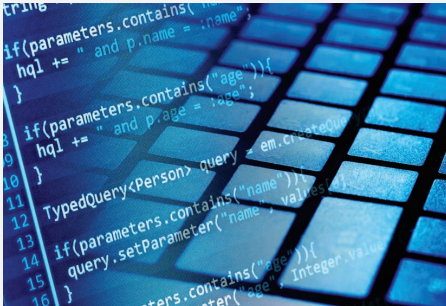


# Monitoring the Health & Performance of Java Virtual Machines (JVM)



## Key Benefits

- ✓ **Proactively detect and resolve Java application problems** to ensure high service uptime and business continuity
- ✓ **Troubleshoot faster:** Real-time alerts pinpoint the exact line of code that is impacting Java applications
- ✓ **In-depth analytics** enable architects to optimize Java applications to scale and support additional users
- ✓ **Single pane of glass to monitor everything Java**—from application server, JVM and database, all the way down to server and storage infrastructure
- ✓ **Eliminate finger-pointing:** Easily determine if it is a server, network, virtualization, or code-level issue

“We can now quickly identify root causes of incidents, resolving them before users are impacted. Automatic prioritization and categorization of alerts helps us better focus on the important issues and prioritize our resources accordingly.”

**Andrew Gowlett**  
C5 Alliance

## JVM: The Cornerstone of Java Enterprise Applications

The Java Virtual Machine (JVM) forms the core of the Java application architecture. It plays the crucial role of interpreting and translating Java byte code into operations on the host platform. Since Java middleware (application servers such as Tomcat, JBoss EAP, JBoss AS, WildFly, WebSphere, and WebLogic) runs on the JVM, a performance issue at the JVM level has a major impact on business services supported by it.

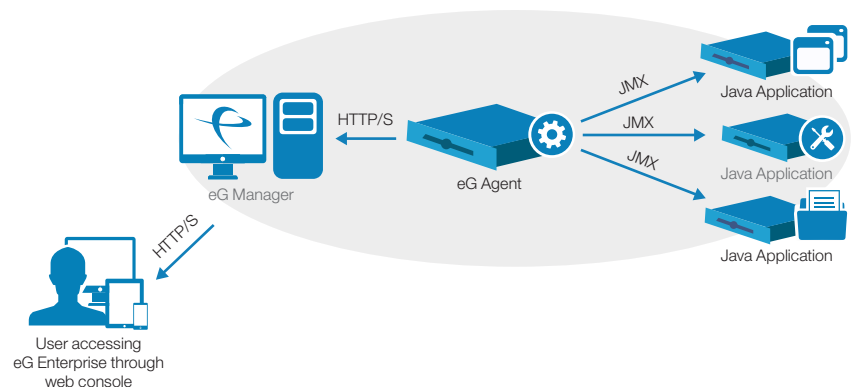
Monitoring of the JVM must be an integral part of any Java application performance monitoring strategy. IT Ops and DevOps teams use JVM performance metrics to troubleshoot server-side bottlenecks. Developers and architects can also benefit from JVM monitoring by uncovering code-level issues.

## Key JVM Performance Questions to Answer

- Is there any runaway thread hogging the CPU? Which line of code is it executing, in which class, and which method?
- Is the JVM heap and non-heap memory sized correctly?
- Are there any out-of-memory-exceptions or memory leaks?
- When does garbage collection happen, and how much memory is freed up each time?
- Are there any thread deadlocks happening that are causing application processing to be hung?

## End-to-End JVM Monitoring with eG Enterprise

eG Enterprise is a comprehensive performance monitoring, diagnosis and reporting solution for Java application infrastructures. Using JMX, eG Enterprise monitors JVMs (such as JRockit, OpenJDK, Oracle, IBM, etc.) in depth, tracking key performance indicators covering all aspects of a JVM's performance.



eG Enterprise auto-correlates JVM performance with other infrastructure metrics (server, virtualization, storage, etc.), and provides actionable insights in the form of proactive alerts and intuitive dashboards for rapid problem diagnosis.

# Key Capabilities of eG Enterprise for JVM Monitoring

## CPU & Memory Management

- Track JVM CPU utilization and easily identify high CPU consuming threads
- Monitor heap and non-heap memory usage (growing memory utilization can indicate memory leak and out-of-memory errors)
- Monitor JVM uptime statistics and whether restarts are occurring unexpectedly
- Identify class names that are memory leak suspects

## Thread Monitoring & Garbage Collection

- Uncover deadlocked and blocked threads, and easily isolate the Java class, method or object that is causing these issues
- In a single click, look up the stack trace to pinpoint the exact line of code encountering an error or exception. See real-time as well as historical data.
- Track time taken for garbage collection (GC). Fine-tune GC settings based on GC activity analysis.

The screenshot displays the eG Enterprise monitoring interface for a Java application. On the left, a navigation pane shows 'JVM Engine' and 'JVM Internals'. The main area shows 'JVM Threads' with a table of thread statistics:

Thread Type	Count
Total threads (Number)	65
Runnable threads (Number)	46
Blocked threads (Number)	1
Waiting threads (Number)	5
Timed waiting threads (Number)	13
Low CPU threads (Number)	64
Medium CPU threads (Number)	0
High CPU threads (Number)	1
Peak threads (Number)	68

A red callout box labeled 'Detecting a High CPU Thread' points to the 'High CPU threads' row. Below this, a 'Detailed Diagnosis' window shows 'ZapConnector' thread ID 23 in a 'TIMED\_WAITING' state. A red callout box labeled 'Isolating the Runaway Thread' points to this entry. A 'THREAD DIAGNOSIS' window shows a table of thread metrics for 'ZapConnector' and a stack trace. A red callout box labeled 'Understanding the Code Stack Trace' points to the stack trace. Another red callout box labeled 'Pinpointing the Problematic Code' points to a specific line in the code snippet:

```
213 public void createLogic()  
214 {  
215     long count = 0L;  
216     while(!finish)  
217     {  
218         try  
219         {  
220             if(count > 1500000)  
221             {  
222                 count = 0;  
223                 Thread.sleep(1);  
224             }  
225             count ++;  
226         }  
227         catch(Exception ex)  
228         {  
229             ex.printStackTrace();  
230         }  
231     }  
232 }  
233 }
```

## Get Visibility Beyond the JVM

JVM monitoring is necessary, but is not sufficient for delivering great Java application performance. The performance of the server infrastructure (OS, hardware, virtualization platform, storage, etc.), the web container hosting the application, and the database server used affect Java application performance. With eG Enterprise, you get:

- **Business Transaction Monitoring:** Trace business transactions across all of your application tiers. Using a tag-and-follow approach, eG Enterprise reports the processing time for a request at each tier and highlights which tier is causing slowdowns and why: is it Java processing (which method?), database queries (which ones?), web service, or calls to third-party applications?
- **Deep Insight into Java Containers:** Monitor thread pools, request queues, connection pools and other application processing functions supported by Java containers (such as Tomcat, JBoss EAP, JBoss AS, WildFly, WebSphere, and WebLogic) to identify application bottlenecks.
- **Unified and Correlated IT Infrastructure Monitoring:** Automatically discover dependencies between infrastructure tiers, analyze performance insights, and correlate them with Java application performance and end-user experience to isolate the root cause of performance issues across the IT environment.

## About eG Innovations

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

CONTACT US: [sales@eginnovations.com](mailto:sales@eginnovations.com) | [www.eginnovations.com](http://www.eginnovations.com)

US +1 866 526 6700 | SINGAPORE +65 6423 0928 | UK +44 (0)20 7935 6721 | NETHERLANDS +31 (0)70 205 5210

INDIA +91 44 4263 9553 | LATIN AMERICA +52 55 5533 3395 | HONG KONG: +852 3972 2415