BEA JRockit® Mission Control™

Using JRockit Mission Control in the Eclipse IDE

Mission Control 3.0.2
Document Revised: June, 2008

Contents

CHAPTER 1*

Introduction

The Eclipse plug-in edition of Oracle JRockit Mission Control provides seamless integration of JRockit Mission Control's application profiling and monitoring toolset with the Eclipse development platform. By integrating the JRockit Mission Control Client with Eclipse, you can combine the features of Eclipse with the power tool set that comprises JRockit Mission Control.

This document describes this integration and provides instructions for using the special functionality enabled by integrating the JRockit Mission Control Client with Eclipse. The topics include:

- Benefits of the Integration
- Differences between Eclipse IDE and RCP Versions of JRockit Mission Control
- Making the Oracle JRockit JVM Your JVM
- Selecting a Perspective
- Jumping to Application Source

Benefits of the Integration

When the JRockit Mission Control Client is run within the Eclipse IDE, you have access to IDE features that aren't otherwise available in the toolset when it is run as a standalone Rich Client Platform (RCP) application. The most significant of these features is the ability to see specific code in the running application by opening it directly from the JRockit Mission Control Client, a function called Jump-to-Source.

The other obvious benefit of integrating the JRockit Mission Control Client with the Eclipse IDE is that it allows you to profile and monitor an application during their development phase just as you would during their production phase. This allows you to spot potential runtime problems before you actually deploy your application to production; for example, you might, while monitoring an application during its development notice a memory leak. By catching the memory leak during development, you can correct it before you migrate your application to a production environment.

Differences between Eclipse IDE and RCP Versions of JRockit Mission Control

Generally, the Eclipse version of the JRockit Mission Control Client works identically to the RCP version. Any component in the Eclipse version offers the same functionality and user interface as the comparable component delivered on the RCP.

The biggest difference that the Eclipse version has over the RCP version is the Jump-to-Source feature. Jump-to-Source, described in Jumping to Application Source, is enabled by close coupling of the monitoring and profiling toolset with the development environment.With this feature, you can not only see the name of a "problem" class or method displayed in the JRockit Mission Control Client, but you can jump from the displayed name directly to that class or method's source, where you can evaluate the code to see what might be causing the problem. Jump-to-Source is enabled for the Management Console, the JRockit Runtime Analyzer, and the Memory Leak Detector.

Making the Oracle JRockit JVM Your JVM

While the JRockit Mission Control client can work with many different JVMs, it is highly recommended that you use the Oracle JRockit JVM as your JVM when running the JRockit Mission Control Client on the Eclipse platform. Not only will you avail yourself of the JRockit JVM's exceptional performance but, by using this JVM, JRockit Mission Control's autodetect feature is enabled, which makes it simple to connect JRockit Mission Control to your running application.

To make the JRockit JVM the JVM on which you will run JRockit Mission Control

1. Go to your file system browser (for example, Windows Explorer).

2. Locate your Eclipse installation folder (for example, `C:\Program Files\Eclipse`) and, with a file editor other than Notepad, open the file `eclipse.ini`. It will look something like the example in Listing 1-1.

Listing 1-1 eclipse.ini Example

```
-showsplash
org.eclipse.platform
--launcher.XXMaxPermSize
256M
-vmargs
-Dosgi.requiredJavaVersion=1.5
-Xms40m
-Xmx512m
```

3. Make the following changes to eclipse.ini:
   - Remove all flags related to other JVMs than the JRockit JVM (for example, `--launcher.XXMaxPermSize 256M`)
   - On the third line down (after `org.eclipse.platform`), add the following:

     -vm
     <*`Full path to JRockit JDK's javaw file`*>

     The full path to the JRockit JDK's javaw file might look like this on Windows:

     `C:\Program Files\Java\jrockit-R27.4.0-jdk1.6.0_02\bin\javaw.exe`

     or like this on Linux and Solaris:

     `$HOME/jrockit-R27.4.0-jdk1.6.0_02/bin/javaw`
   - Depending upon your particular JRockit JVM implementation and the applications running on it, you can set any valid JRockit JVM command-line option. For example, you might want to set a garbage collector that meets your system priorities by using the `-XgcPrio:` option or increase (or decrease) the initial and maximum heap size by changing the values for `-Xms` and `-Xmx`.

     For more information on tuning the JVM, please refer to Profiling and Performance Tuning in the *Oracle JRockit JVM Diagnostics Guide*.

For more information on the available command-line options, please refer to the *Oracle JRockit JVM Command-Line Reference*.

4. When you are done making the necessary changes to `eclipse.ini`, save and close the file. Listing 1-2 shows an example of the `eclipse.ini` file updated to make the JRockit JVM the active JVM.

Listing 1-2 Updated eclipse.ini file for a Windows implementation

```
-showsplash
org.eclipse.platform
-vm
C:\Program Files\Java\jrockit-R27.4.0-jdk1.6.0_02\bin\javaw.exe
-vmargs
-Dosgi.requiredJavaVersion=1.5
-Xms256m
-Xmx512m
-XgcPrio:pausetime
```

Selecting a Perspective

A "perspective" defines a set of views and their relative positions within the Eclipse window; in other words, it is a template for graphically presenting different types of information in Eclipse. For example, the Java perspective combines views that you would commonly use while editing Java source files, while the Debug perspective contains the views that you would use while debugging Java programs.

JRockit Mission Control plug-ins to Eclipse come with a predefined perspective called Mission Control. This perspective shows the JRockit Mission Control user interface so that you can use the tools that comprise JRockit Mission Control to profile applications as you develop them in Eclipse.

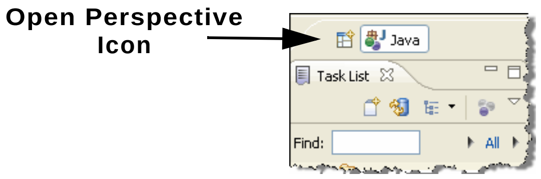
This topic will show you how:

- To open the JRockit Mission Control Perspective
- To change perspective from JRockit Mission Control
- To reopen the Mission Control Perspective

To open the JRockit Mission Control Perspective

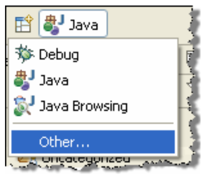
1. In top right corner of the Eclipse window, click the Open Perspective icon (Figure 1-1).

Figure 1-1 Open Perspective Icon



The Open Perspective context menu appears (Figure 1-2).

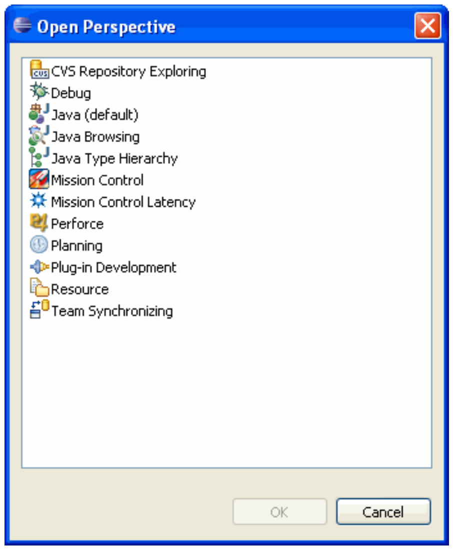
Figure 1-2 Open Perspective Context Menu



2. Select **Other...**

   The Open Perspective dialog box appears (Figure 1-3).

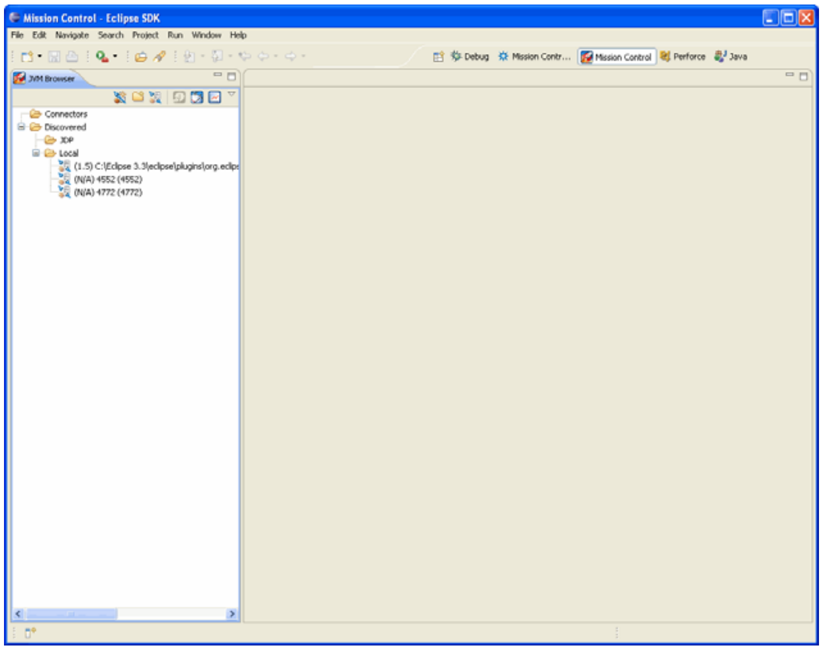
Figure 1-3 Open Perspective Dialog Box



3. Select **Mission Control** and click **OK**.

   The Eclipse window reconfigures to show the Mission Control Perspective (Figure 1-4).

Figure 1-4 Mission Control Perspective


Mission Control - Eclipse SDK
File Edit Navigate Search Project Run Window Help
Debug
Mission Contr...
Mission Control
Perforce
Java
JVM Browser
Connectors
Discovered
JDP
Local
(1.5) C:\Eclipse 3.3\eclipse\plugins\org.eclips
(N/A) 4552 (4552)
(N/A) 4772 (4772)

To change perspective from JRockit Mission Control

You can change perspectives from JRockit Mission Control to another perspective by using one of the methods described in Table 1-1:

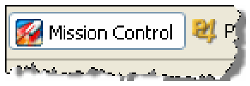
Table 1-1 Changing Perspectives

If...	Do this...
You've never opened the perspective	1. Click the Open Perspective icon. 2. Either: – Select the perspective you want to open. – If the perspective name does not appear on the context menu, select **Other...** to open the Open Perspective dialog box and select the perspective from there.
You've opened the perspective before	If you've opened the perspective before, a button for that perspective will appear in the top right corner of the Standard Mission Control Perspective, near the Open Perspective icon. Simply click the button for the perspective you want to open.

To reopen the Mission Control Perspective

If you have already opened the Mission Control Perspective for this project, a Mission Control button will appear next to the Open Perspective button in the top right corner of the Eclipse window (Figure 1-5).

Figure 1-5 Open Mission Control Perspective Button



To reopen the perspective, simply click that button.

Jumping to Application Source

When running JRockit Mission Control plug-ins in an Eclipse IDE you can select a method or class and jump from the JRockit Mission Control Client directly to the source code where that method or class is declared. An editor will open up showing you the source file. Jump-to-Source is available in JRA, the Management Console and the Memory Leak Detector:

This topic contains the following information:

- Using Jump-to-Source
- JRockit Mission Control Plug-ins with Jump-to-Source Enabled

Using Jump-to-Source

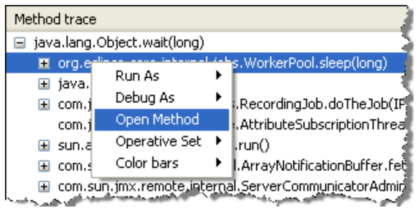
To jump from the JRockit Mission Control Client to source code

Note: The following procedure is generic. See JRockit Mission Control Plug-ins with Jump-to-Source Enabled for a list of plug-ins where this feature is enabled.

1. On the table, tree or other GUI component listing classes or methods, right-click the class or method for which you want to see the source code.

   A context menu appears (Figure 1-6).

Figure 1-6 Jump to Source Command on the Context Menu


Method trace
java.lang.Object.wait(long)
org.e
s.WorkerPool.sleep(long)
java.
com.j
.RecordingJob.doTheJob(IP
com.j
.AttributeSubscriptionThrea
sun.a
.run()
com.s
l.ArrayNotificationBuffer.fet
com.sun.jmx.remote.internal.ServerCommunicatorAdmi
Run As
Debug As
Open Method
Operative Set
Color bars

2. Select **Open Method** (or **Open Type**, if you are jumping from to a class call).

   The associated source code will appear in a new editor.

JRockit Mission Control Plug-ins with Jump-to-Source Enabled

Note: This feature only works with versions of the JRockit Mission Control Client integrated into the Eclipse IDE.

Table 1-2 lists the JRockit Mission Control plug-ins where Jump-to-Source is enabled.

Table 1-2 Plug-ins with Jump-to-Source Enabled

Plug-in	Component
Management Console	• Threads tab – Stack traces for selected threads • Exception Counter – Profiling Information table
JRA	• Methods Tab – The tables and both the trees. • GCs Tab – The GC method call tree for a garbage collection. • GC General Tab – Garbage collection call trees. • Objects Tab – Both **Start of Recording** and **End of recording** • Optimizations: – In the table. • Locks – Java Locks • Latency Log: – Event Details – Event Properties – Stack Trace • Latency Log – Event Property Histogram, • Latency Traces – The trace trees.
Memory Leak Detector	• Trend Table • Application Stack Traces