eG Innovations

The IIS Administrator's Troubleshooting Guide

How to become an expert in IIS monitoring and troubleshooting

An eG Innovations Technical White Paper

This eBook is aimed at the IIS hands-on administrator or architect looking to optimize their monitoring strategy and implement best practices to ensure their web site and application delivery is designed and managed to ensure high-availability and optimal user experience

We will cover the technicalities of the key metrics every IIS administrator should be aware of when evaluating third party solutions or developing your own tooling and strategy in-house. This guide will also help you become an expert on proactive monitoring and management of IIS to improve security and detect the early warning signs of malicious attacks on your infrastructure and services.

❖ What is Microsoft IIS?

Microsoft Internet Information Services (IIS, formerly known as Internet Information Server) is an extensible web server software created by Microsoft for use with the Windows family. IIS supports various protocols, including HTTP, HTTP/2, HTTPS, FTP, FTPS, SMTP, and NNTP. Key benefits include componentization, extensibility and ASP.NET integration.

IIS has numerous extensibility features. Swappable interfaces, such as ISAPI and FastCGI make it possible to use IIS with a variety of backend technologies, from micro-frameworks such as Flask to runtimes such as Node.js, along with Windows technologies such as ASP.NET.

Many organizations and businesses rely on IIS for their core web applications, so ensuring that these applications are performing well is critical. In this eBook, we will review the key metrics that must be tracked to ensure peak performance and high-availability of Microsoft IIS-based web applications.

❖ Microsoft IIS Architecture

To understand the key metrics to monitor on an IIS server, it is important to first understand the architecture of an IIS server. IIS has three main components – HTTP.sys, Worker Processes (Application Pools), and IIS Admin Services.

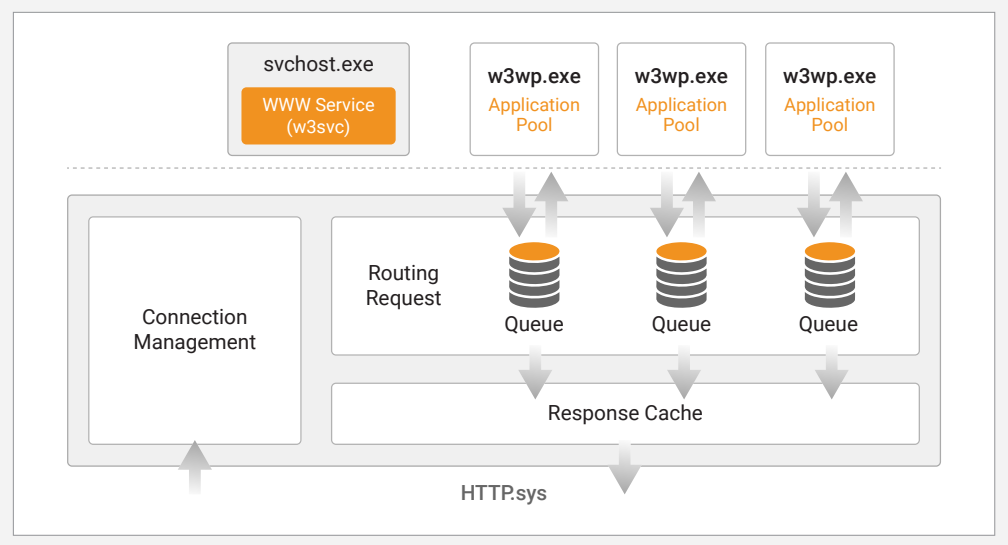


Figure 1: Architecture of a Microsoft IIS Server

HTTP.sys :

This is a kernel component of IIS. That is, this is not a part of user mode processes. A kernel component never uses any virtual memory addresses of user processes. It is isolated and separated from user processes.

HTTP.sys listens for HTTP and HTTPS requests and validates each one before passing it to a worker process. If no worker process is available to handle a request, HTTP.sys places the request in a kernel-mode queue.

HTTP.sys has several important roles – (client) connection management, routing requests from browsers, requesting pre-processing and security filtering, and managing the response cache.

When it routes requests, HTTP.sys dispatches requests to the correct 'Application Pool Queue' for each worker process.

Worker Processes:

W3WP (WWW Worker Process) conducts the main work of a web server: handling client requests and serving responses. IIS can handle requests with multiple worker processes at a time (depending on your configuration), each of which runs as an executable w3wp.exe.

Static content, such as HTML/GIF/JPG files are transmitted by it when requests to these files are received. The worker process is also responsible for generating dynamic content.

For example, by supporting ASP/ASP.NET applications. Therefore, the status of W3WP processes is critical for the performance and stability of web applications, or websites hosted on an IIS server.

IIS Admin Service:

The key component here is the World Wide Web Publishing Service (WWW Service) which runs as a process – svchost.exe. This service passes IIS configuration settings to HTTP.sys and collects performance metrics.

IIS uses the concept of **application pools** to ensure reliability and manageability. An application pool is a group of one or more worker processes, configured with common settings that serve requests to one or more applications that are assigned to that application pool.

Because application pools allow a set of web applications to share one or more similarly configured worker processes, they provide a convenient way to isolate a set of web applications from other web applications. Process boundaries separate each worker process. Therefore, application problems in one application pool do not affect websites or applications in other application pools.

Figure 2 shows different configurations of applications, worker processes, and application pools. While (A) shows one worker process supporting one web application, (B) shows one worker process supporting multiple web applications. (C) shows a configuration of multiple worker processes supporting one or more web applications. This configuration is referred to as a **web garden.**

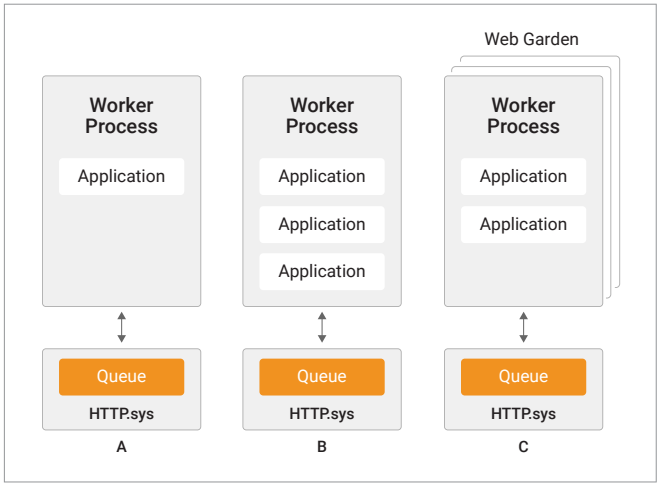


Figure 2: Different configurations of applications, worker processes, and application pools

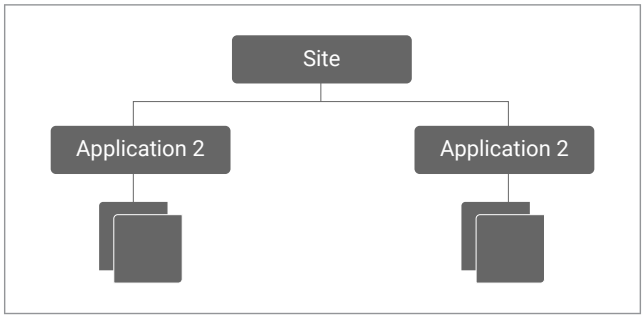


Figure 3: Mapping of application pools, applications, and sites in an IIS server

Routing of a request to an application pool by HTTP.sys is based on the URL of the request. The main component of a URL is a site – which maps to an IP address, a TCP port, and a host header. A site can have multiple applications. Each application can map to an application pool.

❖ Key Challenges in Monitoring IIS Web Servers

- A single IIS web server can host multiple websites. When there are excessive requests to one website, this can affect the performance of other websites.
- Application pools form an important part of an IIS server system. An individual application pool could have many IIS worker processes associated with it. Websites are mapped to different application pools and share the pool's resources. As resource sharing happens across websites, a memory leak, or a runaway thread during processing of one website can affect other websites as well. Identifying which website is the cause of a problem can be a challenge.

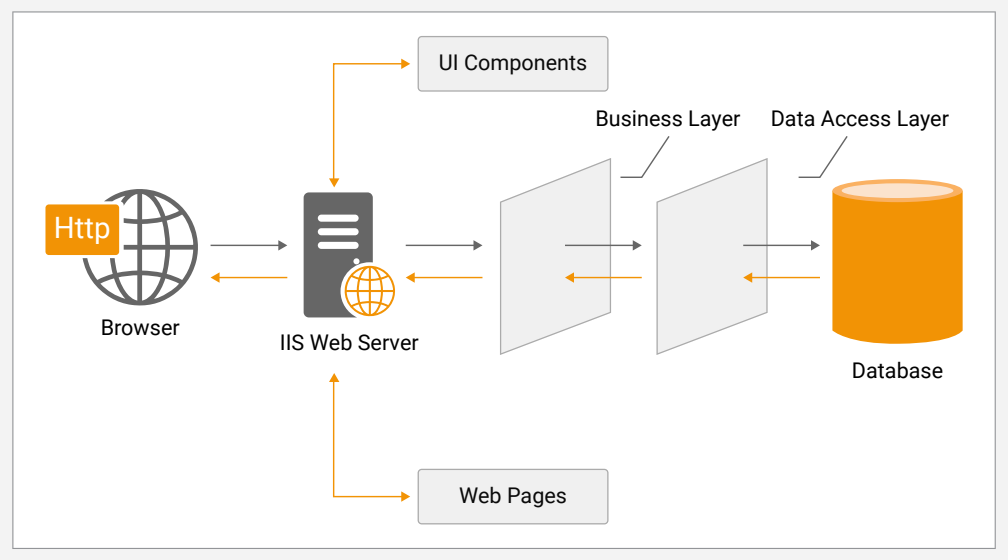


Figure 4: Web applications are multi-tiered. Microsoft IIS is the front-end. When a problem occurs, identifying the root-cause takes time as slowness in one tier (e.g., the database) affects all other tiers (e.g., IIS)

- As web applications involve multiple processing tiers, a slowdown in any backend tier will often manifest in a bottleneck at the web tier. Hence, identifying when a problem occurs, where the root cause is, is often challenging: is it the web server, or the database, or the business logic, or the supporting infrastructure?

- Performance issues may also be in the application code. Visibility into the malfunctioning portions of the application code is critical for development and DevOps teams to detect and resolve issues quickly.

❖ Understanding IIS Usage Within the Topology of IT Application and Infrastructure Architectures

Before considering the key metrics, it is wise to ensure you have a thorough understanding of how IIS components are being used within your organization's IT infrastructure and their interdependencies on other components. Many monitoring tools will auto-discover these inter-dependencies, and some can produce visual topology maps that clarify not only the end-to-end IT infrastructure but also the end-to-end business service.

Here are a few examples of real-world IIS topology maps which your monitoring strategy may need to support.

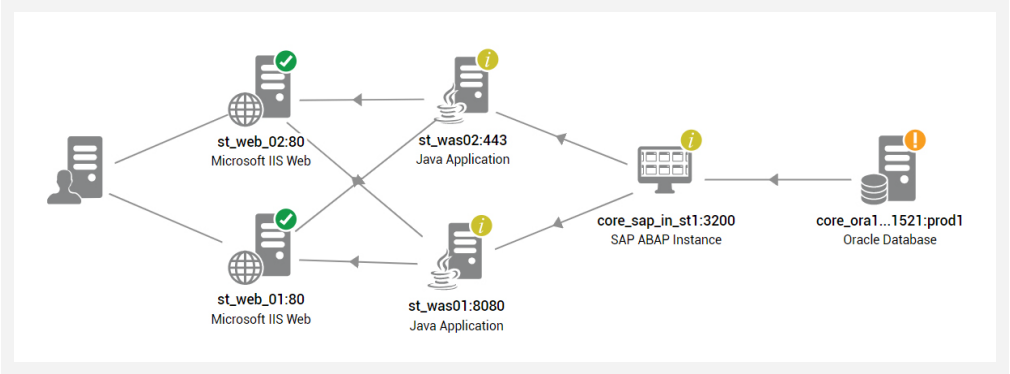


Figure 5: IIS deployments may include redundancy for scale and failover, and depend on other components such as Oracle Databases or SAP services

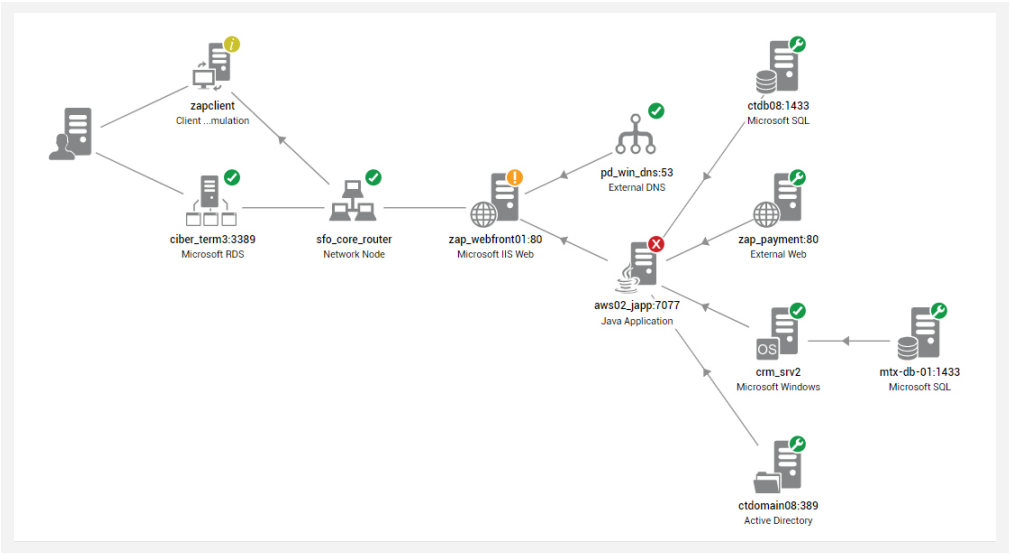


Figure 6: Successful operation of IIS may involve dependencies on Active Directory for authentication, external DNS services and backend SQL servers

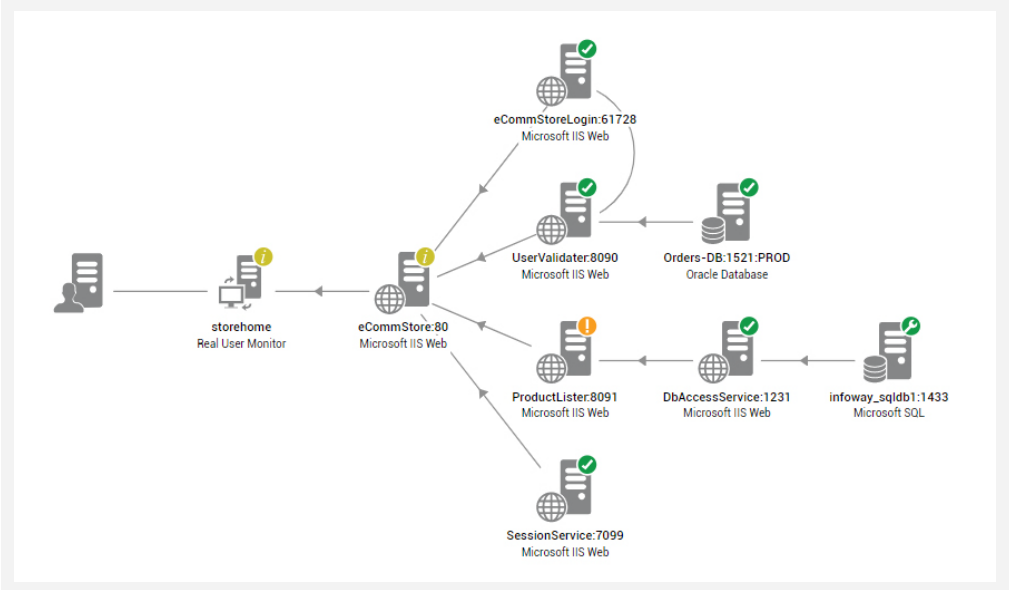


Figure 7: Many IIS components may be involved in delivering a web service such as an online E-Commerce store

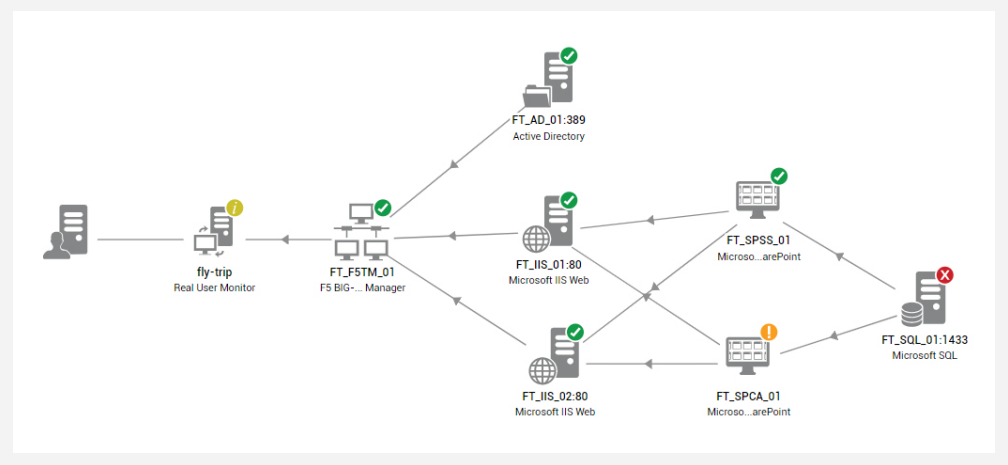


Figure 8: An online booking system may involve components such as an F5 load-balancer, Active Directory, Microsoft SharePoint and SQL servers and include redundancy for failover capacity

❖ Top 10 Key Metrics to Monitor for an IIS Server

Having looked at the architecture of Microsoft IIS, let us now turn our attention to what are the key metrics to monitor regarding a Microsoft IIS server. Since IIS runs as an application on a Windows operating system, it is essential to monitor all the key operating system parameters. CPU utilization, memory utilization, activity of each of the disk drives, usage of handles in the operating system, page file usage, etc., are just a few of the key metrics to be tracked at the OS layer. To *ensure that IIS is working,* it is important to monitor the status of the World Wide Web Publishing service.

#1 Monitor IIS Availability and Response Time:

Sometimes, the service could be up, but the IIS server may not be responding to incoming requests. *Synthetic monitoring* is recommended to ensure that IIS is monitored at all times. Robot users simulate accessing IIS resources from multiple locations, regularly and when real users may not be using resources, to proactively detect slowdowns and availability issues before real users become aware of problems.

Synthetic monitoring involves protocol emulation – either HTTP, FTP or NTP requests are issued to IIS and the response is checked to ensure that the service is responding. The time taken for the response is compared with prior baselines to determine whether IIS is performing within expectations.

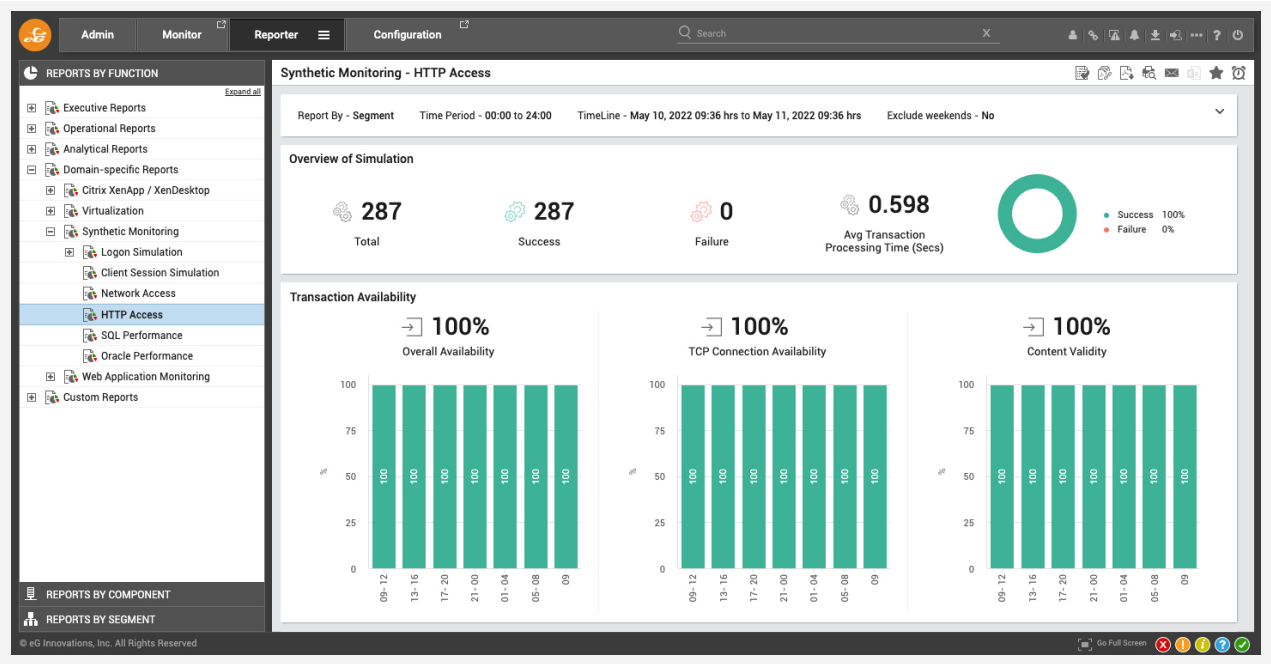


Figure 9: Performance of a web service measured by synthetic monitoring

#2 Monitor the Workload on your IIS Server and Individual Websites:

IIS response times could degrade because of excessive load on it. Therefore, it is important to track the workload of the IIS server – how many requests/second is it processing?

Comparing the request rate across all IIS servers in your server farm can help you determine whether your workload is being balanced across all the servers. Workload imbalance can result in performance bottlenecks and must be addressed quickly.

When an IIS server supports multiple websites, it is important to know which site is receiving the most requests. Along with load, it is essential to track the responses from the server. Is the server generating any error responses – 400 errors? 500 errors? Which site is seeing the most errors?

Related Reading: *Find out how to answer the question "why is my web application is slow?"*

#3 Get Granular Details on Website Performance:

Break down responses with 400 and 500 status codes further, by website, to identify the cause of error responses from each website to clients. For example, which URLs are causing not found errors, which ones have logon failures, are there URLs seeing authorization failures, did the server respond with busy errors and at what times, etc. Report average response time for accesses to each website and identify the slowest URLs by response time during each measurement period. This granular level of detail per website can be obtained by analyzing the IIS server's access log in real time.

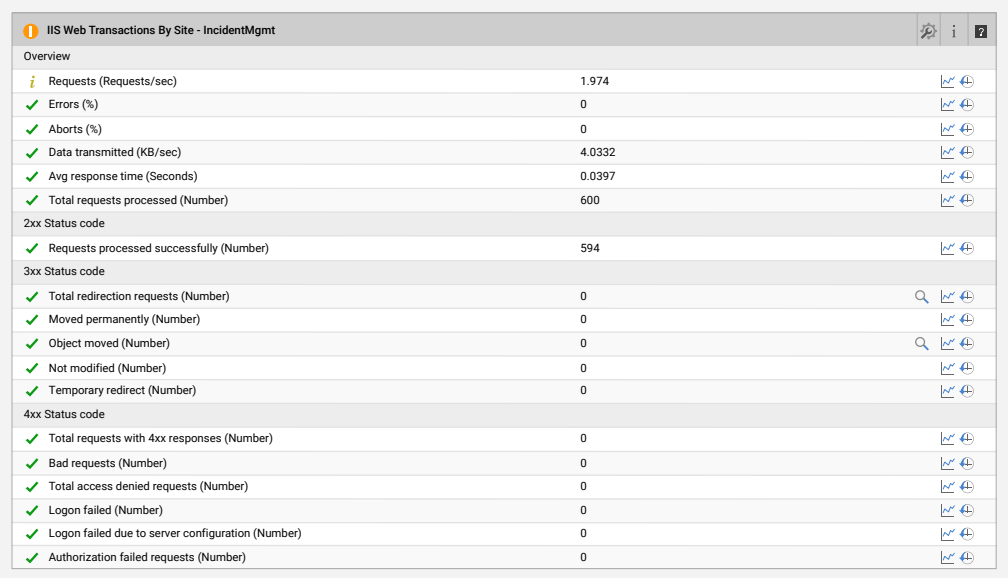


Figure 10: Overall performance of an IIS website

#4 Snapshot Requests Currently Executing for each Website and Identify Requests that are being Processed Slowly:

Track the currently executing requests for each website. Ideally, this number should be as small as possible. If the number of currently executing requests is growing for a website, this means that requests are piling up and could lead to higher latencies and request queueing.

Monitor the processing time for each currently executing request and alert if the processing time exceeds a preset limit, highlighting that a specific request has been executing for a long time. The URL being accessed is highlighted. If the hang is being caused by a specific ASP.NET controller or page, the module will indicate if it is IsapiModule (classic mode) or ManagedPipelineHandler (integrated mode). Typically, this analysis points to specific URLs that may be causing slowness when accessing a website.

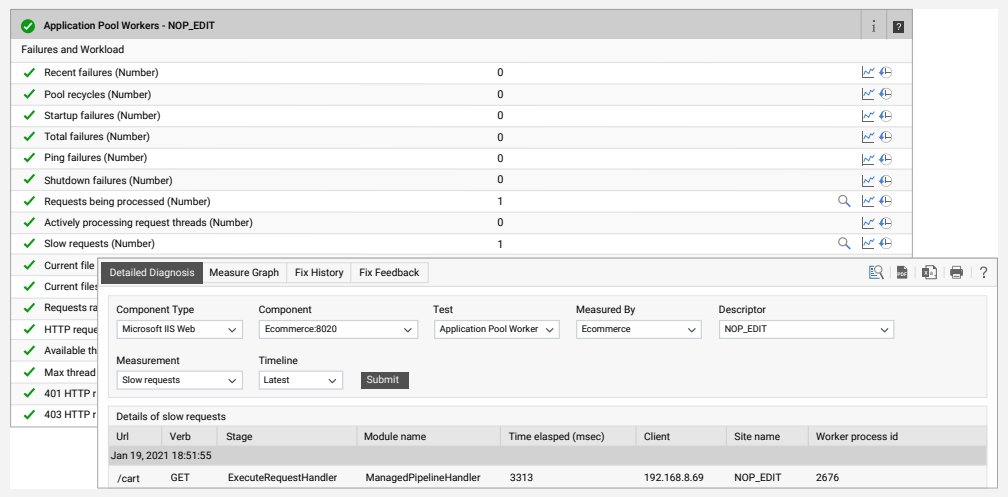


Figure 11: Identifying slow requests on an IIS web server

Now, let us go deeper into the IIS components that could cause slowness in request processing. Most of the requests to an IIS server are processed by the .NET runtime. The figure below shows how this processing occurs. A request is first received by HTTP.sys and added in the queue of the corresponding application pool at kernel level. One IIS worker thread takes the request from the queue and passes it to the ASP.NET queue after its processing.

The request may get returned from IIS itself if it is not an ASP.NET request. A thread from the CLR (Common Language Runtime) thread pool gets assigned for processing the request. ASP.NET does not create any thread or own any thread pool to handle the request; instead, it gets a thread from the CLR thread pool.

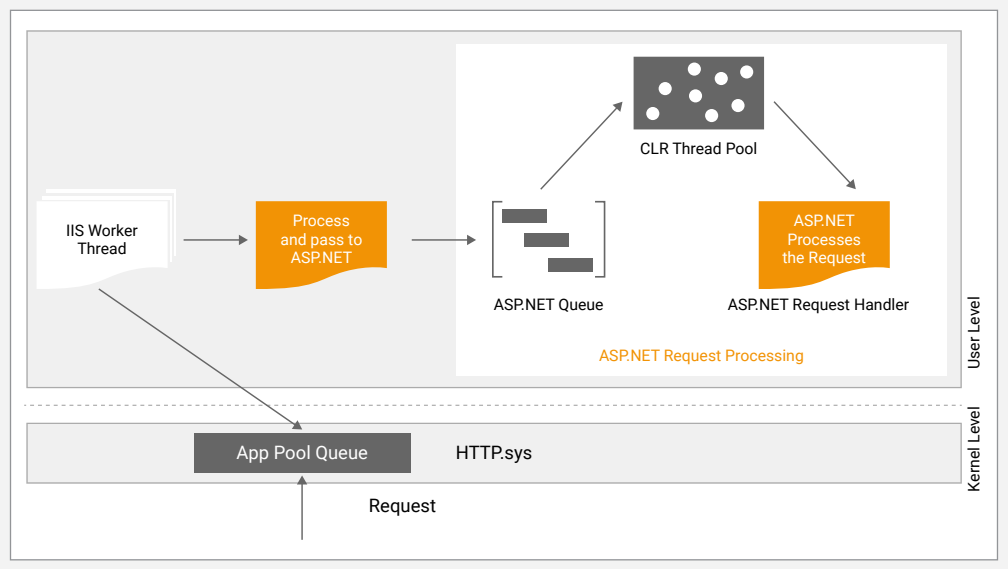


Figure 12: How ASP.NET requests are received and processed within an IIS web server

#5 Look out for ASP.NET Exceptions in the Application Event Log:

If a request to an ASP.NET application generates a 500 error, it is most likely that the error is due to an unhandled ASP.NET exception. To find it, go to the Application EventLog and look for Warning events from the appropriate ASP.NET version. The details of the warning will point you to your application logic that could be causing the exception.

Related Reading:

- *Best Practices to Monitor Microsoft IIS*
- *How to Monitor the Performance of Microsoft .NET Applications*

#6 Monitor .NET Processing Efficiency:

For each .NET application, track the following metrics:

Metric	Interpretation
Requests in the Application Queue	If this number is high, your server may not be able to handle requests fast enough.
Requests Executing Currently	A sustained high value is an indicator of high load on the .NET runtime.
Request Execution Time	If this value increases, it may be indicative of a problem with the application logic or in the .NET runtime.
Request Waiting Time	This is the time period for which the last request was held in queue before it was taken for processing by the .NET runtime. For good performance, this value should be close to 0. A value closer to 1 sec is indicative of a .NET processing bottleneck on the IIS server.

In addition to the above metrics, check for .NET processing errors. Unhandled runtime errors are indicative of exceptions that would have been visible to users and hence must be tracked. Total errors, including errors during compilations, pre-processing, and execution should also be monitored to determine anomalous situations.

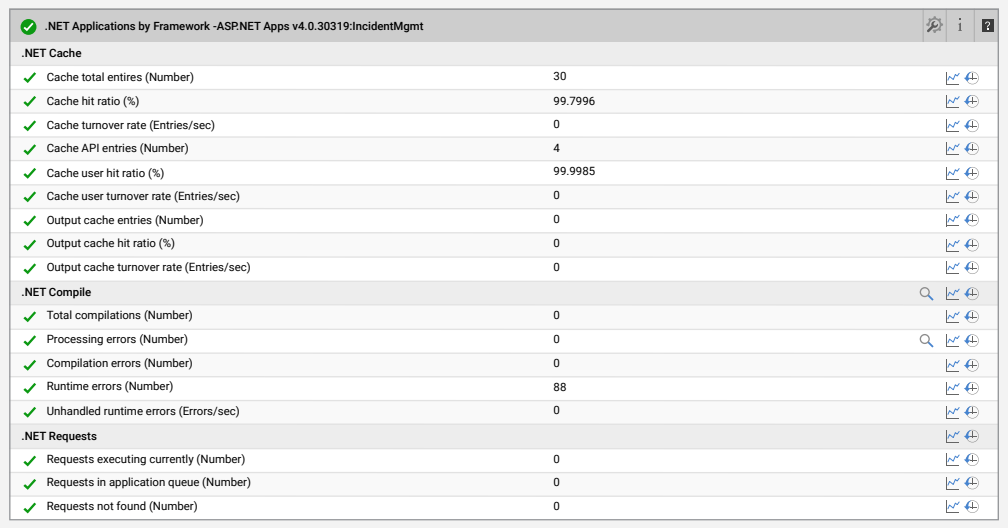
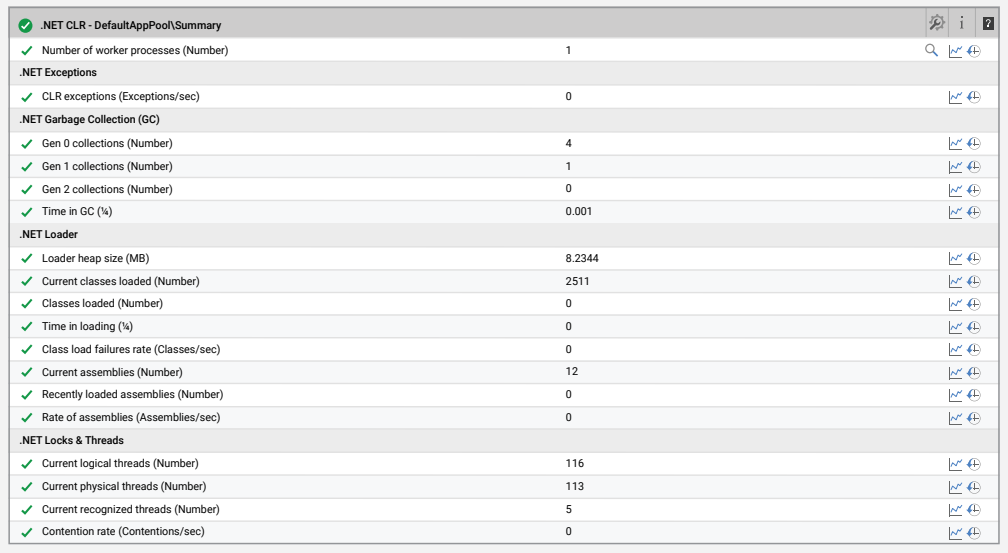


Figure 13: Monitoring .NET processing performance

#7 Monitor .NET CLR Engine:

Each worker process has a .NET CLR. One of the key performance measures of the .NET CLR engine is the percentage of time spent in garbage collection. If a process spends more than 5% of time in garbage collection, this may be indicative of object allocation problems in your application.

Threads in the CLR can acquire and release locks. When multiple threads execute in parallel, this can create contention. Track the number of contentions for locks in the CLR. Also track the queue length – i.e., the number of threads that are currently waiting to acquire a managed lock in the application. These metrics can be indicative of multi-thread contention bottlenecks.



.NET CLR - DefaultAppPool\Summary		
Number of worker processes (Number)	1	
.NET Exceptions		
CLR exceptions (Exceptions/sec)	0	
.NET Garbage Collection (GC)		
Gen 0 collections (Number)	4	
Gen 1 collections (Number)	1	
Gen 2 collections (Number)	0	
Time in GC (%)	0.001	
.NET Loader		
Loader heap size (MB)	8.2344	
Current classes loaded (Number)	2511	
Classes loaded (Number)	0	
Time in loading (%)	0	
Class load failures rate (Classes/sec)	0	
Current assemblies (Number)	12	
Recently loaded assemblies (Number)	0	
Rate of assemblies (Assemblies/sec)	0	
.NET Locks & Threads		
Current logical threads (Number)	116	
Current physical threads (Number)	113	
Current recognized threads (Number)	5	
Contention rate (Contentions/sec)	0	

Figure 14: Monitoring the .NET CLR engine

#8 Monitor Application Pools and Worker Processes:

As we have seen, application pools and worker processes are integral to processing of requests by IIS. Hence, it is essential to monitor them. Key metrics to collect regarding application pools include:

- Current state of the application pool; alert if a pool is inadvertently stopped.
- Track uptime of each application pool and detect if any unusual restarts have happened recently.
- Monitor the number of worker processes running in each pool and the resources used – CPU, memory, IOPS, OS handles used, etc. Comparing the resource utilization across application pools can reveal which application pool may have resource usage issues.
- Report rate of requests to each application pool, failures in processing in the application pool and if there are excessive running threads in any pool.

#9 Monitor Windows Process Activation Service (WAS):

The Windows Process Activation Service (WAS) of IIS is the key component that provides process model and configuration features to web applications and Web Services. WAS' major task is to manage application pools. For each application pool, track the active listener and protocol channels. Time taken to respond to WAS messages is another important measure of the health of the worker processes in each application pool.

#10 Track HTTP.sys Performance:

Requests from clients have to pass through HTTP.sys, which handles the TCP connections, before they reach the IIS worker processes for processing.

HTTP.sys also establishes the SSL/TLS SChannel used in HTTPS. Request validation happens in HTTP.sys; for instance, too large an HTTP request header may cause rejection. And with certain configurations, HTTP.sys also performs the Windows Authentication of HTTP requests with Kerberos.

All these happen before relaying requests to IIS; any failure here would prevent an HTTP request from reaching IIS. If requests are not getting to IIS, the HTTPERR log file may reveal why requests are rejected by HTTP.sys. This will not show up in the web server's access logs. Hence, looking at the access logs alone is not sufficient.

HTTP.sys rejections can happen for several reasons: a request violated the HTTP protocol (client saw HTTP 400: Bad Request) or there was a WAS/application pool failure (client saw HTTP 503: Service Unavailable). Monitoring of the HTTPERR log reveals many such error conditions.

HTTP Errors	
Application offline (Number)	0
Busy application pool processes (Number)	0
Application shutdown (Number)	0
Bad requests (Number)	0
Connections reset by client (Number)	0
Connections abandoned by application pool (Number)	0
Connections abandoned by request queue (Number)	0
Connections dropped (Number)	0
Connections limit has reached (Number)	0
Connections refused (Number)	0
Internal errors (Number)	0
Application request queue limit reached (Number)	0
Connections expired (Number)	0
URL processing errors (Number)	0

Figure 15: Tracking any HTTP.sys errors

Microsoft also recommends monitoring for spikes in Timer_ConnectionIdle messages, which appear when a client's connection has reached its keep-alive timeout. An increase in the number of Timer_ConnectionIdle messages may be indicative of DoS (Denial of Service) attacks to your server. These messages can also occur if several users are having issues with connectivity. Hence, it is important to track these messages, but additional investigation is needed to diagnose if this is a connectivity issue or an attack.

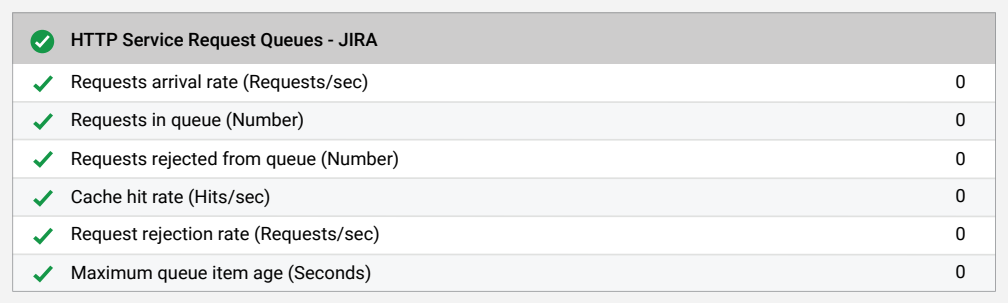


Figure 16: Monitoring request queues at the HTTP.sys layer. An increase in request queue length is an indicator of potential request rejections in future

Additionally, it is important to monitor the HTTP.sys application pool queue. Requests are first queued in the HTTP.sys' application pool queue and IIS worker processes have to dequeue these requests. When the worker processes are slow to dequeue, requests accumulate in this queue. The application pool's configured queueLength attribute determines how many requests are stored in the queue. By default, the queueLength is 1000.

When this limit is reached, HTTP.sys returns a 503 Service Unavailable response. Hence, it is critical to monitor the requests in this queue. At the same time, also track the request rejection rate, which is another sign of an IIS bottleneck. Another important attribute is the age of oldest requests in queue (age of the last request in the queue). This can give an idea of the queueing delay at the HTTP.sys layer.

❖ Comprehensive Microsoft IIS Performance Monitoring with eG Enterprise

As you have read in this eBook so far, there are hundreds of metrics to track regarding the performance of a Microsoft IIS Server. eG Enterprise automates the collection of data on IIS performance and its analysis to provide root-cause diagnosis of issues and automated alerting of performance and availability issues and errors. eG Enterprise simplifies and automates all aspects of *monitoring of Microsoft IIS* by:

- Providing auto-discovered rich visual topological views of IIS components within the end-to-end application and infrastructure landscape
- Available to deploy on-premises, in cloud or via fully managed SaaS (Software as a Service) in multiple geo-locations to comply with all your data protection, compliance and regulatory requirements
- Automated thresholding, alerting and anomaly detection powered by an industry leading AIOps (Artificial Intelligence for Operations) platform leveraging proven, industry leading machine learning technologies

- Out-of-the box dashboards and reports for the whole organization including front line L1/L2 help desk support staff, system administrators and finance and business executives

- eG Enterprise agents have built in capability to collect all the KPIs (Key Performance Indicators) regarding Microsoft IIS on an on-going basis. The profile of Microsoft IIS, built into the solution, dictates what metrics to collect and at what frequency.

- Thresholds and baselines configured in the system indicate the expected range of each metric. eG Enterprise also auto-baselines key usage metrics, so if there is unusual activity or usage or queueing in any of the IIS components, eG Enterprise can proactively alert administrators to this situation.

- Rather than provide a ton of measurements on a dashboard, eG Enterprise groups metrics by functional layers, so it is very simple to understand which layers are performing well and which are not. For instance, is an IIS performance issue due to insufficient hardware/OS configuration, or due to errors in the application stack, or due to a memory leak in one of the application pools?

- Performance metrics collected by eG Enterprise can be analyzed to determine trends for post-mortem diagnosis of problems and for intelligent capacity planning.

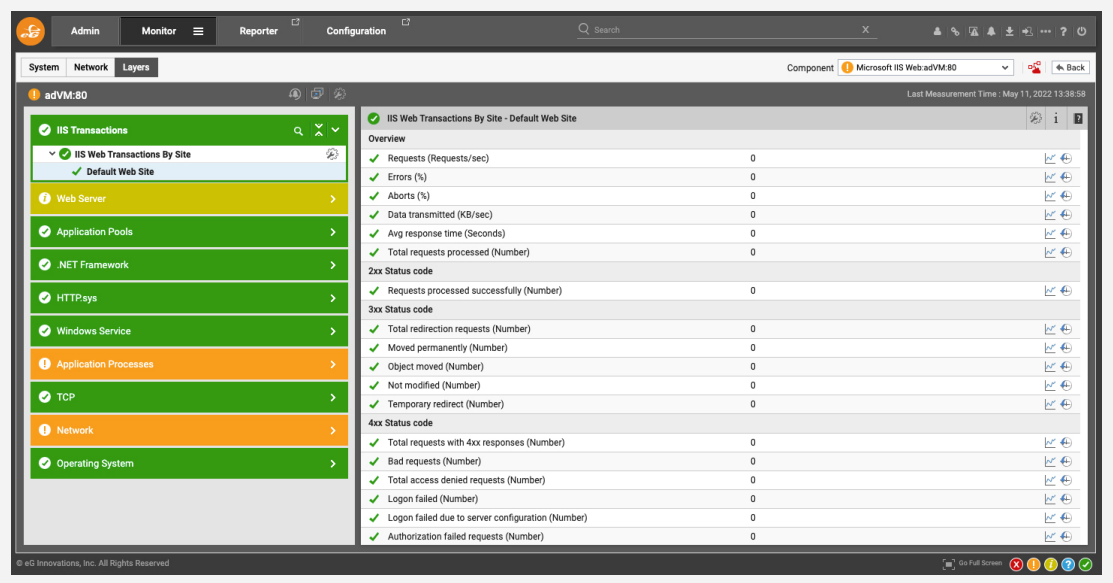


Figure 17: Comprehensive layer model of an IIS web server in eG Enterprise

❖ Learn More

- Microsoft IIS Monitoring | eG Innovations
- Supported technologies : IT Monitoring Technologies & Supported Platforms | eG Innovations

❖ Next Steps

- ✉ | To contact eG Innovations sales team : sales@eginnovations.com
- 🌐 | Get a free trial of eG Enterprise : www.eginnovations.com/FreeTrial
- ✉ | For support queries and feature requests : support@eginnovations.com

❖ About eG Innovations

eG Innovations provides the world's leading enterprise-class performance management solution that enables organizations to reliably deliver mission-critical business services across complex cloud, virtual, and physical IT 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, eG Innovations helps companies proactively discover, instantly diagnose, and rapidly resolve even the most challenging performance and user experience issues.

eG Innovations' award-winning solutions are trusted by the world's most demanding companies to ensure end user productivity, deliver return on transformational IT investments, and keep business services up and running. Customers include Anthem, Humana, Staples, T-Mobile, Cox Communications, eBay, Denver Health, AXA, Aviva, Southern California Edison, Samsung, and many more.
To learn more visit www.eginnovations.com

❖ References

Overview : The Official Microsoft IIS Site

Internet Information Services - Wikipedia