J-CLASS
solutions

JVM Analyzer Monitor

The most qualified support team

Introducing **JVM JAM**, the in-depth Java JVM Analyzer/Monitoring application for Java versions 6, 7, 8, and the latest versions, 9, 10, and 11, that reads Java Garbage Collection (GC) log files produced by your JVM while running your Java application. These log files can be either be static log files or files that are dynamically generated in real-time.

With JVM JAM you can perform in-depth analysis and real time monitoring and alerting of dozens of metrics relative to the behavior of your JVM and application and system that they are running on. Works with single or multi-hosts. Some metrics that you can analyze and monitor are:

- Allocation Rate, to see how quickly your applications tasks are being executed
- Heap Use, to see how much memory your application is using
- Garbage Collection, to see how much of your servers' resources are taken away from your application by Java housekeeping processes
- System, to see how much of your servers' resources are being used in general

JVM JAM can perform deep JVM analysis to help Enterprise Java applications keep running at peak performance and offers a way to find inherent Java and "ghost in the machine" issues that only highly skilled Java engineers understand. JVM JAM will be the most complete and sophisticated Java analyzer and monitoring tool available.

We are introducing JVM JAM on the Splunk platform. We chose Splunk because it is a popular platform that captures, indexes, and correlates real-time data in a searchable repository that can generate graphs, reports, alerts, dashboards and visualizations.

Using JVM JAM you can now monitor and troubleshoot your Java instances and keep your data secure by not having to upload your log files to a third-party site.

J-CLASS solutions

Full list of planned features

- **Thread Delay** - App Thread Delay
- **Memory** - Memory Use
- **Page Faults** - Page Faults
- **Threads** - Thread Snapshots
- **Allocation Rate** - Allocation Rate
- **Heap Chart** - Heap Before and After GC - Connected
- **Live Set** - Live Java Objects
- **Paging** - Heap Paging Behavior
- **Page Cache** - Page Cache
- **Heap Use** - Detailed Heap Use per Specific Phase
- **Undo** - Undo
- **References** - Object References
- **Marking** - GC Marking Details
- **Relocation** - GC Relocation Details
- **Phase Contention** - GC Phase Contention
- **Critical Stalls** - Critical Stalls
- **GC Count** - Running Aggregate of the number of GCs that have occurred
- **Duration** - GC Duration
- **STW** - Stop The World Cause
- **GC Triggers** - GC Trigger Causes
- **GC Subphase** - GC Subphase Details
- **CPU** - CPU Utilization
- **Safe Point Details** - Safe Point and GC Details
- **References** - Object References
- **System Load** - System Load Average
- **Page Cache** - Page Cache Memory Use
- **Linux Memory** - Linux Memory
- **Paging** - Paging Behavior
- **Threads** - Thread Snapshots

Contact us to add or customize any features you may require.

Try our Free open source high performance JVM SEGUE2LL for low latency performance and up to 4TB heap capability.