Monitoring Corillian's Voyager

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 2000, Windows 2003 and Windows 2008 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

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.

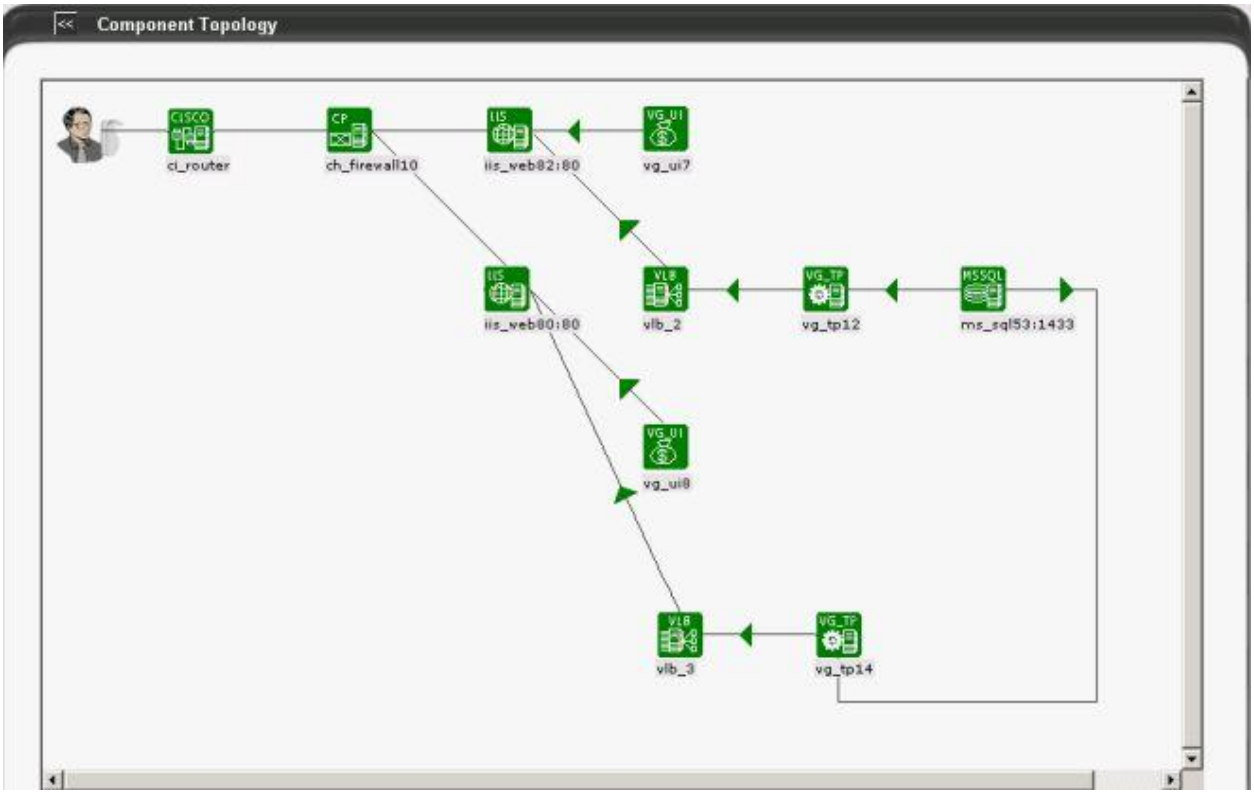

Figure 1.1: The topology of Voyager

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

Monitoring the Voyager Front End (FE)

eG Enterprise uses the *Voyager FrontEnd* hierarchical model (see Figure 2.1) to represent the front end of the Voyager application.

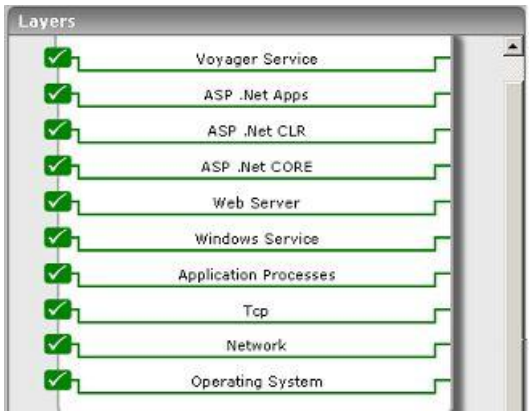

Figure 2.1: The layer model of the Voyager Front End component

Of the top 4 layers in Figure 2.1, 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 2.1 only, as all other layers have been dealt with in the *Monitoring Unix and Windows Servers* document.

2.1 The ASP .Net CORE Layer

The test mapped to this layer (see Figure 2.3) monitors the performance of the worker process of the ASP .NET objects.

Figure 2.2: The tests associated with the ASP .Net CORE Layer

2.1.1 AspNetWorker Test

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

Purpose	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		
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 objects in the Voyager user interface 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.

2.2 The ASP .Net CLR Layer

The tests associated with this layer (see Figure 2.3) 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 2.3: The tests associated with the ASP .Net CLR layer

2.2.1 AspLockThread 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	The ASP .NET objects in the Voyager user interface		
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 objects in the Voyager user interface 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	

2.2.2 AspNetClrException Test

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	The ASP .NET objects in the Voyager user interface
Agent deploying the	An internal agent

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 every worker process on the ASP .NET objects in the Voyager user interface being monitored		
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.

2.2.3 AspNetClrGc 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.

Purpose	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		
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 objects in the Voyager user interface being monitored		
Measurements made by the test	**Measurement**	**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	

	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.

2.2.4 AspClrLoad 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.

Purpose	Monitors the classes and assemblies loaded on to an ASP .Net application		
Target of the test	The ASP .NET objects in the Voyager user interface		
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 objects in the Voyager user interface being monitored		
Measurements made by the test	**Measurement**	**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 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	
	Number of assemblies: The number of assemblies loaded during the last measurement period.	Number	

2.2.5 ClrLockThread Test

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

Purpose	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

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 objects in the Voyager user interface 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.

2.2.6 ClrSecurity Test

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

Purpose	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		
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 objects in the Voyager user interface 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.

2.2.7 AspNetClrJit 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 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	The ASP .NET objects in the Voyager user interface		
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 objects in the Voyager user interface 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.

2.3 The ASP .Net Apps Layer

The tests associated with this layer (see Figure 2.4) 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

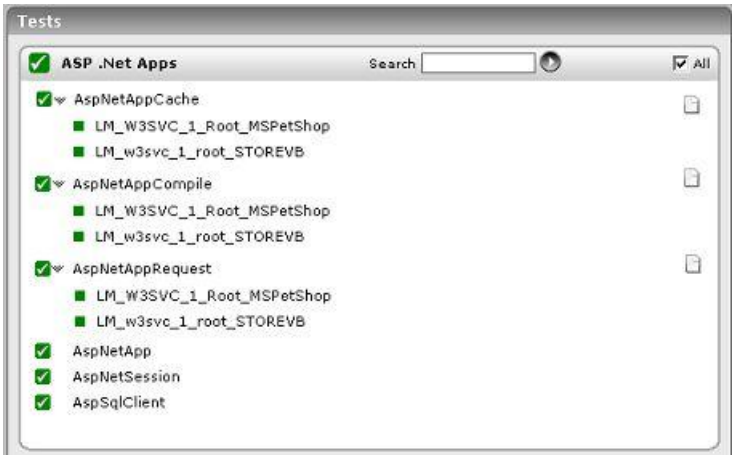

Figure 2.4: The tests associated with the ASP .Net Apps layer

2.3.1 AspNetAppCache 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	The ASP .NET objects in the Voyager user interface		
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 Voyager user interface		
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	Sudden increases in the value of this measure are indicative of backend latency.

2.3.2 AspNetAppCompile 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	The ASP .NET objects in the Voyager user interface
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 Voyager user interface		
Measurements made by the test	**Measurement**	**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 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

2.3.3 AspNetAppRequest Test

This test monitors how well the application domain handles requests.

Purpose	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		
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 Voyager user interface		
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	

2.3.4 AspNetApp Test

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

Purpose	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		
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 objects in the Voyager user interface 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.

2.3.5 AspSqlClient Test

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

Purpose	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		
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 objects in the Voyager user interface being monitored		
Measurements made by the test	**Measurement**	**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 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.

2.3.6 AspNetSession 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 in the Voyager user interface		
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 objects in the Voyager user interface 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	
	Total ASPNet application sessions: Indicates the total number of sessions during the last measurement period.	Number	

2.4 The Voyager Service Layer

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

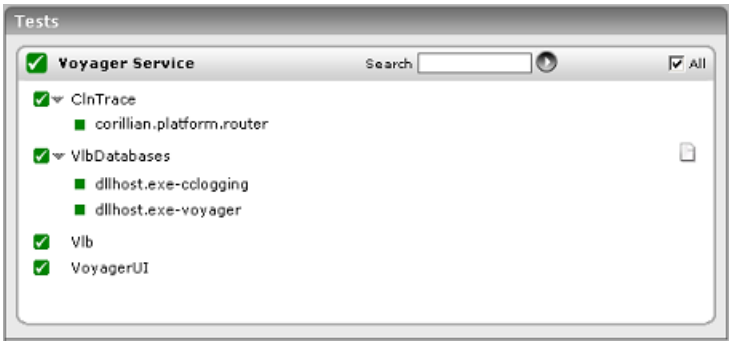

Figure 2.5: The test associated with the Voyager Service layer

Since the VlbTest, VlbDatabaseTest, and ClnTraceTest in Figure 2.5 have been dealt with in the Chapter 3 of this document, this section will discuss only the VoyagerUI test.

2.4.1 VoyagerUI Test

The test reports statistics pertaining to the Voyager UI.

Purpose	Reports statistics relating to the Voyager front end		
Target	The Voyager front end		
Agent deploying this test	Internal agent		
Configurable parameters for this 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 to which the specified **HOST** listens		
Outputs of the test	One set of outputs for the monitored Voyager user interface		
Measurements of the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Number of executions: Indicates the number of executions.	Number	
	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	

Chapter 3

Monitoring the Transaction Processor (TP)

The eG Enterprise system monitors the Transaction Processor using the *TP* monitoring model (see Figure 3.1). The Voyager Transaction Processor handles many functions including the following:

- Session management: The Voyager Load Balancer (VLB) shares this responsibility with the Transaction Processor.
- Templates
- Response engine
- Host server well-being, and
- Garbage collection.

The layer model of the *TP* server is shown in Figure 3.1.

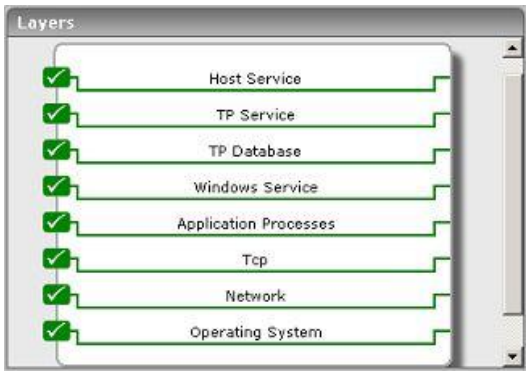

Figure 3.1: The layer model of the TP server

The sub-sections that will follow will elaborately discuss the top 3 layers of Figure 3.1 only as the remaining layers have been discussed extensively in the *Monitoring Unix and Windows Servers* document.

3.1 The TP Database Layer

This layer closely monitors the accesses to the TP database and authentication database and returns a wide variety of metrics pertaining to the database. The tests associated with this layer are as follows (see Figure 3.2):

Figure 3.2: The tests associated with the TP Database layer

3.1.1 AtcDatabase Test

This test reports statistics relating to the authentication database, which is a SQL database that contains data used in processing requests and responses. The authentication database consists of authentication data and authorization data, though there is no direct sharing of tables.

Purpose	Reports statistics relating to the authentication database		
Target	The TP server		
Agent deploying this test	Internal agent		
Configurable parameters for this 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 to which the specified **HOST** listens		
Outputs of the test	One set of outputs for every authentication database that is monitored		
Measurements of the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Average ADO response time: Indicates the average ADO response time over the last minute.	Secs	A significant increase in the value of this measure denotes a slowdown of the ATC database.

	Average connections used: Indicates the average number of database connections used in the last minute.	Number	
	Database pool size: Indicates the number of connections that are available in the DBManager Connection Pool to execute transactions through ADO.	Number	Compare the pool size with the connections in use. If the connections is use approaches the pool size, you may want to resize the pool.
	Pool availability: Indicates whether the pool is available or not.	Percent	If the value of this measure is 0, it indicates that the pool is not available. The value 100, on the other hand, indicates that the pool is available.
	Request execution rate: Indicates the rate of requests currently being executed through ADO.	Reqs/Sec	
	Requests waiting rate: Indicates the rate at which requests are waiting to get executed.	Reqs/Sec	Ideally, this value should be close to 0. An increase in requests waiting for database execution could result in the login process being slow.
	Error rate of SQL queries to ATC DB: Indicates the number of SQL errors thrown per second.	Errors/Sec	
	Average transaction response time: Indicates the average transaction response time during the last minute.	Secs	

3.1.2 TpDatabase Test

This test reports statistics pertaining to the transaction processor's database.

Purpose	Reports statistics pertaining to the transaction processor's database
Target	The TP server
Agent deploying this test	Internal agent

Configurable parameters for this 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 to which the specified **HOST** listens		
Outputs of the test	One set of outputs for every TP database monitored		
Measurements of the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Avg ADO response time: Indicates the average ADO response time over the last minute.	Secs	A significant increase in the value of this measure denotes a slowdown of the TP database.
	Avg connections used: Indicates the average number of connections used in the last minute.	Number	
	Database pool size: Indicates the number of connections that is available in the DBManager Connection Pool to execute transactions through ADO.	Number	Compare the pool size with the connections in use. If the connections is use approaches the pool size, you may want to resize the pool.
	Database pool availability: Indicates whether the pool is available or not.	Percent	If the value of this measure is 0, it indicates that the pool is not available. The value 100, on the other hand, indicates that the pool is available.
	Request execution rate: Indicates the rate of requests currently being executed through ADO.	Reqs/Sec	
	Requests waiting rate: Indicates the rate at which requests are waiting to get executed.	Reqs/Sec	Ideally, this value should be close to 0. An increase in requests waiting for database execution could result in the login process being slow.
	SQL error rate: Indicates the number of SQL errors thrown per second.	Errors/Sec	
	Avg transaction response time: Indicates the average transaction response time during the last minute.	Secs	

3.2 The TP Service Layer

Figure 3.3 depicts the tests associated with this layer. These tests monitor the overall health of the transaction processor.

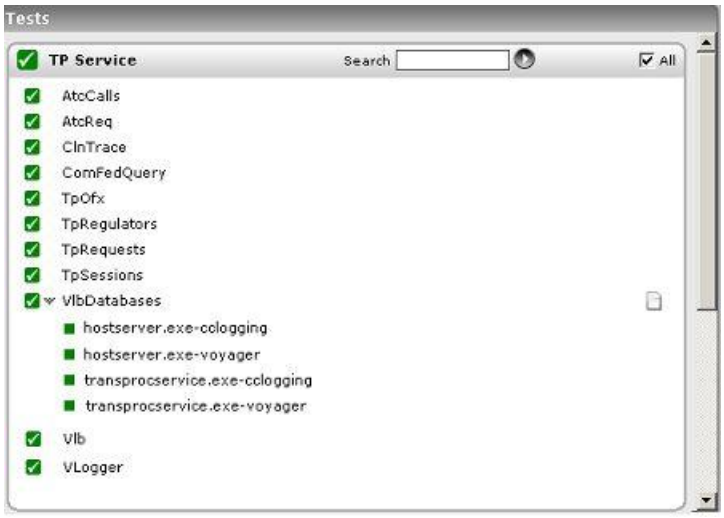

Figure 3.3: The tests associated with the TP Service layer

3.2.1 AtcCalls Test

This test reports statistics pertaining to the authentication service of the Voyager. The authentication service, which exists within each Transaction Processor, provides a central user authentication source for a number of Voyager products and add-ons, including consumer and business banking, bill payment, and One Source Register (OSR).

Purpose	Reports statistics pertaining to the authentication service of the Voyager		
Target	The TP server		
Agent deploying this test	Internal agent		
Configurable parameters for this 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 to which the specified **HOST** listens		
Outputs of the test	One set of outputs for the TP server being monitored		
Measurements	**Measurement**	**Measurement Unit**	**Interpretation**

of the test	**Database calls:** Indicates the rate of direct database calls during the last minute.	Calls/Sec	A decrease in the value of this measure would indicate that there is a significant slowing of the Authentication Service and it cannot keep up with the load required.
	HostServer queue size: Indicates the length of a queue of transactions awaiting assignment to a host server.	Number	Ideally, the queue size should be low.
	HostServer response rate: Indicates the rate of responses from the host server during the past minute.	Reqs/min	
	Authentication calls pending: Indicates the count of authentication API calls currently in progress.	Number	A large increase in the value of this measure points towards a degraded response time between the Transaction Processor and the host, or a slowdown in the host itself.
	Pending database calls: Indicates the count of database API calls currently in progress.	Number	A large increase in the value of this measure points towards a degraded response time between the Transaction Processor and the TP database.
	Pending identity calls: Indicates the count of identity API calls currently in progress.	Number	
	Pending pinvault calls: Indicates the count of pin vault API calls currently in progress.	Number	
	Pin vault calls: Indicates the rate of pin vault calls. The Pin Vault is where encrypted aliases are stored and the data is stored with the rest of the authorization and authentication database, but is managed by a distinct COM (in-proc) object.	Calls/Sec	A decrease in the value of this measure would indicate that there is a significant slowing of the Authentication Service and it cannot keep up with the load required.

	Token cache ratio: Indicates the token cache ratio during the last minute	Percent	
	Valid auth tokens: Indicates the count of valid authentication tokens. A token is a 32-character string that is a hex representation of a 128-bit random number. A token is created by combining a GUID with other environmental data; this information is then sent through an MD5 secure hash.	Number	A sharp increase in this measure may mean sessions are not expiring, there are too many users on the system, and response times are high. If this measure continues to increase even after peak load, make sure that the AuthTokenPurge Task in SQL Server is running regularly.

3.2.2 AtcReq Test

This test monitors the requests placed with the authentication service.

Purpose	Monitors the requests placed with the authentication service		
Target	The TP server		
Agent deploying this test	Internal agent		
Configurable parameters for this 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 to which the specified **HOST** listens		
Outputs of the test	One set of outputs for the TP server being monitored		
Measurements of the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Active requests to the ATC: Indicates the number of requests that are actively executing.	Number	
	Current requests to the ATC: Indicates the total number of requests upon the service.	Number	

	Current signoff requests: Indicates the number of sign off requests that are currently being executed.	Number	
	Current signon requests: Indicates the number of sign on requests that are currently being executed.	Number	Occasional spikes in this value are OK, as long as it is able to recover and this value returns to the normal level. However, a consistent increase in this value should be investigated, as it indicates a bottleneck elsewhere in the system (typically, the host or the database).
	Avg. request wait time in ATC: Indicates the average time for which the requests that began active execution in the past second had to wait.	Secs	
	Signon execution time in ATC: Indicates the average time it took to execute signon requests which completed in the past minute. This time includes time authenticating against backend host and updating some of the user information.	Secs	
	Signon requests to ATC: Indicates the total number of signon requests received since the last measurement period.	Number	
	Signoff requests to ATC: Indicates the total number of signoff requests received since the last measurement period.	Number	
	Waiting requests for ATC: Indicates the number of requests that are waiting to begin active execution.	Number	This value should be close to 0.

	Requests timed out in the ATC: Indicates the number of requests that timed out in the past second while waiting to become active.	Number	

3.2.3 ClnTrace Test

The CCLogging database stores trace logging information generated by Voyager components and plug-ins. This information can be used for monitoring and troubleshooting Voyager. The ClnTrace test monitors the trace logs and reports metrics such as the number of enqueued requests and the number that can be traced to the database.

Purpose	Monitors the trace logs and reports metrics such as the number of enqueued requests and the number that can be traced to the database		
Target	The TP server		
Agent deploying this test	Internal agent		
Configurable parameters for this 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 to which the specified **HOST** listens		
Outputs of the test	One set of outputs for every CCLogging database monitored		
Measurements of the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Logging queue length: Indicates the number of requests that are currently in the queue waiting to be logged.	Number	This should ideally be 0.
	Logging threads count: Indicates the number of threads that are currently tracing to the database.	Number	Many threads indicate slow CCLogging database.

3.2.4 ComFedQuery Test

This test monitors the error statistics in the CCLogging database.

Purpose	Monitors the error statistics in the CCLogging database
Target	The TP server

<table>
<tr><td>Agent deploying this test</td><td colspan="3">Internal agent</td></tr>
<tr><td>Configurable parameters for this test</td><td colspan="3">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 to which the specified HOST listens
4. instance - Enter the name of a specific MS SQL instance that is to be monitored. The default value of this parameter is "default". To monitor an MS SQL instance named "CFS", enter this as the value of the INSTANCE parameter.
5. dbhost - The IP/host name of the database host
6. dbport - The port at which the database host listens
7. username – The user name to be used for logging into the database. This user should be authorized to connect to the CCLogging database.
8. password - Enter the password for logging into the CClogging database.
9. confirm password - Confirm the password by retyping it in the CONFIRM PASSWORD text box.
10. message - Provide a comma-separated list of error message patterns to be monitored in the format, Displayname:messagepattern. Here, the Displayname is just a name that is used to uniquely identify the message pattern, and will be displayed in the monitor interface. The messagepattern is an expression of the form - *expr* or expr or *expr or expr* or *expr1*expr2*... or expr1*expr2, etc. A leading '*' signifies any number of leading characters, while a trailing '*' signifies any number of trailing characters. For example, Error1:*cc_err*,Error2:*login*.
11. detailed diagnosis - To make diagnosis more efficient and accurate, the eG Enterprise system embeds an optional detailed diagnostic capability. With this capability, the eG agents can be configured to run detailed, more elaborate tests as and when specific problems are detected. To enable the detailed diagnosis capability of this test for a particular server, choose the On option against DETAILED DIAGNOSIS. To disable the capability, click on the Off option.
The option to selectively enabled/disable the detailed diagnosis capability will be available only if the following conditions are fulfilled:
➢ The eG manager license should allow the detailed diagnosis capability
➢ Both the bad and normal frequencies configured for the detailed diagnosis measures should not be 0.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of outputs for every CCLogging database monitored</td></tr>
<tr><td rowspan="2">Measurements of the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
<tr><td>Errors found:
Indicates the number of error messages logged into the CCLogging database.</td><td>Number</td><td>The detailed diagnosis of this measure, if enabled, provides the details of the error messages that were logged into the CCLogging database.</td></tr>
</table>

3.2.5 TpOfx Test

The Voyager platform supports a wide variety of applications built to support the Online Banking community. One such application is OFX, which is a protocol used by Personal Finance Managers such as Intuit Quicken and Microsoft Money to download financial data. This test reports statistics pertaining to OFX.

Purpose	Reports statistics pertaining to OFX		
Target	The TP server		
Agent deploying this test	Internal agent		
Configurable parameters for this 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 to which the specified **HOST** listens		
Outputs of the test	One set of outputs for the TP server being monitored		
Measurements of the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Current requests: Indicates the current number of OFX requests being handled by this TP.	Number	
	Request execution time: Indicates the average time it took to execute OFX requests which completed in the past minute.	Secs	A high value indicates a TP slowdown.
	Request rate: Indicates the number of OFX requests that have been executed by this TP in the past second.	Reqs/Sec	

3.2.6 TpRegulators Test

This test monitors the performance of the regulator on a Transaction Processor (TP).

Purpose	Reports statistics pertaining to the regulator on a TP
Target	The TP server
Agent deploying this test	Internal agent

Configurable parameters for this 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 to which the specified **HOST** listens		
Outputs of the test	One set of outputs for every regulator monitored		
Measurements of the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Active requests: Indicates the number of requests that are actively executing. If the regulator is enabled, this value will never exceed the value of RegulatorMaxActive on the TP. When a new request arrives, the server checks to see if it is already processing the maximum number of requests. If it has reached the limit, it defers processing new requests until the number of active requests drops below the maximum amount.	Number	The number of active requests should ramp up as new users are added and remain constant once the highest load is reached.
	Avg request wait time: Indicates average time for which requests that began active execution in the past second had to wait on the Regulator.	Secs	
	Timed out requests: Indicates the number of requests that timed out in the past second while waiting to become active.	Number	A high value of this measure generally means that there are too many users on the system and response times will be high.
	Waiting requests: Indicates the number of requests that are waiting to begin active execution.	Number	If the value of this measure increases then it indicates that the time that will be spent on processing the transactions will increase as well.

3.2.7 TpRequests Test

This test monitors the requests to a Transaction Processor (TP).

Purpose	Monitors the requests to a Transaction Processor (TP)

Target	The TP server		
Agent deploying this test	Internal agent		
Configurable parameters for this 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 to which the specified **HOST** listens		
Outputs of the test	One set of outputs for the TP server monitored		
Measurements of the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Active requests: Indicates the number of requests that are actively executing.	Number	
	Current requests to the TP: Indicates the current number of requests being handled by this TP.	Number	
	Audit log requests: Indicates the number of current requests for the audit log.	Number	
	Requests to the TP: Indicates the total number of requests processed by this TP during the last measurement period. Does not include Create/Delete session.	Number	
	Requests waiting in the TP: Indicates the number of requests that are waiting to begin active execution.	Number	
	Requests timed out in the TP: Indicates the number of requests that timed out in the past second while waiting to become active.	Number	

	Requests to the TP in the past minute: Indicates the no. of requests were executed by this TP in the last minute.	Number	
	HTML response threads in the TP: Indicates the current HTML Response Threads on the TP.	Number	
	HTML transactions in the TP: Indicates the current HTML transactions on the TP.	Number	A steady number of HTML Transactions on each TP usually is a good sign. If there is a steady decrease, it usually means that the occurrence of a few timeouts or fewer sessions to the TP. An increase in HTML Transactions on another TP may mean that a TP has died and the other TPs are inheriting the increased load.
	Queued transactions for the Host Server: Indicates the number of requests in the HostServer queue waiting to be processed.	Number	
	Avg request wait time in TP: Indicates the average time for which requests that began active execution in the past second had to wait.	Secs	
	Avg response time for the HostServer process: Indicates the average time it took all HostServers under this TP to process their requests in the past minute. Does not include time spent waiting in the HostServer queue.	Secs	If this value increases, it points towards a degraded response time between the TP and the host, or a slowdown in the host itself.
	Avg response time for TP processing: Indicates the average TP response time per request for the past minute.	Secs	This value includes both queueing time and execution time.

	Request execution time: Indicates the average time it took to execute requests which completed in the past minute. This time includes time spent to lookup the session key in the session map. It does not include time spent waiting on the Regulator.	Secs	

3.2.8 TpSessions Test

This test monitors the Transaction Processor sessions.

Purpose	Monitors the Transaction Processor sessions		
Target	The TP server		
Agent deploying this test	Internal agent		
Configurable parameters for this 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 to which the specified **HOST** listens		
Outputs of the test	One set of outputs for the TP server monitored		
Measurements of the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Current sessions to TP: Indicates the total number of Voyager sessions on the TP machine.	Number	Ideally, the value of this measure should rise as the load increases, stabilize at target level and decrease when the session timeout period is exceeded. With multiple TPs, look for balanced values. If this is unbalanced, check the routing policy. If this value continues to increase, there could be a problem with garbage collection of expired sessions.
	New sessions to TP: Indicates the number of new sessions that have been created in this TP in the past minute.	Number	If this measure increases with load, and gains stability once peak load has been reached, it is a good sign. If there is no value, or the value suddenly drops, make sure that the web servers are functioning properly.

	Session garbage collected in TP: Indicates the number of sessions which have been garbage collected on this TP in the past minute. Sessions garbage collected include timed out and signed off sessions.	Number	
	Session GC time: Indicates the average time spent in session garbage collection on this TP in the past minute.	Secs	
	Sessions timed out: Indicates the number of sessions that have timed out in the past minute.	Number	

3.2.9 VlLgger Test

The VLog component, also referred to as the VLogger, is responsible for logging the transactions performed by the online banking consumer to the audit log database. This test keeps tabs on the functioning of the VLogger.

Purpose	Reports statistics relating to the VLogger		
Target	The TP server		
Agent deploying this test	Internal agent		
Configurable parameters for this 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 to which the specified **HOST** listens		
Outputs of the test	One set of outputs for the TP server monitored		
Measurements of the test	**Measurement**	**Measurement Unit**	**Interpretation**
	AuditLog entry data: Indicates the current AuditLog entry named-pipe total bytes to read.	Number	Many log entries could cause logging to slowdown.

	Pending audit log records: Indicates the number of audit log records that have been read but not yet written to the AuditLog database.	Number	Increasing values could indicate a backup in the AuditLog database or a slowdown in the VLOGGER's ability to write data to the database. Consider increasing the value of the AuditConsumers parameter of the VLOGGER, which will give the VLOGGER more threads for writing data to the database.

3.2.10 Vlb Test

The Voyager Load Balancer (VLB), along with the TP, shares the responsibility for session creation and management. This test reveals performance statistics pertaining to the VLB.

Purpose	Reports statistics relating to a VLB		
Target	The TP server		
Agent deploying this test	Internal agent		
Configurable parameters for this 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 to which the specified **HOST** listens		
Outputs of the test	One set of outputs for every VLB monitored		
Measurements of the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Current VLB requests: Indicates the number of requests executing in this VLB instance. A request consists of a call to the TP to execute but does not include CreateSession or DeleteSession Calls. It does include successful and failed calls.	Number	A sudden increase or a spike in this value is acceptable, but watch for any value that is consistently high. A consistently high or increasing value of this metric is an indicator of a problem. This value should correlate with the value of the Request_execution_time measure.
	Create session calls: Indicates the total number of CreateSession calls processed by all VLB's on this computer during the last measurement period. This includes both successful and failed calls.	Number	

	Create session failures: Indicates the total number of failed Create Session calls by all VLB's during the last measurement period. "Failures" means an external error like an RPC network error.	Number	Watch for any increase in this value over time.
	Execute failures: Indicates the total number of failed Execute calls by all VLBs on this computer during the last measurement period. This value does not include Create/DeleteSession.	Number	
	Request count: Indicates the total number of requests processed by all VLB's on this computer during the last measurement period. This value does not include Create/Delete session.	Number	
	VLB request execution time: Indicates the time it took to execute requests which completed in the past minute.	Secs	After peak load this value should be stable, but watch for any value that is consistently high and increasing numbers after peak load. This value should correlate with the value of the Current_requests measure.

3.2.11 VlbDatabases Test

This test reports statistics relating to a VLB's database.

Purpose	Reports statistics relating to a VLB's database
Target	The TP server
Agent deploying this test	Internal agent
Configurable parameters for this 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 to which the specified **HOST** listens
Outputs of the test	One set of outputs for every VLB database instance monitored

Measurements of the test	Measurement	Measurement Unit	Interpretation
	Current VLB database requests: Indicates the number of database requests currently executing in this VLBinstance.	Number	A consistently large value indicates a bottleneck at the database tier
	VLB database request execution time: Indicates the average time it took to execute database requests which completed in the past minute. Response time includes the time for parsing the request, to get the cached stored procedure (SP)information, executing the SAP, and the reading/parsing of the SP results.	Secs	A very high value of this measure indicates a performance bottleneck that requires further investigation.
	VLB database requests handled: Indicates the number of requests to the database that have begun to execute in the past minute.	Number	
	VLB SP execution time: Indicates the average time the ADO.NET took to execute a stored procedure in the past minute.	Secs	

3.3 The Host Service Layer

Voyager processes several different types of operations, some of which interact with the financial institution's host. Voyager's host servers handle these operations. Examples of these types of operations are Balance, History, or any operation that requires account data. This layer monitors the activities of these host servers. The tests associated with this layer have been depicted by Figure 3.4.

Figure 3.4: The tests associated with the Host Service layer

3.3.1 HostServer Test

This test monitors the host server functions and reports critical metrics relating to the same.

Purpose	Monitors the activities of host servers		
Target	The TP server		
Agent deploying this test	Internal agent		
Configurable parameters for this 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 to which the specified **HOST** listens		
Outputs of the test	One set of outputs for every host server monitored		
Measurements of the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Avg host server time: Indicates the average time taken by the backend host to execute the transaction.	Secs	
	Avg transaction time: Indicates the average time taken b the transaction executed in the past minute.	Secs	
	Longest transaction time: Indicates the longest time it took to process a transaction for the past hour.	Secs	

3.3.2 HSConnector Test

This test reports metrics pertaining to the host server connectors.

Purpose	Reports metrics pertaining to the host server connectors		
Target	The TP server		
Agent deploying this test	Internal agent		
Configurable parameters for this 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 to which the specified **HOST** listens		
Outputs of the test	One set of outputs for every host server monitored		
Measurements of the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Avg transaction time: Indicates the time taken for executing a transaction.	Secs	
	Host request time: Indicates the time taken by a host server to place a request.	Secs	
	Host server time: Indicates the time taken by a host server to complete processing a transaction.	Secs	
	Host servers: Indicates the number of transactions inside Voyager's host server.	Number	
	Current executions: Indicates the number of transactions currently executing on a host server.	Number	
	Request rate: Indicates the rate at which requests were placed by the host server.	Reqs/Sec	

	Wire rate: Indicates the rate at which the requests were on wire.	Reqs/Sec	
	Transaction processing time: Indicates the time taken by a host server to process a transaction.	Secs	
	Transaction rate: Indicates the rate at which the host server processed transactions.	Trans/Sec	

3.3.3 HSTrans Test

This test reports information relating to the transactions executing on a host server.

Purpose	Reports information relating to the transactions executing on a host server		
Target	The TP server		
Agent deploying this test	Internal agent		
Configurable parameters for this 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 to which the specified **HOST** listens		
Outputs of the test	One set of outputs for every host server monitored		
Measurements of the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Host server request rate: Indicates the rate at which the host server placed requests.	Reqs/Sec	
	Host time: Indicates the time taken by a host server to service a request.	Secs	
	Executions count: Indicates the number of executions that occurred on the host server.	Number	

	Transaction processing time: Indicates the time taken by the host server to process transactions.	Secs	A very high value indicates an host server overload.

Chapter 4

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 **Corillian's Voyager**. 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.