Oracle® JRockit Mission Control

Introduction to Mission Control Client

Release 4.1

E15067-03

December 2011

This document provides background information on the Oracle JRockit Mission Control Client.

ORACLE®

Oracle JRockit Mission Control Introduction to Mission Control Client, Release 4.1

E15067-03

Primary Author: Edwin Spear, Savija Vijayaraghavan

Contents

Preface v

About the Document v
Documentation Accessibility v
Conventions v

1 Introduction to Oracle JRockit Mission Control

1.1 Installation Information 1-1
1.2 Starting the JRockit Mission Control Client 1-1
1.3 JRockit Mission Control FAQ 1-2
1.4 JRockit Mission Control Documentation 1-3
1.5 JRockit Mission Control Support 1-3
1.6 Giving Feedback the JRockit Mission Control Development Team 1-3

2 JRockit Mission Control Client

2.1 Architectural Overview of JRockit Mission Control Client 2-1
2.1.1 The Flight Recorder 2-1
2.1.2 The Management Console 2-1
2.1.3 The JMX Agent 2-2
2.1.4 The Memory Leak Detector 2-2
2.2 Starting JRockit Mission Control 2-2
2.3 The JRockit Browser 2-2
2.4 The JRockit Management Console 2-2
2.5 The JRockit Flight Recorder 2-2
2.6 The JRockit Memory Leak Detector 2-3

3 JRockit Mission Control Communications

3.1 J2SE 5.0 and Later 3-1
3.2 All Versions 3-2

4 Integration with the Eclipse IDE

4.1 Benefits of the Integration 4-1
4.2 Differences between the Eclipse Version and the RCP Version 4-1
4.3 Making the JRockit JVM Your JVM 4-2
4.4 Selecting a Perspective 4-3

4.5 Jumping to Application Source 4-5
4.5.1 Using Jump-to-Source 4-5
4.5.2 JRockit Mission Control Plug-ins with Jump-to-Source Enabled 4-5

5 Accessibility Notes

5.1 Screen Readers 5-1
5.2 JRockit Mission Control Accessibility Mode 5-1
5.2.1 Using the Accessibility Mode 5-2
5.2.2 Showing Text Labels on Graphic Controls 5-2
5.2.3 For Additional Information 5-2
5.3 Workarounds 5-2
5.3.1 Navigating in a Tree Table with Only One Row 5-2
5.3.2 Switching Between Tabs or Pages 5-3
5.3.3 Reading Table Data with a Screen Reader 5-3
5.3.4 Reading Console General or Overview Tab Data 5-3
5.3.5 Resizing Online Help Text 5-3
5.4 Abbreviations Used in the JRockit Mission Control Client 5-3

6 Abbreviations and Acronyms

Preface

This document provides background information on the Oracle JRockit Mission Control Client.

About the Document

This document contains the following chapters:

- Chapter 1, "Introduction to Oracle JRockit Mission Control", which contains introductory information about JRockit Mission Control.
- Chapter 2, "JRockit Mission Control Client", which describes the features in this version of JRockit Mission Control.
- Chapter 3, "JRockit Mission Control Communications", which describes the communication protocols used by JRockit Mission Control and how they differ based upon the J2SE version used.
- Chapter 4, "Integration with the Eclipse IDE", describe how JRockit Mission Control integrates with the Eclipse IDE.
- Chapter 5, "Accessibility Notes", provides accessibility features in this version of JRockit Mission Control.
- Chapter 6, "Abbreviations and Acronyms", defines the acronyms used in the JRockit Mission Control graphical user interface and this documentation.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc`.

Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info` or visit `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs` if you are hearing impaired.

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

1

Introduction to Oracle JRockit Mission Control

This chapter provides an overview about Oracle JRockit Mission Control and its features.

The Oracle JRockit Mission Control Client tools suite was introduced with Oracle JRockit JVM R26.0.0. It includes tools to monitor, manage, profile, and eliminate memory leaks in your Java application without introducing the performance overhead normally associated with these types of tools.

The JRockit Mission Control's low performance overhead is a result of using data collected as part of the JRockit JVM's normal adaptive dynamic optimization. This also eliminates the problem with the Heisenberg anomaly that can occur when tools using byte code instrumentation alters the execution characteristics of the system. The JRockit Mission Control functionality can always be available on-demand and the small performance overhead is only in effect while the tools are running.

This section contains information on the following subjects:

- Section 1.1, "Installation Information"
- Section 1.2, "Starting the JRockit Mission Control Client"
- Section 1.3, "JRockit Mission Control FAQ"
- Section 1.4, "JRockit Mission Control Documentation"
- Section 1.5, "JRockit Mission Control Support"
- Section 1.6, "Giving Feedback the JRockit Mission Control Development Team"

1.1 Installation Information

Installation instructions can be found in the *Oracle JRockit Installation Guide* on the Oracle Technology Network.

1.2 Starting the JRockit Mission Control Client

The JRockit Mission Control Client executable is located in `JROCKIT_HOME/bin`. If this directory is on your system path, you can start JRockit Mission Control by simply typing `jrmc` in a command (shell) prompt.

Otherwise, you have to type the full path to the executable file, as shown below:

```
JAVA_HOME\bin\jrmc.exe (Windows)
JAVA_HOME/bin/jrmc (Linux)
```

On Windows installations, you can start JRockit Mission Control from the **Start** menu.

1.3 JRockit Mission Control FAQ

This topic lists and provides answers to questions frequently asked about JRockit Mission Control.

I cannot install extra plug-ins from the JRockit Mission Control application when a web proxy is configured. How should I start the application when a web proxy is enabled?

Set the proxy properties and start JRockit Mission Control as follows:

```
JAVA_HOME/bin/java -Dhttp.proxyHost=proxyhost
-Dhttp.proxyPort=proxyport [-Dhttp.proxyUser=user Dhttp.proxyPassword=password]
-jar JAVA_HOME/missioncontrol/mc.jar
```

I cannot connect the JRockit Mission Control Client. What could be the problem?

Consider the following:

- Are you using the correct protocol?

  The easiest way is to ensure that you are using the same version of the JRockit JVM you want to monitor as the JVM running the JRockit Mission Control client. If that is not an option, you can use the radio buttons in the connection dialog box in the JRockit Mission Control Client to select which protocol to use: 1.4 will select RMP and 1.5 and later will select JMXRMI.

  For earlier versions of JRockit Mission Control these radio buttons do not exist and, to make a JRockit JVM 5.0 instance connect to a 1.4 version, you must explicitly specify the JMX Service URL. The format of the service URL is:

  ```
  service:jmx:rmp://<hostname>:<port>
  ```

  for example:

  ```
  service:jmx:rmp://localhost:7091
  ```

- Are the correct ports opened?

  JMX over RMI uses two ports and that one of the ports will not be known beforehand.

- Is the communication caught in the firewall?

  See Chapter 2, "JRockit Mission Control Communications" for more information.

When attempting to connect to JRockit Mission Control I get a stack trace indicating that JRockit Mission Control attempts to communicate with a strange IP or host name.

Sometimes RMI can have a problem determining which address to use. This can happen because of

- Access restrictions in the Security manager.
- The machine being multihomed and RMI picking the wrong interface.
- A misconfigured hosts file or a number of different network related configuration problems.

If all else fails you can try specifying the java.rmi.server.hostname system property. Please note that this can affect applications running in the JVM.

I'm getting exceptions during startup about classes not being found

Make sure you are using the proper launcher to start up the JRockit Mission Control Client. You must only use `JAVA_HOME/bin/jrmc`.

JRockit Mission Control can't find any local JVMs

Make sure you are using the proper launcher to start up the JRockit Mission Control Client. You must only use `JAVA_HOME/bin/jrmc`.

Why can't I see any Method Profiling information in my flight recording?

By default, the JRockit Mission Control Client will not show data on tabs if the recording lacks the necessary events. Ensure that method profiling was enabled for your flight recording and that the application was under load. If the JRockit JVM is spending most of the time with threads that are not doing any work, samples will not be recorded.

When using the Memory Leak Detector, nothing happens in the growth column of the trend table

The algorithm needs at least three data points to kick in and the data is collected as part of the old space mark phase of the garbage collection. If you see no data, possibly not enough garbage has been collected for these collections to occur. To speed up the process, try clicking the garbage can in the toolbar of the Memory Leak application to force three successive garbage collections, with a brief pause in between each collection.

1.4 JRockit Mission Control Documentation

Documentation for JRockit Mission Control Client is available as online help with the installation of the tool.

1.5 JRockit Mission Control Support

You are entitled to support if you have an Enterprise licence.

1.6 Giving Feedback the JRockit Mission Control Development Team

If you have any suggestions about how to improve the JRockit Mission Control plug-ins or information on how it is most commonly used in your development environments, post a comment on the JRockit forum. This information would contribute to our understanding on how to best further improve these tools in the future.

To submit an idea or other feedback, post a comment on the Oracle Technology Network's JRockit forum, at
`http://forums.oracle.com/forums/forum.jspa?forumID=561`.

The feedback will be considered by the development team designing the Oracle JRockit Mission Control plug-ins. Oracle will look at collected ideas and improve the plug-ins to make them even easier to use. Oracle's goal with these plug-ins is to simplify the tasks in getting your applications to run as smoothly as possible on the Oracle JRockit JVM.

2

JRockit Mission Control Client

This chapter describes various tools provided with JRockit Mission Control Client.

The Oracle JRockit Mission Control tools suite includes tools to monitor, manage, profile, and eliminate memory leaks in your Java application without introducing the performance overhead normally associated with these types of tools.

This chapter contains information on the following subjects:

- Section 2.1, "Architectural Overview of JRockit Mission Control Client"
- Section 2.2, "Starting JRockit Mission Control"
- Section 2.3, "The JRockit Browser"
- Section 2.4, "The JRockit Management Console"
- Section 2.5, "The JRockit Flight Recorder"
- Section 2.6, "The JRockit Memory Leak Detector"

2.1 Architectural Overview of JRockit Mission Control Client

This section describes what transpires during a typical JRockit Mission Control Client session.

2.1.1 The Flight Recorder

When a flight recording is started from within the JRockit Mission Control Client, it records the status of the JRockit JVM process during the specified time period. The Flight Recorder then creates a file containing the recorded data. The recording file is automatically opened in the JRockit Flight Recorder tool upon completion of the recording, valid for JDK level 1.5 and later. Typical information that is recorded includes Java heap distribution, garbage collections, method samples, thread latency, and lock profiling information."

2.1.2 The Management Console

To view real-time behavior of your application and of Oracle JRockit JVM, you can connect to an instance of the JRockit JVM and view real-time information through the JRockit Management Console. Typical data that you can view is thread usage, CPU usage, and memory usage. All graphs are configurable and you can both add your own attributes and redefine their respective labels. In the Management Console you can also create rules that trigger on certain events, for example, an mail will be sent if the CPU reaches 90% of the size.

2.1.3 The JMX Agent

With the JMX Agent you have access to all MBeans deployed in the platform MBean server. From these MBeans, you can read attribute information, such as garbage collection pause times.

2.1.4 The Memory Leak Detector

To find memory leaks in your Java application, you connect the JRockit Memory Leak Detector to the running JRockit JVM process. The Memory Leak Detector connects to the JMX (RMP) Agent that instructs to start a Memory Leak server where all further communication takes place.

2.2 Starting JRockit Mission Control

The JRockit Mission Control Client executable is located in `JAVA_HOME/bin`. If this directory is on your system path, you can start the JRockit Mission Control Client by simply typing `jrmc` in a command (shell) prompt.

- On Windows: `JAVA_HOME\bin\jrmc.exe`
- On Linux: `JAVA_HOME/bin/jrmc`

2.3 The JRockit Browser

The JRockit Browser was introduced in JRockit Mission Control 2.0. This tool allows you to set up and manage all running instances of JRockit JVM on your system. From the JRockit Browser you activate different tools, such as starting a flight recording, connecting a Management Console, and starting memory leak detection. Each JRockit JVM instance is called a Connector.

2.4 The JRockit Management Console

The JRockit Management Console is used to monitor a JRockit JVM instance. Several Management Consoles can be running concurrently side by side. The tool captures and presents live data about memory, CPU usage, and other runtime metrics. For the Management Console that is connected to JRockit JDK 5.0, information from any JMX MBean deployed in the Oracle JRockit JVM internal MBean server can be displayed as well. For a Console connected to Oracle JRockit JDK 1.4, RMP capabilities are exposed by a JMX proxy. JVM management includes dynamic control over CPU affinity, garbage collection strategy, memory pool sizes, and more.

2.5 The JRockit Flight Recorder

The JRockit Flight Recorder is a performance monitoring and profiling tool that makes diagnostics information always available, even in the wake of catastrophic failure, such as a system crash. At its most basic, JFR is a rotating buffer of diagnostics and profiling data that is always available, on demand. You might consider it a sort “time machine” that enables you to go back in time to gather diagnostics data leading up to an event. The data stored in the rotating buffer includes JVM and application events.

In JRockit Mission Control Client, the Flight Recorder allows users who are running a Flight Recorder-compliant version of the Oracle JRockit JVM (that is, version R28.0.0 or later) to view the JVM’s recordings, current recording settings, and runtime parameters on a series of tabs that aggregate performance data into logical, task-based groups. The data on these tabs is presented by way of an assortment of dials, chart,

and tables. At the top of each tab is a sliding window, called the Range Navigator, with which you can expand or narrow the range of reporting; for example, if you see a group of events clustered around a specific time period, you can adjust the Range Navigator to include just those events, with the resulting data for just those events appearing on the tab components.

For information on the Flight Recorder run time component, see the *Flight Recorder Run Time Guide* on the Oracle Technology Network.

2.6 The JRockit Memory Leak Detector

The JRockit Memory Leak Detector is a tool for discovering and finding the cause for memory leaks in a Java application. The JRockit Memory Leak Detector's trend analyzer discovers slow leaks, it shows detailed heap statistics (including referring types and instances to leaking objects), allocation sites, and it provides a quick drill down to the cause of the memory leak. The Memory Leak Detector uses advanced graphical presentation techniques to make it easier to navigate and understand the sometimes complex information.

3

JRockit Mission Control Communications

This chapter describes the protocols and their differences resultant from the different J2SE versions.

Depending upon which J2SE version on which you are running the Oracle JRockit JVM, certain aspects of the communications protocols will differ.

This section includes information on the following subjects:

- Section 3.1, "J2SE 5.0 and Later"
- Section 3.2, "All Versions"

3.1 J2SE 5.0 and Later

J2SE 5.0 and later versions of the JRockit JDK use JMXRMI (JMX over RMI). This protocol uses one port for the RMI registry, which is configured with the `-Xmanagement:port` option, and a second port (on an anonymous port) for communication with the RMI server. Note that you cannot configure the port for the RMI server; however, you can write your own agent that defines a fixed port for the RMI server. For further information, see "Mimicking Out-of-the-Box Management Using the JMX Remote API" at:

`http://java.sun.com/javase/6/docs/technotes/guides/management/agent.html#gdfvv`

Table 3–1 lists the options available for the `-Xmanagement` flag:

Table 3–1 -Xmanagement Option

Option	Description	Default
`authenticate`	Use password authentication	True
`ssl`	Use secure sockets layer	True
`port`	What port to use for the RMI registry	7091

For a more comprehensive discussion on what these options mean, please see "Monitoring and Management Using JMX" at:

`http://java.sun.com/j2se/1.5.0/docs/guide/management/agent.html`

3.2 All Versions

For all J2SE versions, you can use the `-Xmanagement` option autodiscovery to make the JRockit JVM use the JRockit Discovery Protocol (JDP) to announce its presence; for example `-Xmanagement:autodiscovery=true`.

Table 3–2 lists additional system properties you can use to control the behavior of the JDP server:

Table 3–2 System Properties Used to Control the JDP Server

System property	Description	Default
`jrockit.managementserver.discovery.period`	The time to wait between multicasting the presence in ms	5000
`jrockit.managementserver.discovery.ttl`	The number of router hops the packets being multicasted should survive	1
`jrockit.managementserver.discovery.address`	The multicast group/address to use	232.192.1.212
`jrockit.managementserver.discovery.targetport`	The target port to broadcast	7090(1.4)/7091(1.5)

All versions of JRockit Mission Control also employ an additional protocol when using the Memory Leak Detector. The memory leak server is not written in Java; rather it is an integral part of the JVM. This is because a potential use case for the memory leak server is to optionally be able to start it when an out of memory condition occurs in the JVM. When such a condition occurs, it is impossible to execute Java code because no heap would be available.

MLP (MemLeak Protocol) is used by the native memory leak server during a memory leak session. JRockit Mission Control communicates over RMP (1.4) or JMXRMI (5.0 and higher) to ask the Oracle JRockit JVM to start up the server. You can configure the port on which you want to start the memory leak server on, and to use for the session, by using Oracle JRockit Mission Control preferences.

4

Integration with the Eclipse IDE

This chapter describes the integration of Oracle JRockit Mission Control with Eclipse IDE and provides instructions for using the special functionality enabled by integrating the JRockit Mission Control Client with Eclipse.

In addition to the standalone Rich Client Platform (RCP) version of Oracle JRockit Mission Control 4.0, the toolset is also available as a plug-in to the Eclipse IDE (Eclipse 3.3 or above). This version of 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 toolset in Mission Control.

The chapter contains the following topics:

- Section 4.1, "Benefits of the Integration"
- Section 4.2, "Differences between the Eclipse Version and the RCP Version"
- Section 4.3, "Making the JRockit JVM Your JVM"
- Section 4.4, "Selecting a Perspective"
- Section 4.5, "Jumping to Application Source"

4.1 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.

4.2 Differences between the Eclipse Version and the RCP Version

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 Eclipse version of the JRockit Mission Control Client has over the RCP version is the Jump-to-Source feature, described in Section 4.5, "Jumping to Application Source". 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 Flight Recorder, and the Memory Leak Detector.

4.3 Making the JRockit JVM Your JVM

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

To run Eclipse (and thus the JRockit Mission Control Client) on the JRockit JVM

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 Example 4–1.

Example 4–1 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 non-Oracle JRockit JVMs (for example, `--launcher.XXMaxPermSize 256M`)
   - On the third line down (after org.eclipse.platform), add the following:

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

     The full path to JRockit'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/java
     ```

   - 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* on the Oracle Technology Network

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. Example 4–2 shows an example of the eclipse.ini file updated to make Oracle JRockit JVM the JVM.

Example 4–2 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
```

4.4 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 for Eclipse come with a predefined perspective called JRockit Mission Control. This perspective shows the JRockit Mission Control Client interface so that you can use the tools in JRockit Mission Control to profile applications as you develop them in Eclipse.

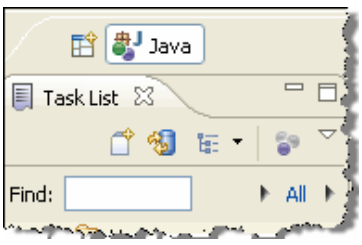
This topic will show you how:

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

To open the Mission Control Perspective

1. In top right corner of the Eclipse window, click the Open Perspective icon (Figure 4–1).

Figure 4–1 Open Perspective Icon



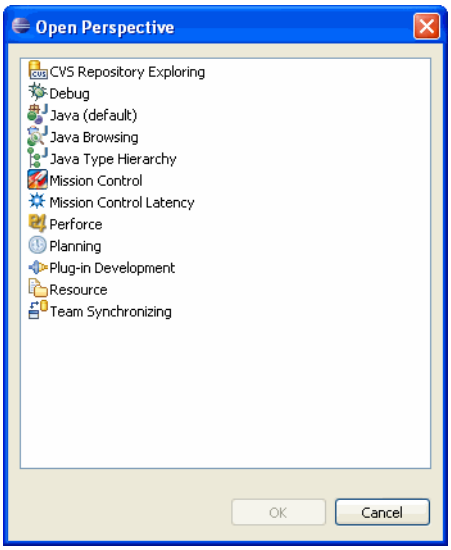
The Open Perspective context menu appears (Figure 4–2).

Figure 4–2 Open Perspective Context Menu

2. Select **Other**.

   The Open Perspective dialog box appears (Figure 4–3).

Figure 4–3 Open Perspective Dialog Box



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

   The Eclipse window reconfigures to show the Mission Control Standard Perspective.

To change perspective from Mission Control

You can change perspectives from Mission Control to another perspective by using one of the methods described in Table 3–1:

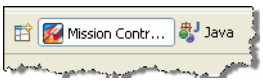
Table 4–1 Changing Perspectives

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 Standard Perspective

If you have already opened the Mission Control Standard 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 4–4).

Figure 4–4 Open Standard Mission Control Perspective Button



To reopen the perspective, simply click that button.

4.5 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 on the Management Console and the Memory Leak Detector:

This topic contains the following information:

- Section 4.5.1, "Using Jump-to-Source"
- Section 4.5.2, "JRockit Mission Control Plug-ins with Jump-to-Source Enabled"

4.5.1 Using Jump-to-Source

To jump from the JRockit Mission Control Client to source code

> **Note:** The following procedure is generic. See Section 4.5.2, "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.

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.

4.5.2 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 3–2 lists the Oracle JRockit Mission Control plug-ins where Jump-to-Source is enabled.

Table 4–2 Plug-ins with Jump-to-Source Enabled

Plug-in	Component
Management Console	Jump-to-source is enabled on: ■ Threads tab ■ Stack traces for selected threads ■ Exception Counter ■ Profiling Information table
Memory Leak Detector	Jump-to-source is enabled on: ■ Trend Table ■ Application Stack Traces

5 Accessibility Notes

This chapter describes the accessibility features provided with Oracle JRockit Mission Control.

Oracle is dedicated to providing high quality information technology that is accessible to people with disabilities. To this end, Oracle has undertaken a substantial project to ensure the accessibility of Oracle JRockit Mission Control Client. Oracle is implementing these enhancements and will continue to address all accessibility issues that come to its attention.

This chapter includes information on the following subjects:

- Section 5.1, "Screen Readers"
- Section 5.2, "JRockit Mission Control Accessibility Mode"
- Section 5.3, "Workarounds"
- Section 5.4, "Abbreviations Used in the JRockit Mission Control Client"

5.1 Screen Readers

Oracle supports a number of different screen readers, technology that translates screen-based information into spoken word to assist vision-impaired users.

Configuration options are currently available for the JAWS screen reader produced by Freedom Scientific, Inc. For information on configuring this product, please refer to the Freedom Scientific screen reader website, at:

`http://www.freedomscientific.com/documentation/screen-readers.asp`

Note: If you are using JAWS, be aware that tab/page switching does not work as expected. Please refer to Section 5.3.2, "Switching Between Tabs or Pages" for a workaround.

5.2 JRockit Mission Control Accessibility Mode

JRockit Mission Control Client displays performance data dials and charts. For most users, these charts provide a valuable graphical view of the data that can reveal trends and help identify minimum and maximum values for performance metrics; however, charts do not convey information in a manner that can be read by a screen reader. To remedy this problem, you can configure JRockit Mission Control Client accessibility mode to provide dial and chart data in tabular format.

Note: When in the accessibility mode, JRockit Mission Control might exhibit behavior that is different from that when running in the normal mode. This is a known issue in this release.

5.2.1 Using the Accessibility Mode

To use the accessibility mode, do the following:

1. In JRockit Mission Control Client, select **Preferences** from the **Window** menu. The Preferences window opens.
2. Select JRockit Mission Control in the left pane. Under **Accessibility Options** in the right pane, select **Use accessibility mode**.
3. Click **Apply**.

5.2.2 Showing Text Labels on Graphic Controls

From the Preferences window, you can configure JRockit Mission Control to show the text label for certain graphical controls. To do so, use this procedure:

1. With JRockit Mission Control Client running, select Windows then Preferences.

   The Preferences window appears.
2. In the Accessibility Options pane, select **Show text labels** on buttons (you will need to restart the console for this change to appear).
3. Click **OK**.

   If you are running the Console, the labels will appear. If you are running the Flight Recorder, you must reopen a recording to see the labels.

Note: If you configure the accessibility mode before actually starting the console, the new configuration will appear when you start the console.

5.2.3 For Additional Information

For important information on using a screen reader to read table data, please refer to Section 5.3.3, "Reading Table Data with a Screen Reader".

5.3 Workarounds

This section contains additional instructions for enhancing your experience with JRockit Mission Control's accessibility features. These instructions include:

5.3.1 Navigating in a Tree Table with Only One Row

When navigating in a tree table component containing only one row, such as the Type Tree and Instance Tree tables in Memleak Detector, user might need to press the Space key or the Shift key and use the Up and Down keys to get to the row.

5.3.2 Switching Between Tabs or Pages

When reaching a tab component in the JRockit Mission Control GUI, JAWS erroneously tells the user “to switch pages, press Ctrl+Tab”. The correct way to switch between tabs or pages is to use the left or right arrow keys.

5.3.3 Reading Table Data with a Screen Reader

To read JRockit Mission Control table data more efficiently with screen reading software, copy and paste the table data into a text editor and read it from there. To copy and paste, do the following:

1. Right-click the table you want to read to open the context menu.
2. Select all items by selecting **Select All**.
3. Select **Copy**.
4. Paste the text in a text editor.

5.3.4 Reading Console General or Overview Tab Data

To enable a screen reader to read General or Overview tab data, each dial and graph section has its own hot-swap control that you can use to switch between graphical and table representation. You can also display labels for graphic controls by using the procedure described in Section 5.2.2, "Showing Text Labels on Graphic Controls".

5.3.5 Resizing Online Help Text

Vision-impaired users might find it difficult to read the online help documents in the standalone (RCP) version of JRockit Mission Control unless the text size is increased. If they need to change the font size, they must view the help in another browser.

JRockit Mission Control will use the default web browser specified in the operating system; you can't specify a different one from within JRockit Mission Control. How you specify a browser in the operating system depends on the version of the operating system.

To change the default viewer for JRockit Mission Control Help, do the following:

1. Select Window then Preferences.

   The Preferences dialog box appears.
2. Under **Specify how help information is displayed**, select **Use external browser**.
3. Click **Apply** or **OK**.

> **Note:** Due to a limitation in the Eclipse help viewer, currently you cannot resize its text. If you need to resize the text the workaround is to use an external browser.

5.4 Abbreviations Used in the JRockit Mission Control Client

JRockit Mission Control Client uses a number of abbreviations and acronyms to save space on the GUI and in the documentation. Wherever possible, these abbreviations and acronyms are spelled out upon first usage; however, that is not always possible. For a list of all common JRockit Mission Control Client abbreviations and acronyms see Chapter 6, "Abbreviations and Acronyms".

6

Abbreviations and Acronyms

This section lists the abbreviations and acronyms used in Oracle JRockit Mission Control Client (Table 6–1).

Table 6–1 Abbreviations and Acronyms Used in Oracle JRockit Mission Control Client

Abbreviation/Acronym	Meaning
GC	Garbage Collection (memory management)
JDK	Java Development Kit
JDP	JRockit Discovery Protocol
JIT	Just In Time (compilation)
JMX	Java Management Extensions
JFR	JRockit Flight Recorder
JVM	Java Virtual Machine
MBean	Managed Bean (Java)
MLP	Memory Leak Protocol
RCP	Rich Client Platform (Eclipse)
RMI	Remote Method Invocation (Java)
RMP	Rockit Management Protocol (communication)
SSL	Secure Sockets Layer (communication)
TLA	Thread Local Area (memory management)