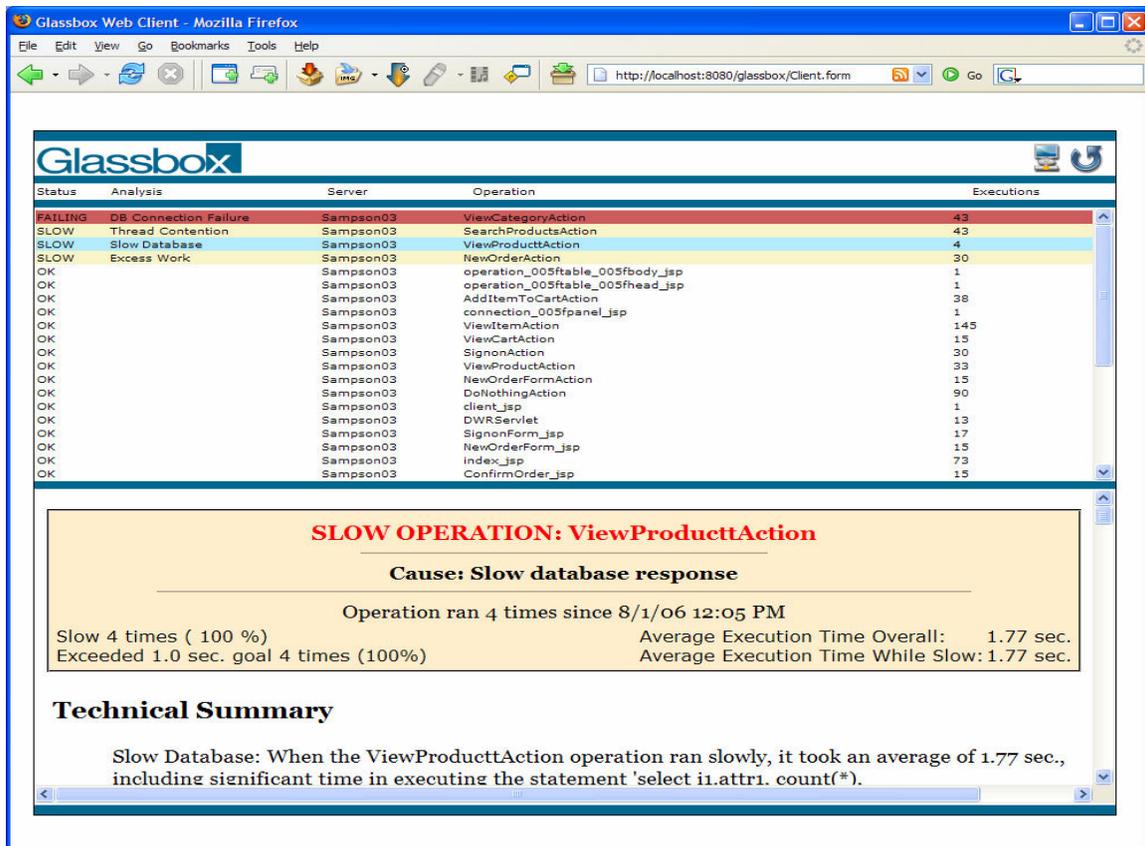# Glassbox 2.0 User Guide

I.   Welcome to Open Source Troubleshooting

Congratulations, you're about to experience easier Java problem diagnosis than you've ever seen before. Glassbox is a management agent with a browser-based interface that automatically pinpoints common problems in Enterprise Java applications.

Glassbox 2.0 is a free and open source product which is the logical successor to both Glassbox Troubleshooter 1.x and the Glassbox Inspector project. Upgrade to 2.0 from either project.

Locating the reason for an enterprise application failure represents the lion's share of the time spent getting your system healthy. Glassbox embeds the troubleshooting logic, empowering you to identify common problems immediately. You may still need your sniffers, test harnesses and probes, but thankfully they can now be focused on the exotic problems and their complexity can be avoided when dealing with day to day issues.

The Glassbox troubleshooter uses Aspect-Oriented Programming (AOP) and Java Management Extensions (JMX) technology to monitor your enterprise Java, without forcing you to embed anything or change a single line of code. Glassbox provides a real time diagnosis of your system and cross-references it against both your service levels and our knowledge base of failures.

II. Table of Contents

III. Concepts

Glassbox troubleshooting

The Glassbox troubleshooter includes three components:

The browser-based *client* is an Ajax GUI which connects to Glassbox agents, and displays the problems found in your Java application.   Usually the client just runs as a standard web application on the same server as the monitored application, but multiple clients can also connect

to a single agent simultaneously or a single client can connect to multiple agents in a clustered environment.

The *agent* component runs on your application server, the same app server running the application you want diagnosed. It can also be run in a clustered environment.

The *monitor* consists of a shared jar file(s) which is installed in the library directories of your application server. This uses AOP technology to dynamically monitor your application, all without any code changes or rebuilding.
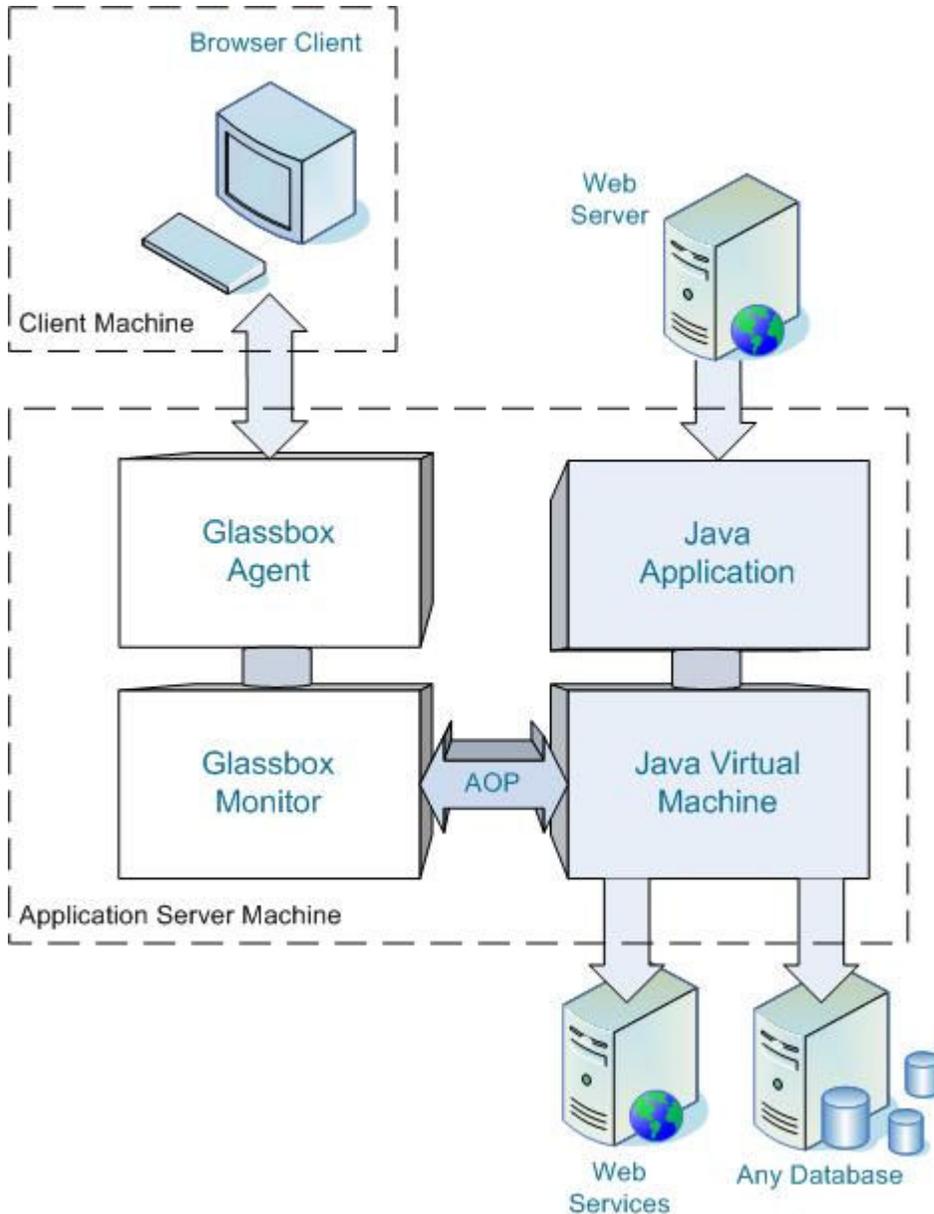
Operations

The Glassbox troubleshooter watches for the JVM to execute certain high-level methods commonly used to process external requests, such as Servlets and Struts Actions. It refers to these as *operations*, measures them against a performance goal, and presents diagnoses for any that fail or run too slowly. An operation might represent saving a shopping cart, looking up data from a remote financial server with a web service call, or retrieving information from a database, rather than the hundreds of Java methods that underlie these features.

Glassbox understands many modern frameworks such as Struts, Spring MVC, and JAX-RPC and intelligently discovers the top level operations as the JVM executes them. Generally Glassbox will discover reasonable operations even with custom dispatch frameworks, particularly if the coder followed best practices.

Glassbox operations represent high level activities, however custom frameworks sometimes create dispatch points which funnel many types of operations together, and this could cause Glassbox to show multiple operations summarized on the same line. If you would like a more fine grained control over the summarization points represented in the Glassbox summary table, you can create a custom operation plugin (normally 10 lines or less of code). This is dicussed further in the Custom Glassbox Monitors section of this document.

If you have any questions about frameworks supported, or ways to control how operations are summarized, please contact us or ask on the forums, and we'll be happy to assist you.

"Slow" operations

Unlike a profiling tool designed to simply measure *timings*, Glassbox is optimized to detect and pinpoint *performance problems*. To do so it must measure operations against performance goals.

Glassbox's performance goal is defined by a maximum time and a compliance percentage. For example, if you specify your application should fulfill requests in 0.8 seconds or less, 98% of the time, Glassbox will highlight operations as slow only if more than 2% of executions it has observed have taken longer than 0.8 seconds.

Configuring Service Level Agreements (SLAs)

By default Glassbox uses a performance goal that highlights slow operations as those that take longer than 1 second in at least 5% of executions. You can set the goal separately for each Glassbox agent in the beans.xml file.

To configure these properties, you can edit the glassbox.properties file for the Web application. This is located inside the glassbox.war file at Glassbox/WEB-INF/classes/glassbox.properties. Many application servers will also unpack the Web application and allow you edit the properties file, e.g., in tomcat you can typically edit TOMCAT_HOME\webapps\glassbox\WEB-INF\classes\glassbox.properties. If you do edit this file, please keep a back up so you can update the properties when you update the application. The following two properties correspond to 1000ms and 5%, you can edit them as needed to meet your performance criteria.

slowoperationAnalyzer.slowThresholdMillis=1000

operationAnalyzer.minimumSlowFrac=0.05

IV. System Requirements

Monitored Application Requirements
- Any working J2EE based application. Your application must start prior to diagnosing with Glassbox.

System Requirements
- Server Platforms: Windows, Linux, Solaris. Glassbox should work on other servers that support Java but it would require additional testing. Please let us know if you have tried Glassbox on other platforms, or are interested in other certifications.
- Glassbox supports Java 5 VMs as well as Java 1.4. Glassbox does provide some additional information while working on Java5 since we utilize the additional management information they provide. We have tested on Sun VM's, BEA JRockIt, and IBM's J9 VM's. Java 1.3 VM's are currently being tested and should be available soon. If you wish to check the status of a particular version, please contact Glassbox.
- Application Servers: Glassbox should work on any application server that supports Servlet 2.3 or later. We have tested and automated the installation process on Apache Tomcat 5.5.x, 5.0.x, 4.1.x, WebLogic 8.1 and 9.x, JBoss 4.x, Oracle OC4J 10.1.x. We have installed Glassbox on Websphere 5.0, 5.1, 6.0 and 6.1 and are testing and optimizing to support an automated installation, and plan to support WebSphere and probably Glass Fish in the 2.0 Beta release. If you want to get Glassbox working on another server, we'd be glad to help you.
- We are constantly adding support for other platforms. Please check on the Glassbox website under [downloads](#) for the most up to date list of supported platforms. We may be able to move support up, or we may be able to work with you.

Glassbox software required
- glassbox.war includes both the installer for the agent and our web-client.

All software and documentation can be downloaded from www.glassbox.com.

V. Automated Installation

Agent (Server) Installation

Glassbox ships with an automated installation process in its Web client that supports many popular application servers.   This works in the majority of cases, but if you prefer a hands-on install or you have trouble with the automated agent installer, the manual installation instructions are listed in the appendix of this document. If you want to install in a clustered environment, please see also the next section.

Rapid Install Process:

1. Drop the glassbox.war file into your running Application Server. Treat the glassbox.war file like any other application .war file; if your Application Server normally requires additional configuration to deploy and run a new application .war file then you will need to perform those steps here.
2. Use your browser to visit the Glassbox installer application.   For example, if your Application Server was running on the machine 'localhost' on port 8080 you would go to: http://localhost:8080/glassbox/InitializeInstall.form or just http://localhost:8080/glassbox
3. Follow the instructions in the Glassbox installer
4. Restart your Application Server either using the wrapper generated by the Glassbox installer or restart after manually setting Java VM and possibly Classpath environment variables yourself as described by the Glassbox installer.
5. Visit the glassbox/Verify.do page to verify Glassbox started on your application server. This would be http://sampson:8080/glassbox/Verify.do if you are connecting to check that Glassbox on the 'sampson' machine has started correctly.
6. Verify that the following Glassbox information is getting displayed on the console or logged correctly.
   ++
   -   Glassbox Agent has started successfully.
   ++

   A note about logging: If you are not seeing the console output expected, particularly if you see no Glassbox data at all but know you followed the directions above, then please review the appendix section on logging.   Glassbox uses Jakarta Commons Logging and logs helpful information at INFO level.

Viewing Glassbox analysis in your browser

1. Open a browser. Firefox and Microsoft Internet Explorer are both supported. Safari support is planned, but for the time being we recommend that Mac users use Firefox.
2. View the client in your browser. Point your browser at the application 'glassbox' on the machine where Glassbox was installed. Glassbox is a standard web-application in this way, whoever setup Glassbox on your application server should be able to give you this URL.
   For example: If the Glassbox application is being run on the machine 'sampson' on the default port '8080', then the glassbox client would be at http://sampson:8080/glassbox
3. See your troubleshooting data. Glassbox monitors any application local to it by default, so you should see a green dot next to 'local' in the connections tab, and you will see traffic appear in the table when you click or run load on your application.

   Note: This assumes you are running both Glassbox and the application you wish to monitor on the same application server. This is recommended to verify and test Glassbox functionality. To set up Glassbox in a multiple server environment, please see the next section.

VI. Clustered Installation

Agent (Server) Installation

Glassbox ships as a standard Web application .war file.

The .war file contains two major components. First, an automated installer portion that lays down the Glassbox monitoring files and must be run on each application server that you wish to monitor. Second, a standard web application that runs on one or more application servers and provides analysis and a browser interface for the monitored data. This web application is normally run on the same application server being monitored, but it can be run in a remotely in a clustered environment using either a direct RMI protocol or JMX Remote using RMI.

Advanced Clustered Install:

1. Deploy the glassbox.war file to <u>each</u> of your application servers with application(s) that you would like to monitor. Treat the glassbox.war file like any other application .war file, if your Application Servers require additional configuration to deploy and run a new application .war file then you will need to perform those steps here.
2. Use your browser to visit the Glassbox installer at /glassbox/InitializeInstall.form on each application server.
    Example: if the applications to be monitored are running on both 'sampson' and 'mckinley', you would run through the installation steps by visiting both of the following: http://sampson:8080/glassbox/InitializeInstall.form followed by http://mckinley:8080/glassbox/InitializeInstall.form

3. Follow the instructions in the Glassbox installer to install on each of the application servers. If you have more than one server on the same machine, see the section on "Configuring Agent Communication" below.
4. Restart each of the Application Server to be monitored, either using the wrapper generated by the Glassbox installer or restart after manually setting Java options environment variables yourself as described by the Glassbox installer.
5. Visit the glassbox/Verify.do page for each of your connections to verify Glassbox started on your application servers, for example to verify installation on 'sampson', you would go to http://sampson:8080/glassbox/Verify.do
6. Verify that the following Glassbox information is getting displayed on each of the consoles or is logged correctly.
    +++
    -   Glassbox Agent has started successfully.
    +++

    A note about logging: If you are not seeing the console output expected, particularly if you see no Glassbox data at all but know you followed the directions above, then please review the appendix section on logging. Glassbox uses Jakarta Commons Logging and logs helpful information at INFO level.
    A note about using Oracle OC4J: If you are running oc4j in a clustered mode, add a -userThreads argument to CMDARGS in the oc4j launch script

A note about using WebLogic: If you are using WebLogic version 8.1 or before, it includes it's own JMX implementation which is incompatible with other App Server. So you cannot mix and match a clustered environment with WebLogic 8.1 and other servers, it must be all WebLogic 8.1 in your cluster or no WebLogic 8.1.

A note about firewalls and connectivity: Before trying to run a clustered Glassbox, verify you can telnet from the machine running the Glassbox webapp into each of the machines you need monitored. It is not uncommon to have firewall or networking issues, and those need to be configured before running anything clustered.

Configuring Agent Communication

In a clustered environment, by default Glassbox uses port 7232 to listen for remote connections to view server data. It does this for both RMI and JMX Remote over RMI. In most cases, you won't need to change this for Glassbox to work. However, in some cases you might need to change these settings. For example, if you are running multiple agents on the same machine, you will want to pick a different port number. If you want to access remote data through a firewall, you might want to change the port number also.

To configure these properties, you can edit the glassbox.properties file for the Web application. This is located inside the glassbox.war file at Glassbox/WEB-INF/classes/glassbox.properties. Many application servers will also unpack the Web application and allow you edit the properties file, e.g., in tomcat you can typically edit TOMCAT_HOME\webapps\glassbox\WEB-INF\classes\glassbox.properties. If you do edit this file, please keep a back up so you can update the properties when you update the application.

To change the port on a server from the default, uncomment the following three lines and edit the port number, e.g., to use port 8188 add these lines to the glassbox.properties file:

rmiJmxRegistry.port=8188
rmiService.registryPort=8188
glassboxJmxServerConnector.serviceUrl=service:jmx:rmi://localhost:8188/jndi/rmi://localhost:8188/GlassboxTroubleshooter

It is also possible to configure Glassbox to use more advanced RMI and JMX options such as security and to disable use of RMI and/or JMX over RMI. Glassbox by default uses an existing JMX server to store its data. That too can be configured. If you need help editing the property files and JMX system configuration to do these things, please contact us or post on the forum for help.

Viewing Clustered Glassbox Information

1. Open a browser. Firefox and Microsoft Internet Explorer are both supported. Safari isn't working correctly for the 2.0 alpha, but we expect to support it for 2.0 final.
2. View the client in your browser. Point your browser at the application 'glassbox' on the machine where Glassbox was installed. Glassbox is a standard web-application in this

way, whoever setup Glassbox on your application server should be able to give you this URL.

For example: If the Glassbox application is being run on the machine 'sampson' on the default port '8080', then the glassbox client would be at http://sampson:8080/glassbox

3. Open the monitor panel. Click the network connections icon on the upper right corner of your Glassbox client.
4. Add new network connections for each additional agent you want to monitor.
   a. Click the add link
   b. Choose a short display name which helps you remember your remote server
   c. Identify the hostname of the computer where your monitored application is running. Make sure the application server has been restarted using the Glassbox wrapper or Java environment settings, so it has Glassbox monitoring enabled.
   d. Choose the protocol as 'RMI' or 'JMX on RMI'. Using pure RMI is generally simpler. Using JMX on RMI also supports viewing lower-level detailed JMX data in a remote console like JConsole. However, on certain legacy servers (like Weblogic 8.1) using JMX Remote requires additional configuration.
   e. Choose the port where Glassbox is listening. By default this is 7232. , if you did not manually change it as described above, just accept the default.
   f. Click save
   g. On the list of connections, click on the red-dot next to the display name of the connection you just added. The dot should turn green, and this would indicate that you are successfully monitoring your server. Of course you must have clicked on your monitored app, or some traffic must exist for Glassbox to show anything.
5. Rinse and Repeat. Repeat step 4 for each of the servers where you installed Glassbox.
6. Check each connection. Click on each of the monitored applications to generate at least a little load, and then verify that some load for each server is showing up in the Glassbox client.

Troubleshooting a clustered install:
- Are your servers and Glassbox running? Do you see on the console or in the logs that Glassbox started successfully? You may need to re-start, or verify that either you are starting with the Glassbox wrapper or you are manually setting JAVA_OPTS.
- Did you run through the Glassbox install on each of them?
- Did this work previously? If so, you probably need to re-install Glassbox on each application server since the client and server got out of synch.
- Can you reach the failing application server from your client machine? Run the monitored application from the machine where you intend view the Glassbox client.
- If you can see data on each server locally, but can't connect remotely, there are a few common problems that might arise, notably an RMI bug from spaces in directories or JMX incompatibilities. The following sections describe how to diagnose and fix these issues.

RMI Bug Workaround

Whether you use direct RMI or JMX over RMI, Glassbox uses RMI for remote communications. Unfortunately, RMI has a bug in handling limited understanding of directories with a space in

them. This normally comes up on Windows where your application server may have been installed in a 'Program Files' directory by default.   This is a long standing RMI problem, and the solution is to either move your Application Server to a directory without a space in it, or to configure Glassbox to avoid the issue. If a server has any jars or directories on its classpath that have spaces in them, then you should add an additional Java start-up parameter in JAVA_OPTS (or equivalent) for all the servers in the cluster:

Add the following Java start-up parameters to all the application servers in the cluster:

-Djava.rmi.server.useCodebaseOnly=true -Djava.rmi.server.codebase=http://localhost:8080

JMX Incompatibilities

Some servers include older versions of the JMX specification. For these servers, the easiest way to view clustered Glassbox data is to use direct RMI. However, if you are interested in viewing detailed data through JMX Remote, it might be possible to configure your server to use a newer JMX or for older Weblogic servers, to use an adapter to overcome this issue.

As of this writing WebLogic 8.1 and earlier have an older version of JMX that is incompatible, and we are researching alternatives.   Please see the Glassbox Forums for the latest information.

VII. Using Glassbox

The Operation Summary Table

The Glassbox Web interface contains an operation summary table at the top, showing the operations that the JVM has executed and their status, and details for the highlighted operation at the bottom. Hitting control while clicking on an entry in the summary table opens a new window showing the details.

Status

The status column tells at a glance whether the operation is performing normally (OK), exceeding the performance goal (SLOW), or encountering an error (FAILING).

Analysis

For any operation that is slow or failing, you can look in the analysis column for a quick summary of the cause of the problem.

Operation

Glassbox uses internal heuristics and a knowledge of common application frameworks to determine an appropriate name for each operation.

Server

This column displays the name of the agent connection that was configured using the connection panel. In a clustered environment, it allows you to distinguish operations on different servers.

Executions

The executions column indicates how many times this operation has run since the application server was started or Glassbox's statistics were last reset.

The Operation Details Area

The details area provides information relating to operations selected in the summary table. It starts with an Executive Summary, outlining the problem's severity, type and location, then continues through a Technical Summary to the Technical Details providing relevant database, remote service and code level details that allow DBAs and developers to locate and fix issues.

Hint: Control-click on a line in the summary table to open the details in a new window.

Executive Summary

The executive summary is contained in the first table that lives at the top of the detail pane. A high level view of the problem, intended to give someone a quick view of what general area of code is affected, and how severe the problem is.

Technical Summary

The technical summary provides management level details that include additional specifics in paragraph form, including such things as what query or method was slow, what database is affected etc. This is intended to give a 15-second summary of what the problem is, and to be suitable to cut-and-paste into summary reports for management.

Technical Details

Technical details provide an engineer enough information to find and fix the problem. This includes stack traces, the names of thread locks, SQL tables and other forms of exact and highly technical information. It is suitable for a manager to mail to an engineer, or for QA or IT professionals to use when filing a bug.

The details page also contains other helpful information that is not shown above:

Common Solutions

Shows common solutions that people have found to fix problems similar to the one you are experiencing.   These are hints and valuable as a rule of thumb, particularly if you are not familiar with fixing a problem of the type diagnosed.

Ruled out potential problems

Lists the other types of problems Glassbox considered and eliminated as possible problems.   If troubleshooting a particularly sticky issue, it saves time to know what problems have already been ruled out.

Remote Data through JMX Consoles

Glassbox agent supports using popular JMX Consoles to connect to a monitored server to view detailed statistics about the performance of operations and resources they use. You can use a tool like the Java 5 JConsole to view this information. If Glassbox is running correctly on a server and the JMX Remote connector is working properly, you can view data by running the following command:

jconsole service:jmx:rmi:///jndi/rmi://localhost:7232/GlassboxTroubleshooter

You can then browse into the MBeans tab and see more data inside the stats section beneath Glassbox:
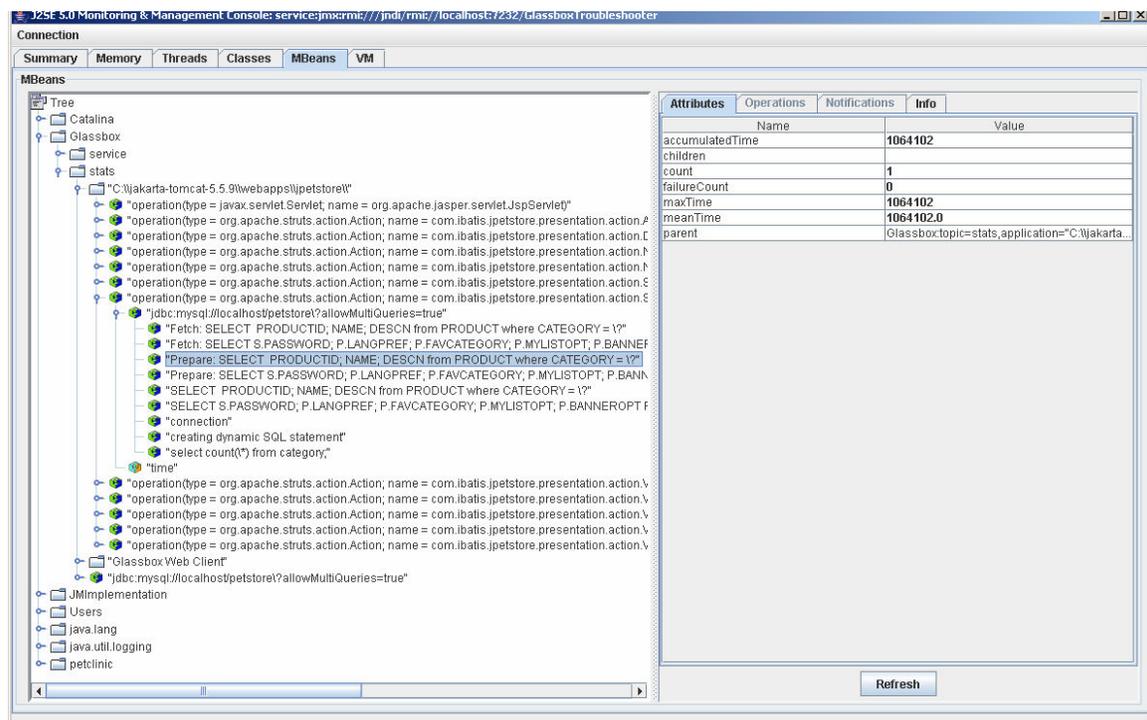
VIII.   Open Source

Glassbox is an open source project, it is free to download and run. It is made available under the Lesser Gnu Public License. This makes it free to use as a library without imposing restrictions on using applications as long as they don't depend on changes to it.

If you might be interested in contributing to Glassbox, please see our website at www.glassbox.com and click on 'get involved'.

Visit the Glassbox licensing page at:

Copyright 2006, Glassbox Corporation

http://www.glassbox.com/glassbox/Licensing.html or
http://www.glassbox.com/glassbox/OSglassbox/Licensing.html

IX. Getting Help and Reporting Problems

If you're encountering problems using Glassbox, please visit our website and community forums at www.glassbox.com/forum/forum/listforums.

X.   Logging

Glassbox uses Jakarta Commons Logging 1.1 by default.   However, if the application server comes with a version of Commons Logging it will use that instead. In general, to configure Glassbox logging, you need to configure your application server logging system. We list example values below to add to a log4j.xml and logging.properties file to best configure Glassbox logging for servers that are using log4j or Java.util.logging respectively.

Log4j.xml Edits for Log4J

```
     <appender name="glassbox"
class="org.apache.log4j.RollingFileAppender">
        <param name="File" value="glassbox.log"/>
        <param name="Append" value="true"/>
        <param name="MaxBackupIndex" value="5"/>
        <param name="MaxFileSize" value="10MB"/>
        <layout class="org.apache.log4j.PatternLayout">
           <param name="ConversionPattern" value="%d{DATE} %-6r [%t]
%-5p %c %x - %m %n"/>
        </layout>
    </appender>

     <appender name="diagnostics"
class="org.apache.log4j.RollingFileAppender">
        <param name="File" value="glassbox-diagnostics.log"/>
        <param name="Append" value="true"/>
        <param name="MaxBackupIndex" value="5"/>
        <param name="MaxFileSize" value="10MB"/>
        <layout class="org.apache.log4j.PatternLayout">
           <param name="ConversionPattern" value="%d{DATE} %-6r [%t]
%-5p %c %x - %m %n"/>
        </layout>
    </appender>

     <appender name="iterations"
class="org.apache.log4j.RollingFileAppender">
        <param name="File" value="glassbox-iterations.log"/>
```

```
        <param name="Append" value="true"/>
        <param name="MaxBackupIndex" value="5"/>
        <param name="MaxFileSize" value="10MB"/>
        <layout class="org.apache.log4j.PatternLayout">
            <param name="ConversionPattern" value="%m %n"/>
        </layout>
    </appender>

    <category name="org.springframework">
        <priority value="WARN" />
    </category>
    <category name="org.springframework.beans">
        <priority value="INFO" />
    </category>
    <category name="org.springframework.remoting">
        <priority value="INFO" />
    </category>
    <category name="org.springframework.jmx">
        <priority value="INFO" />
    </category>
    <category name="log4j.logger.uk.ltd.getahead.dwr">
        <priority value="WARN" />
    </category>
    <category name="glassbox">
        <priority value="INFO" />
        <appender-ref ref="glassbox"/>
    </category>
    <category name="glassbox.track.exceptions" additivity="false">
        <priority value="DEBUG" />
        <appender-ref ref="exceptions"/>
    </category>

    <category name="glassbox.track.diagnostics" additivity="false">
        <priority value="DEBUG" />
        <appender-ref ref="diagnostics"/>
    </category>

    <category name="glassbox.track.diagnostics.iterations"
additivity="false">
        <priority value="DEBUG" />
        <appender-ref ref="iterations"/>
    </category>
```

Logging.properties Edits for Java.Util.Logging

```
glassbox.installer.level=FINEST
uk.ltd.getahead.dwr.level=WARNING
org.springframework.level=WARNING
org.springframework.beans.level=INFO
org.springframework.remoting.level=INFO
org.springframework.jmx.level=INFO

glassbox.track.diagnostics.level=FINEST
glassbox.track.diagnostics.handler=java.util.logging.FileHandler
```

```
glassbox.track.diagnostics.handler.formatter=java.util.logging.SimpleFo
rmatter
glassbox.track.diagnostics.handler.append=true
glassbox.track.diagnostics.handler.pattern=/glassbox_diagnostics.log

glassbox.track.diagnostics.iterations.level=FINEST
glassbox.track.diagnostics.iterations.handler=java.util.logging.FileHan
dler
glassbox.track.diagnostics.iterations.handler.formatter=java.util.loggi
ng.SimpleFormatter
glassbox.track.diagnostics.iterations.handler.append=true
glassbox.track.diagnostics.iterations.handler.pattern=/glassbox_iterati
ons.log
```

XI. Custom Glassbox Monitors

Glassbox builds on the load-time weaving infrastructure of AspectJ. This makes it easy to add a custom monitor to Glassbox. For example, you can simply extend the Glassbox definition of operations by creating a new XML file with these contents:

```
<aspectj>
  <aspects>
      <concrete-aspect name="ServiceProcessingMonitor"

      extends="glassbox.monitor.ui.TemplateOperationMonitor">
          <pointcut name="methodSignatureControllerExecTarget"
                    expression="within(com.myco.service..*)"/>
      </concrete-aspect>
  </aspects>
</aspectj>
```

You can then add this file to a META-INF subdirectory of a directory on your classpath or add a jar containing the file at the location META-INF/aop.xml. For Tomcat, you might just create the directory common/classes/META-INF and install your custom aop.xml file there.

For more information about configuring AspectJ load-time weaving see also the following resources:

- **Next steps with aspects: Extending a library aspect**
- **Performance Monitoring with AspectJ, part 2: Deploying the Glassbox Inspector**
- **AspectJ Development Environment Guide: Load-Time Weaving**

XII. Manual Installation

Agent (Server) Installation

Note: Glassbox provides support for unpacking files and generates recommended environment settings in its automatic installer, even for servers it doesn't recognize. If you want to configure your installation and running process, we'd highly recommend you try using the automated installer with a manual installation option and let the installer unpack the right configuration for your VM and provide you with template start up options.

Manual or automatic installation of Glassbox on a Java application server broadly requires four tasks that are specific to the application server.

1. Open the glassbox.war file and extract the Glassbox Monitor and AspectJ weaving jars to the appropriate place on the application server.
   a. Put the aspectjweaver.jar and glassboxMonitor.jar on the classpath for the server and all applications. For example: /common/lib for tomcat or /server/<SERVERNAME>/lib for JBoss. For some servers, this is done by adding the jars to the CLASSPATH used to run the server.
   b. For a Sun or IBM Java 1.4 VM, you will also want to create an adapter to enable load-time weaving. To do this:
      i. Extract aspectj14Adapter.jar and createJavaAdapter.jar from the glassbox.war located in install/glassbox14.
      ii. Run java -jar createJavaAdapter.jar using the Java version you will use to run the server.
2. Configuring the application server's Java environment to set up the Glassbox Agent. You need to enable the load-time weaving system and to define a directory where Glassbox configuration files are stored. You can create this directory anywhere: by convention it is often a subdirectory named glassbox of the server library directory.
   a. For Java 1.5: your VM arguments must add a '-javaagent:' option (notice it has a colon, not an equals) to point to aspectjweaver.jar. For example, with Tomcat:
      set JAVA_OPTS="-javaagent:<TOMCAT>\common\lib\aspectjweaver.jar" "–Dglassbox.install.dir:<TOMCAT>\common\lib\glassbox"
   b. For a Sun or IBM Java 1.4 JVM: your VM arguments need additional boot classpath entries and a system property. For example, with Tomcat:
      set JAVA_OPTS="-Xbootclasspath/p:<UNPACK_DIR>\java14Adapter.jar" "-Xbootclasspath/a:<UNPACK_DIR>\createJavaAdapter.jar;<UNPACK_DIR>\aspectj14Adapter.jar;<UNPACK_DIR>\aspectjweaver.jar" -Daspectwerkz.classloader.preprocessor=org.aspectj.ext.ltw13.ClassPreProcessorAdapter "–Dglassbox.install.dir=<UNPACK_DIR>"
   c. For a BEA JRockIt 1.4 JVM: your VM arguments need an additional –Xmanagement option. For example, with Weblogic:
      set JAVA_OPTIONS=-Xmanagement:class=org.aspectj.weaver.loadtime.JRockitAgent "–Dglassbox.install.dir=<SERVER>\lib\glassbox"
3. Add the glassbox.war file to your application server as any other webapp and run it.
4. Point your browser at the glassbox webapp, and generate some load to monitor.
5. Configure for clustered use. After you get Glassbox working, you may need to modify two additional parameters to support clustered use.
   a. On Unix or Linux: Unix systems require setting java security parameters for opening external RMI ports. You need to define environment variables for java.rmi.server.hostname and java.security.policy (see example below).

b. If there are spaces in your CLASSPATH. Java RMI has a bug in handling CLASSPATH file paths with spaces in them. If you have deployed a server with jars or directories with spaces in the classpath (e.g., to "c:\Program Files\Apache Software Foundation\..."), then you need to configure RMI codebases by setting system properties with arguments like -Djava.rmi.server.codebase=http://localhost:8080 and -Djava.rmi.server.useCodebaseOnly=true

Example Manual Agent Installation - Apache Tomcat on Java 5 JVMs

The installation location of the Apache Tomcat instance is referred to as <TOMCAT>.

We recommend that you install Glassbox as the same User who installed Tomcat, this avoids any permissions problems around Tomcat or Glassbox writing log files.

1. Verify that your app deploys on Tomcat and can be run. Glassbox only helps diagnose problems with successfully deployed and running applications.
2. Download the glassbox.war file to your application server machine and unzip it (this is only required for manual installation: automatic installation extracts and copies files automatically).
3. Copy the following jars from install directory in glassbox.war to <TOMCAT>/common/lib:

   <TOMCAT>/common/lib
   - aspectjweaver.jar
   - glassboxMonitor.jar

4. Add a Java startup variable "-javaagent" to your environment by setting the JAVA_OPTS environment variable:

   On Windows

   In the System control panel applet, add a JAVA_OPTS environment variable and set it to:

   -javaagent:<TOMCAT>\common\lib\aspectjweaver.jar

   Or you can set the value in the config file, for example add the following line toward the bottom of your setClasspath.bat file:

   set JAVA_OPTS=-javaagent:<TOMCAT>\common\lib\aspectjweaver.jar

   On Unix or Linux

   export JAVA_OPTS=
   "-javaagent:<TOMCAT>/common/lib/aspectjweaver.jar"

On Cygwin

Please look at the windows instructions and change the file separators appropriately.

(You can, of course, define these variables in a configuration file. Remember to replace <TOMCAT> with the full path.)

5. **Clustered Security Permission on Unix or Linux Only:** Unix systems require setting java security parameters for opening external RMI ports. To allow external RMI access, add the following definitions to the JAVA_OPTS environment variable:

> -Djava.rmi.server.hostname=<hostname>
> -Djava.security.policy=<TOMCAT>/common/lib/glassbox/java.policy

We included a sample java.policy file for use if you do not have one already.

If you have an existing policy file, you should add the following lines to your already existing java.policy file and make sure that your –Djava.security.policy environment variable listed above points to your already existing file.

```
grant {
          permission java.net.SocketPermission "*:1024-65535", "connect,accept";
};
```

6. Restart your application server. Startup will take longer as the application server weaves in Glassbox instrumentation.
7. Confirm that the application server console says "Glassbox Agent has started successfully."

Example Manual Agent Installation - Apache Tomcat on Java 5 JVMs

The installation location of the Apache Tomcat instance is referred to as <TOMCAT>.

We recommend that you install Glassbox as the same User who installed Tomcat, this avoids any permissions problems around Tomcat or Glassbox writing log files.

8. Verify that your app deploys on Tomcat and can be run. Glassbox only helps diagnose problems with successfully deployed and running applications.
9. Download the glassbox.war file to your application server machine and unzip it.
10. Copy the following jars from install directory in glassbox.war to <TOMCAT>/common/lib:

> <TOMCAT>/common/lib
> - aspectjweaver.jar
> - glassboxMonitor.jar

11. Create a glassbox directory at <TOMCAT>/common/lib/glassbox

12. Add a Java startup variable "-javaagent" to your environment by setting the JAVA_OPTS environment variable and add a –Dglassbox.install.dir to point to the directory you just created above:

<u>On Windows</u>

In the System control panel applet, add a JAVA_OPTS environment variable and set it to:

> -javaagent:<TOMCAT>\common\lib\aspectjweaver.jar –
> Dglassbox.install.dir=<TOMCAT>\common\lib\glassbox

Or you can set the value in the config file, for example add the following line toward the bottom of your setClasspath.bat file:

> set JAVA_OPTS=-javaagent:<TOMCAT>\common\lib\aspectjweaver.jar –
> Dglassbox.install.dir=<TOMCAT>\common\lib\glassbox

<u>On Unix or Linux</u>

> export JAVA_OPTS=
> "-javaagent:<TOMCAT>/common/lib/aspectjweaver.jar"

<u>On Cygwin</u>

Please look at the windows instructions and change the file separators appropriately.

(You can, of course, define these variables in a configuration file. Remember to replace <TOMCAT> with the full path.)

13. **Clustered Security Permission on Unix or Linux Only:** Unix systems require setting java security parameters for opening external RMI ports. To allow external RMI access, add the following definitions to the JAVA_OPTS environment variable:

> -Djava.rmi.server.hostname=<hostname>
> -Djava.security.policy=<TOMCAT>/common/lib/glassbox/java.policy

We included a sample java.policy file for use if you do not have one already.

If you have an existing policy file, you should add the following lines to your already existing java.policy file and make sure that your –Djava.security.policy environment variable listed above points to your already existing file.

> grant {
>   permission java.net.SocketPermission "*:1024-65535", "connect,accept";

};

14. Restart your application server. Startup will take longer as the application server weaves in Glassbox instrumentation.
15. Confirm that the application server console says "Glassbox Agent has started successfully."