<http://docs.oracle.com/javase/tutorial/jmx/remote/jconsole.html>

Unpack application

Assuming you are inside directory *jmx\_examples* run:

C:\Users\Artur\Downloads\jmx\_examples>java -Dcom.sun.management.jmxremote -Dcom.sun.management.jmxremote.local.only=false -Dcom.sun.management.jmxremote.port=1099 -Dcom.sun.management.jmxremote.ssl=false -Dcom.sun.management.jmxremote.authenticate=false com.example.Main

## **Running the MXBean Example**

The MXBean example uses classes from the [jmx\_examples.zip](http://docs.oracle.com/javase/tutorial/jmx/examples/jmx_examples.zip) bundle that you used in the [Standard MBeans](http://docs.oracle.com/javase/tutorial/jmx/mbeans/standard.html) section. This example requires version 6 of the Java SE platform. To run the MXBeans example follow these steps:

1. If you have not done so already, save [jmx\_examples.zip](http://docs.oracle.com/javase/tutorial/jmx/examples/jmx_examples.zip) into your work\_dir directory.
2. Unzip the bundle of sample classes by using the following command in a terminal window.
3. unzip jmx\_examples.zip
4. Compile the example Java classes from within the work\_dir directory.
5. javac com/example/\*.java
6. Start the Main application. A confirmation that Main is waiting for something to happen is generated.
7. java com.example.Main
8. Start JConsole in a different terminal window on the same machine. The New Connection dialog box is displayed, presenting a list of running JMX agents that you can connect to.
9. jconsole
10. In the New Connection dialog box, select com.example.Main from the list and click Connect.
11. A summary of your platform's current activity is displayed.
12. Click the MBeans tab.
13. This panel shows all the MBeans that are currently registered in the MBean server.
14. In the left frame, expand the com.example node in the MBean tree.
15. You see the example MBean QueueSampler that was created and registered by Main. If you click QueueSampler, you see its associated Attributes and Operations nodes in the MBean tree.
16. Expand the Attributes node.
17. You see the QueueSample attribute appear in the right pane, with its value of javax.management.openmbean.CompositeDataSupport.
18. Double-click the CompositeDataSupport value.
19. You see the QueueSample values date, head, and size because the MXBean framework has converted the QueueSample instance into CompositeData. If you had defined QueueSampler as a standard MBean rather than as an MXBean, JConsole would not have found the QueueSample class because it would not be in its class path. If QueueSampler had been a standard MBean, you would have received a ClassNotFoundException message when retrieving the QueueSample attribute value. The fact that JConsole finds QueueSampler demonstrates the usefulness of using MXBeans when connecting to JMX agents through generic JMX clients such as JConsole.
20. Expand the Operations node.
21. A button to invoke the clearQueue operation is displayed.
22. Click the clearQueue button.
23. A confirmation that the method was invoked successfully is displayed.
24. Expand the Attributes node again, and double click on the CompositeDataSupport value.
25. The head and size values have been reset.
26. To close JConsole, select Connection -> Exit.

# **Exposing a Resource for Remote Management By JConsole**

Exposing your Java applications for remote management by using the JMX API can be extremely simple, if you use the out-of-the-box remote management agent and an existing monitoring and management tool such as JConsole.

To expose your application for remote management, you need to start it with the correct properties. This example shows how to expose the [Main](http://docs.oracle.com/javase/tutorial/jmx/examples/Main.java) JMX agent for remote management.

**Security consideration:**

For the sake of simplicity, the authentication and encryption security mechanisms are disabled in this example. However, you should implement these security mechanisms when implementing remote management in real-world environments. [What Next?](http://docs.oracle.com/javase/tutorial/jmx/end.html) provides pointers to other JMX technology documentation that shows how to activate security.

This example requires version 6 of the Java SE platform. To monitor the Main JMX agent remotely, follow these steps:

1. If you have not done so already, save [jmx\_examples.zip](http://docs.oracle.com/javase/tutorial/jmx/examples/jmx_examples.zip) into your work\_dir directory.
2. Unzip the bundle of sample classes by using the following command in a terminal window.
3. unzip jmx\_examples.zip
4. Compile the example Java classes from within the work\_dir directory.
5. javac com/example/\*.java
6. Start the Main application, specifying the properties that expose Main for remote management. (For Windows, use the carat (^) instead of the backslash (\) to break up a long command into multiple lines):
7. java -Dcom.sun.management.jmxremote.port=9999 \  
    -Dcom.sun.management.jmxremote.authenticate=false \  
    -Dcom.sun.management.jmxremote.ssl=false \  
    com.example.Main
8. A confirmation that Main is waiting for something to happen is generated.
9. Start JConsole in a different terminal window on a **different machine**:
10. jconsole
11. The New Connection dialog box is displayed, presenting a list of running JMX agents that you can connect to locally.
12. Select Remote Process, and type the following in the Remote Process field:
13. *hostname*:9999
14. In this address, hostname is the name of the remote machine on which the Main application is running and 9999 is the number of the port on which the out-of-the-box JMX connector will be connected.
15. Click Connect.
16. A summary of the current activity of the Java Virtual Machine (Java VM) in which Main is running is displayed.
17. Click the MBeans tab.
18. This panel shows all the MBeans that are currently registered in the remote MBean server.
19. In the left-hand frame, expand the com.example node in the MBean tree.
20. You see the example MBean Hello that was created and registered by Main. If you click Hello, you see its associated Attributes and Operations nodes in the MBean tree, even though it is running on a different machine.
21. To close JConsole, select Connection -> Exit.