

Java Bytecode Instrumentation Limitations



A real transaction management product needs to follow through the transaction between different types of application-related components such as proxies, Web servers, app servers (Java and non-Java), message brokers, queues, databases and so forth. In order to do that, you need visibility to different types of transaction-related data, some of which only exists at the actual payload of each request. Java, since it is an interpreter, hides parts of the actual code implementation from the Java

layer. The Java Virtual Machine (JVM) itself is written in C, therefore there are operating system-specific pieces that are not accessible from the Java layer, and thus not accessible to the bytecode instrumentation (BCI). Also, different Java packages utilize native code for reasons of performance or code reuse. This native code (libraries loaded to the JVM) is not accessible by BCI since it is not written in Java.

Learn how analytical data
can help you make
informed business decisions



[Get your Whitepaper here](#)

Contact Us



Get Your Free E-Book Download



For example: Let's say that we need some data that is part of the handshake between the Java application server and Oracle. Using BCI, you can only get access to data stored and managed by the Oracle [JDBC](#) driver. If some of the implementation is part of a .dll/so file the JDBC driver is using, we will not be able to access it by using BCI.

Another example: if we want to use features of TCP/IP packets for tracing a transaction between two servers, the actual structure of packets is not accessible from the Java layer, since it is done by operation system libraries that are utilized by the JVM itself.

Read my previous blog posts that [introduce bytecode instrumentation](#) and discuss [how bytecode instrumentation affects transaction tracing](#).

Overall, Java bytecode instrumentation is a useful concept that makes the lives of Java developers much better by giving potential visibility into every single class and method call. The deeper the visibility, the higher the overhead. However, a transaction management solution for a production environment cannot allow itself to impact such magnitude of overhead, requiring the trace to be limited by design. A limited trace has to be tailored to every Java application, which makes implementation in real-life scenarios a much more costly task. More so, there are pieces of data that can be crucial if you want to trace transactions across more than just one Java hop, and they are available only at the lower layer than the Java code, thus not accessible by BCI and limiting the ability to trace transactions in the real world.

🕒 Posted on January 3, 2011

🔖 [bci](#), [bytecode](#), [bytecode instrumentation](#), [java bytecode](#), [Lanir Shacham](#), [transaction management](#), [transaction tracing](#)

← [End-to-End Transaction Monitoring](#)

[First International Bank of Israel](#) →



Blog Topics

[APM](#)

[Application Performance Management](#)

[Application Performance Monitoring](#)

[Business Transaction Management](#)

[Capacity Planning](#)

[End-to-End Visibility](#)

[Information Security](#)

[Real User Monitoring](#)



Transaction Management

Transaction Tracing

Archive

[May 2018](#)

[April 2018](#)

[March 2018](#)

[February 2018](#)

[January 2018](#)

[December 2017](#)

[November 2017](#)

[October 2017](#)

[September 2017](#)

[July 2017](#)

[April 2017](#)

[March 2017](#)

[January 2017](#)

[December 2016](#)



November 2016
August 2016
July 2016
June 2016
May 2016
April 2016
March 2016
February 2016
January 2016
November 2015
October 2015
September 2015
August 2015
July 2015
June 2015
April 2015
March 2015
February 2015



September 2014

August 2014

July 2014

June 2014

May 2014

April 2014

March 2014

February 2014

January 2014

December 2013

November 2013

October 2013

September 2013

August 2013

July 2013

June 2013

May 2013

April 2013



March 2013

February 2013

January 2013

December 2012

November 2012

October 2012

September 2012

August 2012

July 2012

June 2012

May 2012

April 2012

March 2012

February 2012

January 2012

December 2011

November 2011

October 2011



September 2011

August 2011

July 2011

June 2011

May 2011

April 2011

March 2011

February 2011

January 2011

December 2010

November 2010

October 2010

September 2010

August 2010

July 2010

June 2010

May 2010

April 2010



March 2010

February 2010

January 2010

November 2009

October 2009

August 2009

July 2009

June 2009

May 2009

February 2009

September 2008

July 2008

June 2008

February 2008

January 2008

December 2007



Learn

[About CorrelSense](#)

[Meet the Team](#)

[News](#)

[Testimonials](#)

[Our Partners](#)

[Customers](#)

[Board of Directors](#)

Engage

[Product – SharePath](#)

[Product Features](#)

[Case Studies](#)

[Resources](#)

[Blog](#)

[Get SharePath Free](#)

Follow



The Correlsense website uses cookies. [Find out more in our Privacy Policy](#)

Okay, thanks

