Monitoring Voyager FrontEnd

eG Innovations Product Documentation

www.eginnovations.com

Total Performance Visibility

Table of Contents

Table of Figures

Chapter 1: Introduction

The advent of Internet technologies has probably had the most radical change in the banking and finance sector. Consumers and businesses alike can now perform transactions with banks on-line. Real-time payment and credit processing are now the norm.

Corillian, Inc.'s Voyager platform offers a secure, flexible, scalable set of Internet banking solutions for financial institutions. Voyager is a single platform that supports multiple lines of business - Consumer banking, Small Business Banking, Wealth Management, Credit Card Management, and Corporate Cash Management. Besides supporting all of Corillian's Line of Business Solutions and Enterprise Applications, Voyager also offers a quick and easy way to integrate the Corillian applications with third party and legacy applications.

To provide scalability, improved security, and enhanced performance, most Internet platforms are designed to include a number of tiers of applications. Corillian Voyager is no different. A web server front-end handles all user requests, while a Voyager Load Balancer (VLB) software on the web front-end serves to balance the load across all the servers in the farm. Along with the Transaction Processor (TP), the VLB handles session creation and management. Microsoft COM serves as the transport medium between the VLB and the TP. The Voyager TP processes customer requests received from the web server. Authentication and authorization of users is handled by the Authentication server (ATC) which is an integral part of the TP. Another component of the TP, the Voyager Response Engine interacts with the repositories to retrieve customer-specific data and providing responses back in XML format to the web front-end. The TP also includes a VLOG service which generates a comprehensive log of all customer operations. All log information as well as customer-specific data repositories are maintained in Microsoft SQL database. Some of the operations performed by Voyager involve interaction with third party hosts of the financial institution. The Host Servers handle these operations. The host servers are mainly responsible for communicating with the client systems and converting data from these systems into the Voyager format.

Figure 1.1 depicts the topology of the Voyager application.

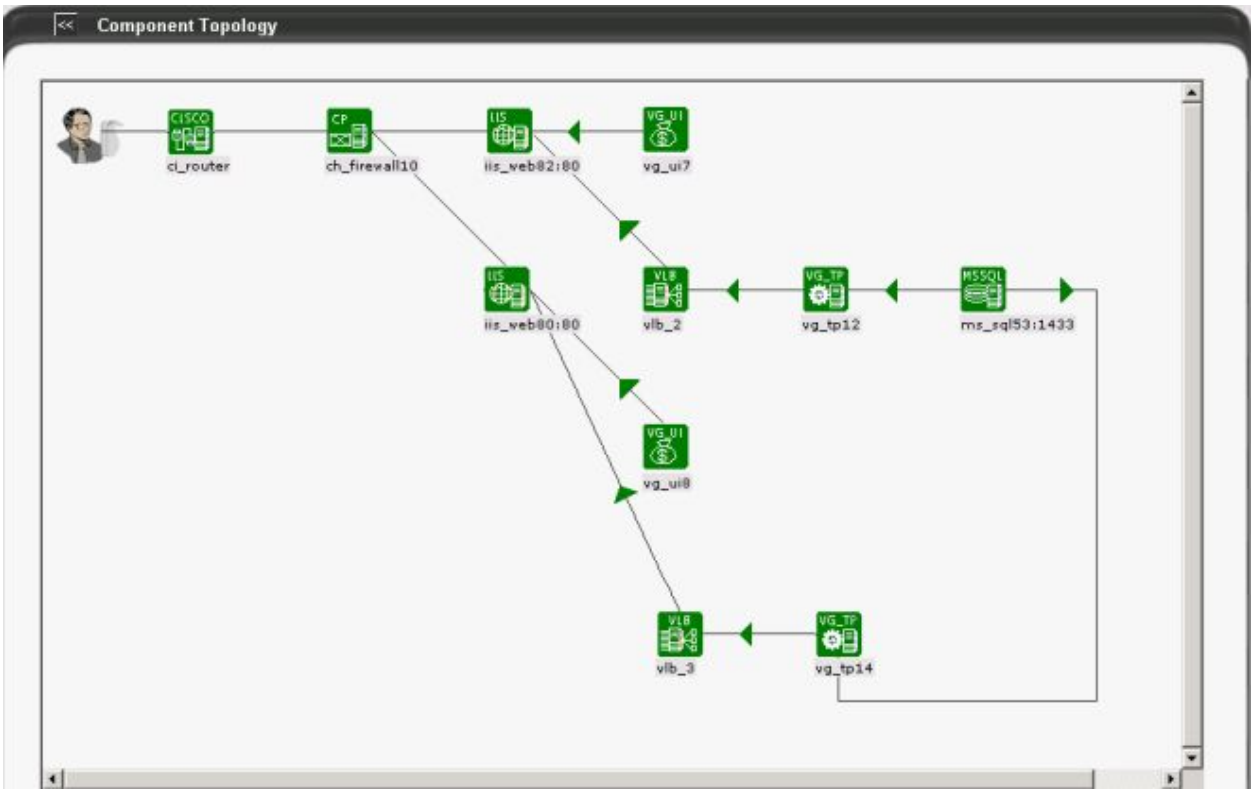

Figure 1.1: The topology of Voyager

The eG Enterprise system treats each component as a separate server and extracts critical performance statistics from them. This document deals with each of the components in great detail.

Chapter 2: How to Monitor Voyager FrontEnd using eG Enterprise?

eG is capable of monitoring the Voyager FrontEnd in both agent based and agentless manners.

2.1 Managing the Voyager FrontEnd

The eG Enterprise cannot automatically discover the Voyager FrontEnd so that you need to manually add the component for monitoring. Remember that the eG Enterprise automatically manages the components that are added manually. To manage a Voyager FrontEnd component, do the following:

1. Log into the eG administrative interface.
2. Follow the Components -> Add/Modify menu sequence in the **Infrastructure** tile of the **Admin** menu.
3. In the **COMPONENT** page that appears next, select Voyager FrontEnd as the **Component type**. Then, click the **Add New Component** button. This will invoke Figure 2.1.

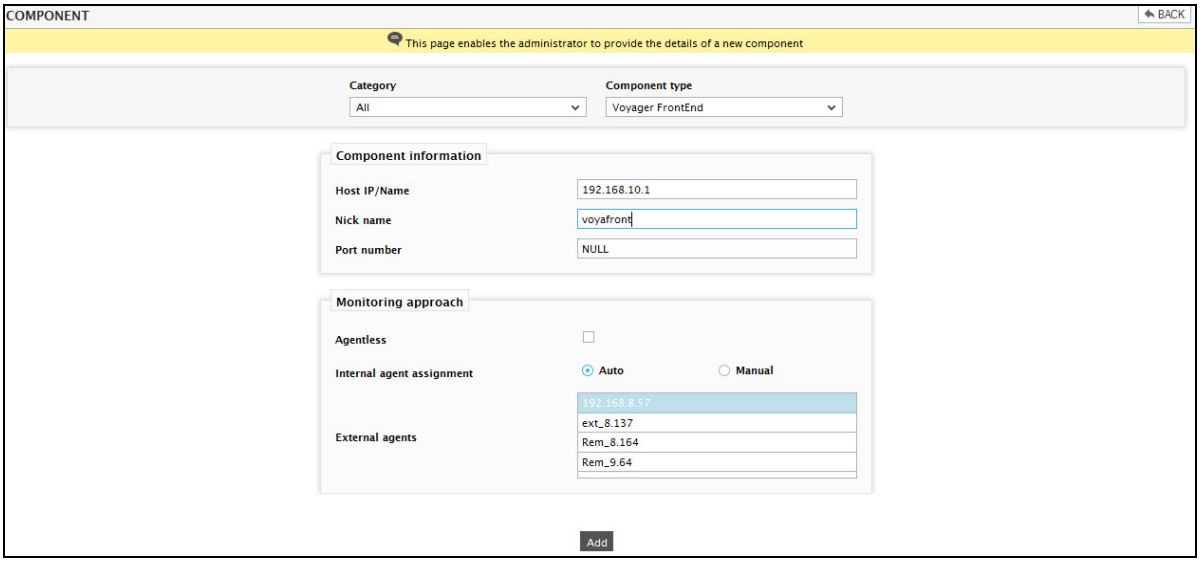

Figure 2.1: Figure 2.1: Adding a Voyager FrontEnd component

3. Specify the **Host IP/Name** and the **Nick name** of the Voyager FrontEnd in Chapter 2. Then, click the **Add** button to register the changes.
4. Then, sign out of the eG administrative interface.

Chapter 3: Monitoring the Voyager Front End (FE)

eG Enterprise uses the Voyager FrontEnd hierarchical model (see Figure 3.1) to represent the front end of the Voyager application.

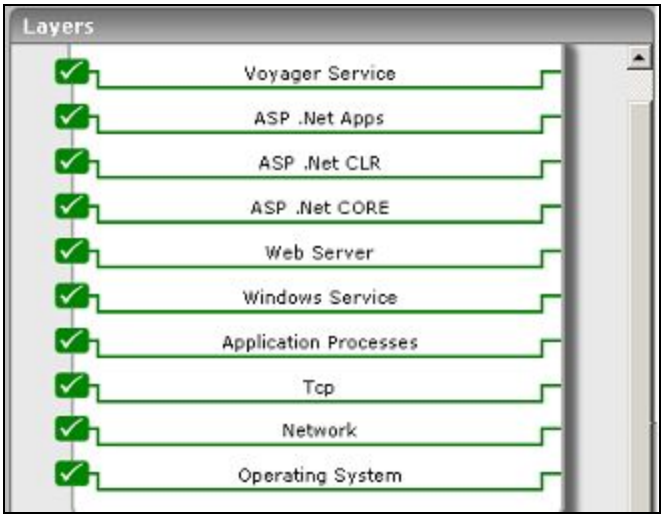

Figure 3.1: The layer model of the Voyager Front End component

Of the top 4 layers in Figure 3.1, the last 3 layers execute tests that extract .NET-specific statistics. The first layer – i.e., the **Voyager Service** layer- is the one that reports **Voyager presentation** layer metrics. The sections to come discuss the top 4 layers of Figure 3.1 only, as all other layers have been dealt with in the *Monitoring Unix and Windows Servers* document.

3.1 The ASP .Net CORE Layer

The test mapped to this layer (see Figure 3.3) monitors the performance of the worker process of the ASP .NET objects.

Figure 3.2: The tests associated with the ASP .Net CORE Layer

3.1.1 ASP .Net Workers Test

This reports statistics pertaining to the performance of the worker process of the ASP .NET objects in the Voyager user interface.

Target of the test : The ASP .NET objects in the Voyager user interface

Agent deploying the test : An internal agent

Outputs of the test : One set of results for the ASP .NET objects in the Voyager user interface being monitored.

Configurable parameters for the test

Parameter	Description
Test Period	How often should the test be executed.
Host	The IP address of the host for which this test is to be configured.
Port	Refers to the port at which the specified host listens to.

Measurements made by the test

Measurement	Description	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

Measurement	Description	Measurement Unit	Interpretation
			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

Measurement	Description	Measurement Unit	Interpretation
			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

Measurement	Description	Measurement Unit	Interpretation
			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.

3.2 The ASP .Net CLR Layer

The tests associated with this layer (see Figure 3.3) 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

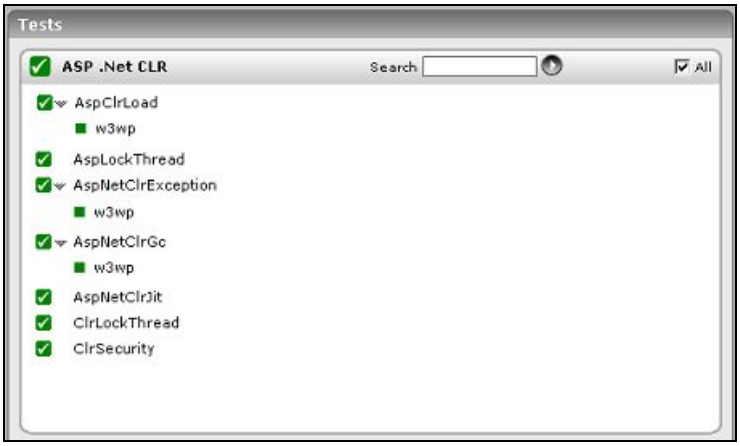

Figure 3.3: The tests associated with the ASP .Net CLR layer

3.2.1 ASP Lock Thread Test

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

Target of the test : The ASP .NET objects in the Voyager user interface

Agent deploying the test : An internal agent

Outputs of the test : One set of results for the ASP .NET objects in the Voyager user interface being monitored.

Configurable parameters for the test

Parameter	Description
Test Period	How often should the test be executed.
Host	The IP address of the host for which this test is to be configured.
Port	Refers to the port at which the specified host listens to.

Measurements made by the test

Measurement	Description	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	

Measurement	Description	Measurement Unit	Interpretation
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	

3.2.2 ASP .Net CLR Exceptions Test

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

Target of the test : The ASP .NET objects in the Voyager user interface

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every worker process on the ASP .NET objects in the Voyager user interface being monitored.

Configurable parameters for the test

Parameter	Description
Test Period	How often should the test be executed.
Host	The IP address of the host for which this test is to be configured.
Port	Refers to the port at which the specified host listens to.

Measurements made by the test

Measurement	Description	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.

3.2.3 ASP .Net CLR GC Test

This test monitors the memory allocation activity of the ASP .NET objects in the Voyager user interface, in terms of heaps when objects are created and managed.

Target of the test : The ASP .NET objects in the Voyager user interface

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every worker process on the ASP .NET objects in the Voyager user interface being monitored.

Configurable parameters for the test

Parameter	Description
Test Period	How often should the test be executed.
Host	The IP address of the host for which this test is to be configured.
Port	Refers to the port at which the specified host listens to.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Heap memory 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	

Measurement	Description	Measurement Unit	Interpretation
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.

3.2.4 ASP CLR Load Test

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.

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.

Target of the test : The ASP .NET objects in the Voyager user interface

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every worker process on the ASP .NET objects in the Voyager user interface being monitored.

Configurable parameters for the test

Parameter	Description
Test Period	How often should the test be executed.
Host	The IP address of the host for which this test is to be configured.
Port	Refers to the port at which the specified host listens to.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Classes loaded	The number of classes currently loaded in all assemblies.	Number	An unusually high value may indicate a sudden increase in classes which loaded on to this .NET application.
Current 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

Measurement	Description	Measurement Unit	Interpretation
			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	Number	

Measurement	Description	Measurement Unit	Interpretation
	AppDomains loaded during the last measurement period.		
Number of assemblies	The number of assemblies loaded during the last measurement period.	Number	

3.2.5 Clr Lock ThreadsTest

This test monitors the thread locking activity on the ASP .NET objects in the Voyager user interface.

Target of the test : The ASP .NET objects in the Voyager user interface

Agent deploying the test : An internal agent

Outputs of the test : One set of results for the ASP .NET objects in the Voyager user interface being monitored.

Configurable parameters for the test

Parameter	Description
Test Period	How often should the test be executed.
Host	The IP address of the host for which this test is to be configured.
Port	Refers to the port at which the specified host listens to.

Measurements made by the test

Measurement	Description	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

Measurement	Description	Measurement Unit	Interpretation
			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.

3.2.6 Clr Security Test

This test monitors the security system activity of the ASP .NET objects in the Voyager user interface.

Target of the test : The ASP .NET objects in the Voyager user interface

Agent deploying the test : An internal agent

Outputs of the test : One set of results for the ASP .NET objects in the Voyager user interface being monitored.

Configurable parameters for the test

Parameter	Description
Test Period	How often should the test be executed.
Host	The IP address of the host for which this test is to be configured.
Port	Refers to the port at which the specified host listens to.

Measurements made by the test

Measurement	Description	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.

3.2.7 ASP .Net CLR JIT Test

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.

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.

This 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.

Target of the test : The ASP .NET objects in the Voyager user interface

Agent deploying the test : An internal agent

Outputs of the test : One set of results for the ASP .NET objects in the Voyager user interface being monitored.

Configurable parameters for the test

Parameter	Description
Test Period	How often should the test be executed.
Host	The IP address of the host for which this test is to be configured.
Port	Refers to the port at which the specified host listens to.

Measurements made by the test

Measurement	Description	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	Indicates the rate at which	KB/Sec	

Measurement	Description	Measurement Unit	Interpretation
rate	IL bytes are jitted.		
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.

3.3 The ASP .Net Apps Layer

The tests associated with this layer (see Figure 3.4) 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 objects of the Voyager FrontEnd
- Client connections to the ASP.Net objects of the Voyager FrontEnd
- Sessions to the ASP .Net objects of the Voyager FrontEnd

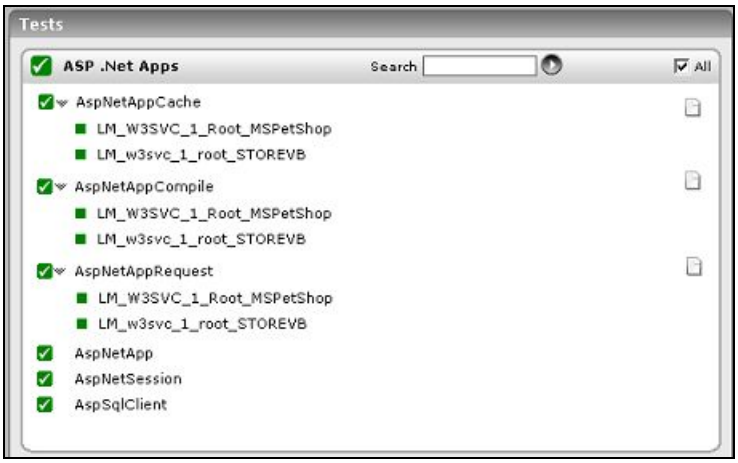

Figure 3.4: The tests associated with the ASP .Net Apps layer

3.3.1 ASP .Net App Cache Test

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

Target of the test : The ASP .NET objects in the Voyager user interface

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every ASP .NET application/application domain cache on a monitored Voyager user interface.

Configurable parameters for the test

Parameter	Description
Test Period	How often should the test be executed.
Host	The IP address of the host for which this test is to be configured.
Port	Refers to the port at which the specified host listens to.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Cache total entries	The current number of entries in the cache (both User and Internal).	Number	

Measurement	Description	Measurement Unit	Interpretation
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	Sudden increases in the value of this measure are indicative of backend latency.

3.3.2 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.

Target of the test : The ASP .NET objects in the Voyager user interface

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every ASP .NET application domain on a monitored Voyager user interface.

Configurable parameters for the test

Parameter	Description
Test Period	How often should the test be executed.
Host	The IP address of the host for which this test is to be configured.
Port	Refers to the port at which the specified host listens to.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
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	
Preprocessing 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

Measurement	Description	Measurement Unit	Interpretation
			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

3.3.3 ASP .Net App Requests Test

This test monitors how well the application domain handles requests.

Target of the test : The ASP .NET objects in the Voyager user interface

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every ASP .NET application domain on a monitored Voyager user interface.

Configurable parameters for the test

Parameter	Description
Test Period	How often should the test be executed.
Host	The IP address of the host for which this test is to be configured.
Port	Refers to the port at which the specified host listens to.

Measurements made by the test

Measurement	Description	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	

3.3.4 ASP .Net App Test

This test reports key statistics pertaining to applications deployed on the ASP .NET objects in the Voyager user interface.

Target of the test : The ASP .NET objects in the Voyager user interface

Agent deploying the test : An internal agent

Outputs of the test : One set of results for the ASP .NET objects in the Voyager user interface being monitored.

Configurable parameters for the test

Parameter	Description
Test Period	How often should the test be executed.

Parameter	Description
Host	The IP address of the host for which this test is to be configured.
Port	Refers to the port at which the specified host listens to.

Measurements made by the test

Measurement	Description	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.

3.3.5 ASP Sql Clients Test

This test reports metrics pertaining to client connections to the ASP .NET objects in the Voyager user interface.

Target of the test : The ASP .NET objects in the Voyager user interface

Agent deploying the test : An internal agent

Outputs of the test : One set of results for the ASP .NET objects in the Voyager user interface being monitored.

Configurable parameters for the test

Parameter	Description
Test Period	How often should the test be executed.

Parameter	Description
Host	The IP address of the host for which this test is to be configured.
Port	Refers to the port at which the specified host listens to.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Number of connection pools	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 will 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.

3.3.6 ASP .Net Sessions Test

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

Target of the test : The ASP .NET objects in the Voyager user interface

Agent deploying the test : An internal agent

Outputs of the test : One set of results for the ASP .NET objects in the Voyager user interface being monitored.

Configurable parameters for the test

Parameter	Description
Test Period	How often should the test be executed.
Host	The IP address of the host for which this test is to be configured.
Port	Refers to the port at which the specified host listens to.

Measurements made by the test

Measurement	Description	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	
Total ASPNet application sessions	Indicates the total number of sessions during the last measurement period.	Number	

3.4 The Voyager Service Layer

This layer reports the presentation server statistics of the Voyager FrontEnd.

Figure 3.5: The test associated with the Voyager Service layer

Since the VlbTest, VlbDatabaseTest, and ClnTraceTest in Figure 3.5 have been dealt with in the *Monitoring Transaction Processor* document, this section will discuss only the VoyagerUI test.

3.4.1 Voyager UI Test

The test reports statistics pertaining to the Voyager UI.

Target of the test : The ASP .NET objects in the Voyager user interface

Agent deploying the test : An internal agent

Outputs of the test : One set of outputs for the monitored Voyager user interface.

Configurable parameters for the test

Parameter	Description
Test Period	How often should the test be executed.
Host	The IP address of the host for which this test is to be configured.
Port	Refers to the port at which the specified host listens to.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Number of executions	Indicates the number of executions.	Number	

Measurement	Description	Measurement Unit	Interpretation
Transaction processing time	Indicates the average time taken to process transactions.	Secs	
Avg operation time	Indicates the average time for performing operations.	Secs	
Input data rate	Indicates the rate of incoming data.	KB/Sec	
Output data rate	Indicates the rate of outgoing data.	KB/Sec	
Operation rate	Indicates the rate of operations performed using the Voyager UI.	Operations/Sec	

About eG Innovations

eG Innovations provides intelligent performance management solutions that automate and dramatically accelerate the discovery, diagnosis, and resolution of IT performance issues in on-premises, cloud and hybrid environments. Where traditional monitoring tools often fail to provide insight into the performance drivers of business services and user experience, eG Innovations provides total performance visibility across every layer and every tier of the IT infrastructure that supports the business service chain. From desktops to applications, from servers to network and storage, from virtualization to cloud, eG Innovations helps companies proactively discover, instantly diagnose, and rapidly resolve even the most challenging performance and user experience issues.

eG Innovations is dedicated to helping businesses across the globe transform IT service delivery into a competitive advantage and a center for productivity, growth and profit. Many of the world's largest businesses use eG Enterprise to enhance IT service performance, increase operational efficiency, ensure IT effectiveness and deliver on the ROI promise of transformational IT investments across physical, virtual and cloud environments.

To learn more visit www.eginnovations.com.

Contact Us

For support queries, email support@eginnovations.com.

To contact eG Innovations sales team, email sales@eginnovations.com.