions between the ASP .Net sever and the Oracle database server		
Target of the test	The ASP .Net server		
Agent deploying the test	An internal agent		
Configurable parameters for the test	1. **TEST PERIOD** - How often should the test be executed 2. **HOST** - The host for which the test is to be configured 3. **PORT** - The port at which the specified **HOST** listens		
Outputs of the test	One set of results for the ASP .Net server being monitored		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Hard connects: Indicates the number of actual connections per second that are being made to a database server.	Connects/Sec	
	Hard disconnects: Indicates the number of actual disconnects per second that are being made to a database server.	Disconnects/Sec	
	Active connection pool groups: Indicates the number of currently active connection pool groups.	Number	The value of this measure is controlled by the number of unique connection strings that are found in the AppDomain.

	Active connection pools: Indicates the number of currently active connection pools.	Number	When a connection is first opened, a connection pool is created based on matching criteria that associates the pool with the connection string in the connection. Each connection pool is associated with a distinct connection string. If the connection string is not an exact match to an existing pool when a new connection is opened, a new pool is created. Connections are pooled per process, per application domain, per connection string, and, when integrated security is used, per Windows identity. When using Windows Authentication (integrated security), both the *Active connection pool groups* and *Active connection pools* measures are significant. The reason is that connection pool groups map to unique connection strings. When integrated security is used, connection pools map to connection strings and additionally create separate pools for individual Windows identities. For example, if Fred and Julie, each within the same AppDomain, both use the connection string **"Data Source=MySqlServer;Integrated Security=true"**, a connection pool group is created for the connection string, and two additional pools are created, one for Fred and one for Julie. If John and Martha use a connection string with an identical SQL Server login, **"Data Source=MySqlServer;User Id=lowPrivUser;Password=Strong?Password"**, then only a single pool is created for the **lowPrivUser** identity.
	Active connections: Indicates the number of connections that are currently in use.	Number	
	Free connections: Indicates the count of unused connections.	Number	Ideally, the value of this measure. A very low value indicates excessive connection usage.
	Inactive connection pools: Indicates the number of connection pools that have had no recent activity and are waiting to be disposed.	Number	

	Inactive connection pool groups: Indicates the number of inactive connection pool groups that were waiting to be deactivated i.e., to be pruned.	Number	
	Non-pooled connections: Indicates the number of active connections that are not using any of the connection pools.	Number	
	Pooled connections: Indicates the number of connections that are managed by the connection pooler.	Number	
	Reclaimed connections: Indicates the number of connections that have been reclaimed through garbage collection where Close or Dispose was not called by the application.	Number	Not explicitly closing or disposing connections hurts performance.
	Waiting connections: Indicates the number of connections that are currently awaiting completion of an action and are therefore unavailable for use by any other application.	Number	

	Soft connects: Indicates the rate at which connections are pulled from the connection pool.	Connects/Sec	
	Soft disconnects: Indicates the rate at which connections are returned to the connection pool.	Disconnects/Sec	

Chapter 2

Conclusion

This document has described in detail the monitoring paradigm used and the measurement capabilities of the eG Enterprise suite of products with respect to **ASP .Net servers**. For details of how to administer and use the eG Enterprise suite of products, refer to the user manuals.

We will be adding new measurement capabilities into the future versions of the eG Enterprise suite. If you can identify new capabilities that you would like us to incorporate in the eG Enterprise suite of products, please contact support@eginnovations.com. We look forward to your support and cooperation. Any feedback regarding this manual or any other aspects of the eG Enterprise suite can be forwarded to feedback@eginnovations.com.