Red Hat JBoss Enterprise Application Platform 7.4

Performance Tuning Guide

Instructions for evaluating Red Hat JBoss Enterprise Application Platform performance, and for configuring updates to improve performance.
Instructions for evaluating Red Hat JBoss Enterprise Application Platform performance, and for configuring updates to improve performance.
Abstract

This book is a guide of performance tuning for Red Hat JBoss Enterprise Application Platform.
# Table of Contents

PROVIDING FEEDBACK ON RED HAT DOCUMENTATION .................................................. 4

MAKING OPEN SOURCE MORE INCLUSIVE ................................................................. 5

CHAPTER 1. INTRODUCTION TO PERFORMANCE TUNING ............................................. 6
  1.1. ABOUT THE USE OF EAP_HOME IN THIS DOCUMENT ......................................... 6

CHAPTER 2. MONITORING PERFORMANCE ................................................................. 7
  2.1. CONFIGURING JBOSS EAP FOR REMOTE MONITORING CONNECTIONS ............. 7
  2.2. JCONSOLE
     2.2.1. Connecting to a Local JBoss EAP JVM Using JConsole ............................ 9
     2.2.2. Connecting to a Remote JBoss EAP JVM Using JConsole ......................... 10
  2.3. JAVA VISUALVM
     2.3.1. Connecting to a Local JBoss EAP JVM Using VisualVM ......................... 12
     2.3.2. Connecting to a Remote JBoss EAP JVM Using VisualVM ..................... 13

CHAPTER 3. DIAGNOSING PERFORMANCE ISSUES ....................................................... 15
  3.1. ENABLING GARBAGE COLLECTION LOGGING ..................................................... 15
  3.2. JAVA HEAP DUMPS
     3.2.1. Creating a Heap Dump
           3.2.1.1. OpenJDK and Oracle JDK ..................................................... 16
           3.2.1.2. IBM JDK .................................................................. 16
     3.2.2. Analyzing a Heap Dump ................................................................. 16
  3.3. IDENTIFYING HIGH CPU UTILIZATION BY JAVA THREADS .................................. 17
  3.4. RUNTIME STATISTICS FOR MANAGED EXECUTOR SERVICES AND MANAGED SCHEDULED EXECUTOR SERVICES ......................................................... 17

CHAPTER 4. JVM TUNING ......................................................................................... 20
  4.1. SETTING UP A FIXED HEAP SIZE ..................................................................... 20
  4.2. CONFIGURING THE GARBAGE COLLECTOR
     Garbage Collection Logging Options .................................................................. 20
  4.3. ACTIVATING LARGE PAGES ............................................................................ 21
  4.4. ACTIVATING AGGRESSIVE OPTIMIZATIONS .................................................... 22
  4.5. SETTING SOFT AND HARD ULIMITS ............................................................. 22
  4.6. HOST AND PROCESS CONTROLLER JVM TUNING ........................................... 23

CHAPTER 5. JAKARTA ENTERPRISE BEANS SUBSYSTEM TUNING ......................... 24
  5.1. BEAN INSTANCE POOLS
     5.1.1. Creating a Bean Instance Pool ............................................................... 24
     5.1.2. Specifying the Instance Pool a Bean Should Use ...................................... 25
     5.1.3. Disabling the Default Bean Instance Pool .............................................. 25
  5.2. BEAN THREAD POOLS
     5.2.1. Creating a Bean Thread Pool .................................................................. 26
     5.2.2. Configuring Enterprise Bean Services to Use a Specific Bean Thread Pool .... 26
  5.3. RUNTIME BEAN DEPLOYMENT INFORMATION
     5.3.1. Command line options for retrieving runtime data from Jakarta Enterprise Beans .......................................................... 27
  5.4. EXCEPTIONS THAT INDICATE AN ENTERPRISE BEAN SUBSYSTEM TUNING MIGHT BE REQUIRED ........................................................... 29
  5.5. DEFAULT GLOBAL TIMEOUT VALUES FOR SFSBS ........................................... 30

CHAPTER 6. DATASOURCE AND RESOURCE ADAPTER TUNING .............................. 32
  6.1. MONITORING POOL STATISTICS
     6.1.1. Datasource Statistics ............................................................................. 32
     6.1.1.1. Enabling Datasource Statistics ......................................................... 32
Enable Datasource Statistics Using the Management CLI
Enable Datasource Statistics Using the Management Console
6.1.2. Viewing Datasource Statistics
View Datasource Statistics Using the Management CLI
View Datasource Statistics Using the Management Console
6.1.2. Resource Adapter Statistics
Enable Resource Adapter Statistics
View Resource Adapter Statistics
6.2. POOL ATTRIBUTES
6.3. CONFIGURING POOL ATTRIBUTES
6.3.1. Configuring Datasource Pool Attributes
6.3.2. Configuring Resource Adapter Pool Attributes

CHAPTER 7. MESSAGING SUBSYSTEM TUNING

CHAPTER 8. LOGGING SUBSYSTEM TUNING
8.1. DISABLING LOGGING TO THE CONSOLE
8.2. CONFIGURING LOGGING LEVELS
8.3. CONFIGURING THE LOCATION OF THE LOG FILES

CHAPTER 9. UNDERTOW SUBSYSTEM TUNING
9.1. BUFFER CACHE CONFIGURATION
9.2. BYTE BUFFER POOLS CONFIGURATION
9.3. JAKARTA SERVER PAGES CONFIGURATION
9.3.1. Enabling Jakarta Server Pages options using the management console
9.3.2. Enabling Jakarta Server Pages options using the management CLI
9.4. LISTENER CONFIGURATION OPTIONS
9.4.1. Configuring listener options using the management console
9.4.2. Configuring listener options using the management CLI

CHAPTER 10. IO SUBSYSTEM TUNING
10.1. CONFIGURING WORKERS
10.1.1. Monitoring Worker Statistics
10.2. CONFIGURING BUFFER POOLS

CHAPTER 11. JGROUPS SUBSYSTEM TUNING
11.1. MONITORING JGROUPS STATISTICS
11.2. NETWORKING AND JUMBO FRAMES
11.3. MESSAGE BUNDLING
11.4. JGROUPS THREAD POOLS
11.5. JGROUPS SEND AND RECEIVE BUFFERS

CHAPTER 12. TRANSACTIONS SUBSYSTEM TUNING

APPENDIX A. REFERENCE MATERIAL
A.1. DATASOURCE STATISTICS
A.2. RESOURCE ADAPTER STATISTICS
A.3. IO SUBSYSTEM ATTRIBUTES
PROVIDING FEEDBACK ON RED HAT DOCUMENTATION

We appreciate your feedback on our documentation. To provide feedback, you can highlight the text in a document and add comments. Follow the steps in the procedure to learn about submitting feedback on Red Hat documentation.

Prerequisites

- Log in to the Red Hat Customer Portal.
- In the Red Hat Customer Portal, view the document in Multi-page HTML format.

Procedure

1. Click Feedback to see existing reader comments.

   NOTE
   The feedback feature is enabled only in the Multi-page HTML format.

2. Highlight the section of the document where you want to provide feedback.

3. In the prompt menu that displays near the text you selected, click Add Feedback.
   A text box opens in the feedback section on the right side of the page.

4. Enter your feedback in the text box and click Submit.
   You have created a documentation issue.

5. To view the issue, click the issue tracker link in the feedback view.
MAKING OPEN SOURCE MORE INCLUSIVE

Red Hat is committed to replacing problematic language in our code, documentation, and web properties. We are beginning with these four terms: master, slave, blacklist, and whitelist. Because of the enormity of this endeavor, these changes will be implemented gradually over several upcoming releases. For more details, see our CTO Chris Wright’s message.
CHAPTER 1. INTRODUCTION TO PERFORMANCE TUNING

A JBoss EAP installation is optimized by default. However, configurations to your environment, applications, and use of JBoss EAP subsystems can impact performance, meaning additional configuration might be needed.

This guide provides optimization recommendations for common JBoss EAP use cases, as well as instructions for monitoring performance and diagnosing performance issues.

You should stress test and verify all performance configuration changes under anticipated conditions in a development or testing environment prior to deploying them to production.

1.1. ABOUT THE USE OF EAP_HOME IN THIS DOCUMENT

In this document, the variable EAP_HOME is used to denote the path to the JBoss EAP installation. Replace this variable with the actual path to your JBoss EAP installation.

- If you installed the JBoss EAP compressed file, the install directory is the jboss-eap-7.4 directory where you extracted the compressed archive.

- If you installed JBoss EAP using the RPM install method, the install directory is /opt/rh/eap7/root/usr/share/wildfly/.

- If you used the installer to install JBoss EAP, the default path for EAP_HOME is ${user.home}/EAP-7.4.0:
  - For Red Hat Enterprise Linux and Solaris: /home/USER_NAME/EAP-7.4.0/
  - For Microsoft Windows: C:\Users\USER_NAME\EAP-7.4.0\n
- If you used the Red Hat CodeReady Studio installer to install and configure the JBoss EAP server, the default path for EAP_HOME is ${user.home}/devstudio/runtimes/jboss-eap:
  - For Red Hat Enterprise Linux: /home/USER_NAME/devstudio/runtimes/jboss-eap/
  - For Microsoft Windows: C:\Users\USER_NAME\devstudio\runtimes\jboss-eap or C:\Documents and Settings\USER_NAME\devstudio\runtimes\jboss-eap\n
NOTE

If you set the Target runtime to 7.4 or a later runtime version in Red Hat CodeReady Studio, your project is compatible with the Jakarta EE 8 specification.

NOTE

EAP_HOME is not an environment variable. JBOSS_HOME is the environment variable used in scripts.
CHAPTER 2. MONITORING PERFORMANCE

You can monitor JBoss EAP performance using any tool that can examine JVMs running on your machine. Red Hat recommends that you use either JConsole, for which JBoss EAP includes a preconfigured wrapper script, or Java VisualVM. Both these tools provide basic monitoring of JVM processes, including memory usage, thread utilization, loaded classes, and other JVM metrics.

If you will be running one of these tools locally on the same machine that JBoss EAP is running on, then no configuration is necessary. However, if you will be running one of these tools to monitor JBoss EAP running on a remote machine, then some configuration is required for JBoss EAP to accept remote JMX connections.

2.1. CONFIGURING JBOSS EAP FOR REMOTE MONITORING CONNECTIONS

For a Standalone Server

1. Ensure that you have created a management user. You might want to create a separate management user to monitor your JBoss EAP server. See the JBoss EAP Configuration Guide for details.

2. When starting JBoss EAP, bind the management interface to the IP address that you will use to remotely monitor the server:

   ```
   $ EAP_HOME/bin/standalone.sh -bmanagement=IP_ADDRESS
   ```

   **WARNING**
   This exposes all the JBoss EAP management interfaces, including the management console and management CLI, to the specified network. Ensure that you only bind the management interface to a private network.

3. Use the following URI with your management user name and password in your JVM monitoring tool to connect to the JBoss EAP server. The URI below uses the default management port (9990).

   ```
   service:jmx:remote+http://IP_ADDRESS:9990
   ```

For a Managed Domain Host

Using the above procedure of binding the management interface on a managed domain host will only expose the host controller JVM for remote monitoring, and not the individual JBoss EAP servers running on that host.

To configure JBoss EAP to remotely monitor individual servers on a managed domain host, follow the procedure below.

1. Create a new user in the ApplicationRealm that you will use to connect to the JBoss EAP servers for remote monitoring. See the JBoss EAP Configuration Guide for details.
2. In the management CLI, run the following commands to add a remoting port to the socket binding group, and add remoting to the ApplicationRealm. If necessary, replace the profile name and socket binding group in the following commands with the ones that you are using.

```
/profile=full/subsystem=jmx/remoting-connector=jmx:add(use-management-endpoint=false)
/socket-binding-group=full-sockets/socket-binding=remoting:add(port=4447)
/profile=full/subsystem=remoting/connector=remoting-connector:add(socket-binding=remoting,security-realm=ApplicationRealm)
```

3. When starting your JBoss EAP managed domain host, bind one or both of the following interfaces to an IP address that you will use for monitoring.

- If you want to connect to individual JBoss EAP server JVMs running on your managed domain host, bind the public interface:

```
$ EAP_HOME/bin/domain.sh -b=IP_ADDRESS
```

- If you want to connect to the JBoss EAP host controller JVM, also bind the management interface:

```
$ EAP_HOME/bin/domain.sh -bmanagement=IP_ADDRESS
```

**WARNING**

This exposes all the JBoss EAP management interfaces, including the management console and management CLI, to the specified network. Ensure that you only bind the management interface to a private network.

4. Use the following details in your JVM monitoring tool:

- To connect to individual JBoss EAP server JVMs running on your managed domain host, use the following URI with your ApplicationRealm user name and password that was created earlier.

```
service:jmx:remote://IP_ADDRESS:4447
```

To connect to different JBoss EAP servers on a single host, add the respective server’s port offset value to the above port number.

- To connect to the JBoss EAP host controller JVM, use the following URI with a management user name and password.

```
service:jmx:remote://IP_ADDRESS:9990
```

### 2.2. Jconsole

A preconfigured JConsole wrapper script is bundled with JBoss EAP. Using this wrapper script ensures...
A preconfigured JConsole wrapper script is bundled with JBoss EAP. Using this wrapper script ensures that all the required libraries are added to the class path, and also provides access to the JBoss EAP management CLI from within JConsole.

2.2.1. Connecting to a Local JBoss EAP JVM Using JConsole

To connect to a JBoss EAP JVM running on the same machine as JConsole:

1. Run the `jconsole` script in `EAP_HOME/bin`.

2. Under **Local Process**, select the JBoss EAP JVM process that you want to monitor.
   - For a standalone JBoss EAP server, there is one JBoss EAP JVM process.

A JBoss EAP managed domain host has multiple JVM processes you can connect to: a host controller JVM process, a process controller JVM process, and a JVM process for each JBoss EAP server on the host. You can determine which JVM you have connected to by...
looking at the JVM arguments.

**Figure 2.2. JConsole Local Managed Domain JBoss EAP JVMs**

3. Click **Connect**.

### 2.2.2. Connecting to a Remote JBoss EAP JVM Using JConsole

#### Prerequisites

- Configure JBoss EAP for remote monitoring connections.
- Download and extract a ZIP installation of JBoss EAP to your local machine. See the [JBoss EAP Installation Guide](#) for details.
1. Run the `jconsole` script in `EAP_HOME/bin`.

2. Under Remote Process, insert the URI for the remote JBoss EAP JVM process that you want to monitor. See the instructions on configuring JBoss EAP for remote monitoring connections for the URI to use.

   **Figure 2.3. JConsole Remote JBoss EAP JVM**

3. Ensure that you provide the user name and password for the monitoring connection.

4. Click Connect.

2.3. JAVA VISUALVM

Java VisualVM is included with the Oracle JDK, and is located at `JAVA_HOME/bin/jvisualvm`. If you are not using the Oracle JDK, VisualVM is also available for download from the VisualVM website. Note that VisualVM does not work with the IBM JDK.
The following sections provide instructions for using VisualVM to connect to a local or remote JBoss EAP JVM. See the VisualVM documentation for other information on using VisualVM.

2.3.1. Connecting to a Local JBoss EAP JVM Using VisualVM

To connect to a JBoss EAP JVM running on the same machine as VisualVM:

1. Open VisualVM, and find the Applications pane on the left side of the VisualVM window.

2. Under Local, double-click the JBoss EAP JVM process that you want to monitor.

   • For a standalone JBoss EAP server, there is one JBoss EAP JVM process.

   Figure 2.4. VisualVM Local Standalone JBoss EAP Server JVM

   ![VisualVM Local Standalone JBoss EAP Server JVM](image)

   • A JBoss EAP managed domain host has multiple JVM processes you can connect to: a host controller JVM process, a process controller JVM process, and a JVM process for each JBoss EAP server on the host. You can determine which JVM you have connected to by looking at the JVM arguments.
2.3.2. Connecting to a Remote JBoss EAP JVM Using VisualVM

Prerequisites

- Configure JBoss EAP for remote monitoring connections.

- Download and extract a ZIP installation of JBoss EAP to your local machine. See the JBoss EAP Installation Guide for details.

1. You must add the required JBoss EAP libraries to your class path to remotely monitor a JBoss EAP JVM. Start VisualVM with the arguments for required libraries on your local machine. For example:

   ```
   $ visualvm -cp:a EAP_HOME/bin/client/jboss-cli-client.jar -J-Dmodule.path=EAP_HOME/modules
   ```

2. In the File menu, select Add JMX Connection.

3. Complete the details for your remote JBoss EAP JVM:

   - In the Connection field, insert the URI for the remote JBoss EAP JVM process that you want to monitor. See the instructions on configuring JBoss EAP for remote monitoring connections for the URI to use.

   - Select the Use security credentials check box, and enter the user name and password for the monitoring connection.
• If you are not using an SSL connection, select the Do not require SSL connection check box.

**Figure 2.6. VisualVM Remote JBoss EAP JVM**

4. Click **OK**.

5. In the **Applications** pane on the left side of the VisualVM window, double-click on the JMX item under the remote host to open the monitoring connection.
CHAPTER 3. DIAGNOSING PERFORMANCE ISSUES

3.1. ENABLING GARBAGE COLLECTION LOGGING

Examining garbage collection logs can be useful when attempting to troubleshoot Java performance issues, especially those related to memory usage.

Other than some additional disk I/O activity for writing the log files, enabling garbage collection logging does not significantly affect server performance.

Garbage collection logging is already enabled by default for a standalone JBoss EAP server running on OpenJDK or Oracle JDK. For a JBoss EAP managed domain, garbage collection logging can be enabled for the host controller, process controller, or individual JBoss EAP servers.

1. Get the correct JVM options for enabling garbage collection logging for your JDK. Replace the path in the options below to where you want the log to be created.

   **NOTE**
   
   The Red Hat Customer Portal has a JVM Options Configuration Tool that can help you generate optimal JVM settings.

   - For OpenJDK 8 or Oracle JDK 8:
     
     ```
     ```

   - For versions 9 or later of OpenJDK, Oracle JDK, or any JDK that supports JEP 271:
     
     ```
     ```

   - For IBM JDK:
     
     ```
     -Xverbosegclog:<path_to_directory>/gc.log
     ```

2. Apply the garbage collection JVM options to your JBoss EAP server.

   See the JBoss EAP Configuration Guide for instructions on how to apply JVM options to a standalone server or servers in a managed domain.

Additional resources

- For more information about JEP 271, see JEP 271: Unified GC Logging on the OpenJDK web page.

3.2. JAVA HEAP DUMPS

A Java heap dump is a snapshot of a JVM heap created at a certain point in time. Creating and analyzing heap dumps can be useful for diagnosing and troubleshooting issues with Java applications.
Depending on which JDK you are using, there are different ways of creating and analyzing a Java heap dump for a JBoss EAP process. This section covers common methods for Oracle JDK, OpenJDK, and IBM JDK.

### 3.2.1. Creating a Heap Dump

#### 3.2.1.1. OpenJDK and Oracle JDK

**Create an On-Demand Heap Dump**

You can use the `jcmd` command to create an on-demand heap dump for JBoss EAP running on OpenJDK or Oracle JDK.

1. Determine the process ID of the JVM that you want to create a heap dump from.
2. Create the heap dump with the following command:

   ```bash
   $ jcmd JAVA_PID GC.heap_dump -all=true FILENAME.hprof
   ```

   This creates a heap dump file in the HPROF format, usually located in `EAP_HOME` or `EAP_HOME/bin`. Alternatively, you can specify a file path to another directory.

**Create a Heap Dump Automatically on OutOfMemoryError**

You can use the `-XX:+HeapDumpOnOutOfMemoryError` JVM option to automatically create a heap dump when an `OutOfMemoryError` exception is thrown.

This creates a heap dump file in the HPROF format, usually located in `EAP_HOME` or `EAP_HOME/bin`. Alternatively, you can set a custom path for the heap dump using `-XX:HeapDumpPath=PATH`. If you specify a file name using `-XX:HeapDumpPath`, for example, `-XX:HeapDumpPath=PATH/filename.hprof`, the heap dumps will overwrite each other.

See the JBoss EAP Configuration Guide for instructions on how to apply JVM options to a standalone server or servers in a managed domain.

#### 3.2.1.2. IBM JDK

When using the IBM JDK, heap dumps are automatically generated when an `OutOfMemoryError` is thrown.

Heap dumps from the IBM JDK are saved in the `/tmp/` directory as a portable heap dump (PHD) formatted file.

### 3.2.2. Analyzing a Heap Dump

#### Heap Dump Analysis Tools

There are many tools that can analyze heap dump files and help identify issues. Red Hat Support recommends using the Eclipse Memory Analyzer tool (MAT), which can analyze heap dumps formatted in either HPROF or PHD formats.

For information on using Eclipse MAT, see the Eclipse MAT documentation.

#### Heap Dump Analysis Tips

Sometimes the cause of the heap performance issues are obvious, but other times you may need an
understanding of your application’s code and the specific circumstances that cause issues like an
OutOfMemoryError. This can help to identify whether an issue is a memory leak, or if the heap is just not
large enough.

Some suggestions for identifying memory usage issues include:

- If a single object is not found to be consuming too much memory, try grouping by class to see if
  many small objects are consuming a lot of memory.

- Check if the biggest usage of memory is a thread. A good indicator of this is if the
  OutOfMemoryError-triggered heap dump is much smaller than the specified Xmx maximum
  heap size.

- A technique to make memory leaks more detectable is to temporarily double the normal
  maximum heap size. When an OutOfMemoryError occurs, the size of the objects related to the
  memory leak will be about half the size of the heap.

When the source of a memory issue is identified, you can view the paths from garbage collection roots to
see what is keeping the objects alive.

### 3.3. IDENTIFYING HIGH CPU UTILIZATION BY JAVA THREADS

**NOTE**

For customers using JBoss EAP on Red Hat Enterprise Linux or Solaris, the JVMPeg lab
tool on the Red Hat Customer Portal helps collect and analyze Java thread information to
identify high CPU utilization. Follow the instructions for using the JVMPeg lab tool
instead of using the following procedure.

For OpenJDK and Oracle JDK environments, Java thread diagnostic information is available using the
jstack utility.

1. Identify the process ID of the Java process that is utilizing a high percentage of the CPU.
   It can also be useful to obtain per-thread CPU data on high-usage processes. This can be done
   using the top -H command on Red Hat Enterprise Linux systems.

2. Using the jstack utility, create a stack dump of the Java process. For example, on Linux and
   Solaris:

   ```
   jstack -l JAVA_PROCESS_ID > high-cpu-tdump.out
   ```

   You might need to create multiple dumps at intervals to see any changes or trends over a
   period of time.

3. Analyze the stack dumps. You can use a tool such as the Thread Dump Analyzer (TDA).

### 3.4. RUNTIME STATISTICS FOR MANAGED EXECUTOR SERVICES AND
MANAGED SCHEDULED EXECUTOR SERVICES

You can monitor the performance of managed executor services and managed scheduled executor
services by viewing the runtime statistics generated with the management CLI attributes. You can view
the runtime statistics for a standalone server or for an individual server mapped to a host.
IMPORTANT

The domain.xml configuration does not include a resource for the runtime statistic management CLI attributes, so you cannot use the management CLI attributes to view the runtime statistics for a managed domain.

Table 3.1. Displays management CLI attributes for monitoring the performance of managed executor services and of managed scheduled executor services.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>active-thread-count</td>
<td>The approximate number of threads that are actively executing tasks.</td>
</tr>
<tr>
<td>completed-task-count</td>
<td>The approximate total number of tasks that have completed execution.</td>
</tr>
<tr>
<td>hung-thread-count</td>
<td>The number of executor threads that are hung.</td>
</tr>
<tr>
<td>max-thread-count</td>
<td>The largest number of executor threads.</td>
</tr>
<tr>
<td>current-queue-size</td>
<td>The current size of the executor’s task queue.</td>
</tr>
<tr>
<td>task-count</td>
<td>The approximate total number of tasks that have been submitted for execution.</td>
</tr>
<tr>
<td>thread-count</td>
<td>The current number of executor threads.</td>
</tr>
</tbody>
</table>

Example of viewing the runtime statistics for a managed executor service running on a standalone server.

```
[standalone@localhost:9990 /] /subsystem=ee/managed-executor-service=default:read-resource(include-runtime=true,recursive=true)
```

Example of the runtime statistics for a managed scheduled executor service running on a standalone server.

```
[standalone@localhost:9990 /] /subsystem=ee/managed-scheduled-executor-service=default:read-resource(include-runtime=true,recursive=true)
```

Example of viewing the runtime statistics for a managed executor service running on a server mapped to a host.

```
[domain@localhost:9990 /] /host=<host_name>/server=<server_name>/subsystem=ee/managed-executor-service=default:read-resource(include-runtime=true,recursive=true)
```

Example of the runtime statistics for a managed scheduled executor service running on a server mapped to a host.
[domain@localhost:9990 /host=<host_name>/server=<server_name>/subsystem=ee/managed-scheduled-executor-service=default:read-resource(include-runtime=true,recursive=true)

Additional resources

- For information about creating a managed executor service, see Managed Executor Service in the JBoss EAP Development Guide.

- For information about creating a managed scheduled executor service, see Managed Scheduled Executor Service in the JBoss EAP Development Guide.
CHAPTER 4. JVM TUNING

Configuring optimal Java virtual machine (JVM) options for your applications and JBoss EAP environment is one of the most fundamental ways to tune performance. This chapter covers configuring some general JVM options.

NOTE

Many of the JVM options listed in this chapter can be easily generated using the JVM Options Configuration Tool on the Red Hat Customer Portal.

Additional resources

- Refer to the JBoss EAP Configuration Guide for instructions on how to apply JVM options to a standalone server or servers in a managed domain.

4.1. SETTING UP A FIXED HEAP SIZE

To pre-allocate and fix the heap size on the production environments, you must set the initial and maximum heap size options to the same size.

Procedure

1. Set an appropriate heap size to prevent out of memory errors.
   a. Use the -Xms option to set the initial heap size and the -Xmx to set the maximum heap size. For example, the following options set a 2048 MB heap size:

   -Xms2048M -Xmx2048M

2. Test your applications under load in a development environment to determine the maximum memory usage.

Your production heap size should be at least 25% higher than the tested maximum to allow room for overhead.

4.2. CONFIGURING THE GARBAGE COLLECTOR

Although the parallel garbage collector, also known as the throughput garbage collector, is the default garbage collector in Java 8 for server-class machines, Red Hat recommends using the G1 garbage collector, which is expected to be the default from Java 9 onward. The G1 garbage collector generally performs better than the CMS and parallel garbage collectors in most scenarios.

Procedure

1. To enable the G1 collector, use the following JVM option:

   -XX:+UseG1GC

Garbage Collection Logging Options

Garbage collection logging is enabled by default for standalone JBoss EAP servers. To enable garbage collection logging for a JBoss EAP managed domain, see enabling garbage collection.
4.3. ACTIVATING LARGE PAGES

Activating large pages for JBoss EAP JVMs results in pages that are locked in memory and cannot be swapped to disk like regular memory.

Especially for memory-intensive applications, the advantage of using large pages is that the heap cannot be paged or swapped to disk, and is thus always readily available.

One disadvantage of using large pages is that other processes running on the system might not have quick access to memory, which might result in excessive paging for these processes.

As with any other performance configuration change, it is recommended that you test the impact of the change in a testing environment.

Prerequisites

- Your operating system configuration is set to use large pages.

Procedure

1. If your operating system is not configured to use large pages for JBoss EAP processes, select one of the following options:
   - For Red Hat Enterprise Linux systems, you must explicitly configure HugeTLB pages to guarantee that JBoss EAP processes will have access to large pages. For information on configuring Red Hat Enterprise Linux memory options, see the Memory chapter in the Red Hat Enterprise Linux Performance Tuning Guide.
   - For Windows Server systems running JBoss EAP, you must assign the large pages privilege:
     2. Select Local Policies → User Rights Assignment.
     3. Double-click Lock pages in memory.
     4. Add the Windows Server users and user groups that you want to use large pages.
     5. Restart the machine.

2. Enable or disable large page support:
   - To explicitly enable large page support for JBoss EAP JVMs, use the following JVM option:
     ```
     -XX:+UseLargePages
     ```
   - To explicitly disable large page support for JBoss EAP JVMs, use the following JVM option:
     ```
     -XX:-UseLargePages
     ```

3. When starting JBoss EAP, ensure that there are no warnings related to reserving memory.
   - On Red Hat Enterprise Linux, an error might look like:
     ```
     OpenJDK 64-Bit Server VM warning: Failed to reserve shared memory. (error = 1)
     ```
On Windows Server, an error might look like:

Java HotSpot(TM) 64-Bit Server VM warning: JVM cannot use large page memory because it does not have enough privilege to lock pages in memory.

If you do see warnings, verify that your operating system configuration and JVM options are configured correctly.

Additional resources

- For more information, see the Oracle documentation on Java support for large pages.

### 4.4. ACTIVATING AGGRESSIVE OPTIMIZATIONS

Using the aggressive optimizations (AggressiveOpts) JVM option can provide performance improvements for your environment. This option provides Java performance optimization features that are expected to become default in future Java releases.

**Procedure**

1. To enable AggressiveOpts, use the following JVM option:

   ```
   -XX:+AggressiveOpts
   ```

### 4.5. SETTING SOFT AND HARD ULIMITS

For Red Hat Enterprise Linux and Solaris platforms, you must configure appropriate `ulimit` values for JBoss EAP JVM processes. The "soft" `ulimit` can be temporarily exceeded, while the "hard" `ulimit` is the strict ceiling for the usage of a resource. Appropriate `ulimit` values vary depending on your environment and applications.

**IMPORTANT**

If you are using IBM JDK, it is important to note that IBM JDK uses the soft limit for the maximum number of open files used by a JVM process. On Red Hat Enterprise Linux, the default soft limit (1024) is considered too low for JBoss EAP processes using IBM JDK.

If the limits applied to JBoss EAP processes are too low, you will see a warning like the following when starting JBoss EAP:

```
WARN  [org.jboss.as.warn.fd-limit] (main) WFLYSRV0071: The operating system has limited the number of open files to 1024 for this process; a value of at least 4096 is recommended.
```

**Procedure**

1. To see your current `ulimit` values, use the following commands:

   - For soft `ulimit` values:
     ```
     ulimit -Sa
     ```

   - For hard `ulimit` values:
To set the `ulimit` for the maximum number of open files, use the following commands with the number you want to apply:

- To set the soft `ulimit` for the maximum number of open files:
  ```
  ulimit -Sn 4096
  ```

- To set the hard `ulimit` for the maximum number of open files:
  ```
  ulimit -Hn 4096
  ```

**NOTE**

To guarantee that a `ulimit` setting is effective, it is recommended on production systems to set the soft and hard limits to the same value.

Additional resources

For more information on setting `ulimit` values using a configuration file, see [How to set ulimit values](#) on the Customer Portal.

### 4.6. HOST AND PROCESS CONTROLLER JVM TUNING

JBoss EAP managed domain hosts have separate JVMs for the host controller and process controller. See the JBoss EAP [Configuration Guide](#) for more information on the roles of [host controllers](#) and [process controllers](#).

You can tune the host controller and process controller JVM settings, but even for large managed domain environments, the default JVM configuration for the host controller and process controller should suffice.

The default configurations for host controller and process controller JVMs have been tested with a managed domain size of up to 20 JBoss EAP hosts each running 10 JBoss EAP servers, for a total domain size of 200 JBoss EAP servers.

If you experience issues with larger managed domains, you might need to monitor the host controller or process controller JVMs in your environment to determine appropriate values for JVM options such as heap size.
CHAPTER 5. JAKARTA ENTERPRISE BEANS SUBSYSTEM TUNING

JBoss EAP can cache Jakarta Enterprise Beans to save initialization time. This is accomplished using bean pools.

There are two different bean pools that can be tuned in JBoss EAP: **bean instance pools** and **bean thread pools**.

Appropriate bean pool sizes depend on your environment and applications. It is recommended that you experiment with different bean pool sizes and perform stress testing in a development environment that emulates your expected real-world conditions.

5.1. BEAN INSTANCE POOLS

Bean instance pools are used for Stateless Session Beans (SLSBs) and Message Driven Beans (MDBs). By default, SLSBs use the instance pool `default-slsb-instance-pool`, and MDBs use the instance pool `default-mdb-instance-pool`.

The size of a bean instance pool limits the number of instances of a particular enterprise bean that can be created at one time. If the pool for a particular enterprise bean is full, the client will block and wait for an instance to become available. If a client does not get an instance within the time set in the pool’s **timeout** attributes, an exception is thrown.

The size of a bean instance pool is configured using either `derive-size` or `max-pool-size`. The `derive-size` attribute allows you to configure the pool size using one of the following values:

- `from-worker-pools`, which indicates that the maximum pool size is derived from the size of the total threads for all worker pools configured on the system.
- `from-cpu-count`, which indicates that the maximum pool size is derived from the total number of processors available on the system. Note that this is not necessarily a 1:1 mapping, and might be augmented by other factors.

If `derive-size` is undefined, then the value of `max-pool-size` is used for the size of the bean instance pool.

**NOTE**

The `derive-size` attribute overrides any value specified in `max-pool-size`. **derive-size** must be undefined for the `max-pool-size` value to take effect.

You can configure an enterprise bean to use a specific instance pool. This allows for finer control of the instances available to each enterprise bean type.

5.1.1. Creating a Bean Instance Pool

This section shows you how to create a new bean instance pool using the management CLI. You can also configure bean instance pools using the management console by navigating to the **Jakarta Enterprise Beans** subsystem from the **Configuration** tab, and then selecting the **Bean Pool** tab.

To create a new instance pool, use one of the following commands:

- To create a bean instance pool with a derived maximum pool size:
The following example creates a bean instance pool named `my_derived_pool` with a maximum size derived from the CPU count, and a timeout of 2 minutes:

```
/subsystem=ejb3/strict-max-bean-instance-pool=my_derived_pool:add(derive-size=from-cpu-count,timeout-unit=MINUTES,timeout=2)
```

- To create a bean instance pool with an explicit maximum pool size:

```
/subsystem=ejb3/strict-max-bean-instance-pool=POOL_NAME:add(max-pool-size=POOL_SIZE,timeout-unit=TIMEOUT_UNIT,timeout=TIMEOUT_VALUE)
```

The following example creates a bean instance pool named `my_pool` with a maximum of 30 instances and a timeout of 30 seconds:

```
/subsystem=ejb3/strict-max-bean-instance-pool=my_pool:add(max-pool-size=30,timeout-unit=SECONDS,timeout=30)
```

### 5.1.2. Specifying the Instance Pool a Bean Should Use

You can set a specific instance pool that a particular bean will use either by using the `@org.jboss.ejb3.annotation.Pool` annotation, or by modifying the `jboss-ejb3.xml` deployment descriptor of the bean. See the `jboss-ejb3.xml` Deployment Descriptor Reference in Developing Jakarta Enterprise Beans Applications for more information.

### 5.1.3. Disabling the Default Bean Instance Pool

The default bean instance pool can be disabled, which results in an enterprise bean not using any instance pool by default. Instead, a new enterprise bean instance is created when a thread needs to invoke a method on an enterprise bean. This might be useful if you do not want any limit on the number of enterprise bean instances that are created.

To disable the default bean instance pool, use the following management CLI command:

```
/subsystem=ejb3:undefine-attribute(name=default-slsb-instance-pool)
```

**NOTE**

If a bean is configured to use a particular bean instance pool, disabling the default instance pool does not affect the pool that the bean uses.

### 5.2. BEAN THREAD POOLS

By default, a bean thread pool named `default` is used for asynchronous enterprise bean calls and enterprise bean timers.

**NOTE**

From JBoss EAP 7 onward, remote enterprise bean requests are handled in the worker defined in the `io` subsystem by default.
If required, you can configure each of these enterprise bean services to use a different bean thread pool. This can be useful if you want finer control of each service’s access to a bean thread pool.

When determining an appropriate thread pool size, consider how many concurrent requests you expect will be processed at once.

5.2.1. Creating a Bean Thread Pool

This section shows you how to create a new bean thread pool using the management CLI. You can also configure bean thread pools using the management console by navigating to the Jakarta Enterprise Beans subsystem from the Configuration tab and selecting Container → Thread Pool in the left menu.

To create a new thread pool, use the following command:

```
/subsystem=ejb3/thread-pool=POOL_NAME:add(max-threads=MAX_THREADS)
```

The following example creates a bean thread pool named `my_thread_pool` with a maximum of 30 threads:

```
/subsystem=ejb3/thread-pool=my_thread_pool:add(max-threads=30)
```

5.2.2. Configuring Enterprise Bean Services to Use a Specific Bean Thread Pool

The enterprise bean asynchronous invocation service and timer service can each be configured to use a specific bean thread pool. By default, both these services use the default bean thread pool.

This section shows you how to configure the above enterprise bean services to use a specific bean thread pool using the management CLI. You can also configure these services using the management console by navigating to the Enterprise Bean subsystem from the Configuration tab, selecting the Services tab, and choosing the appropriate service.

To configure an enterprise bean service to use a specific bean thread pool, use the following command:

```
/subsystem=ejb3/service=SERVICE_NAME:write-attribute(name=thread-pool-name,value=THREAD_POOL_NAME)
```

Replace `SERVICE_NAME` with the an enterprise bean service you want to configure:

- `async` for the enterprise bean asynchronous invocation service
- `timer-service` for the enterprise bean timer service

The following example sets the enterprise bean async service to use the bean thread pool named `my_thread_pool`:

```
/subsystem=ejb3/service=async:write-attribute(name=thread-pool-name,value=my_thread_pool)
```

5.3. RUNTIME BEAN DEPLOYMENT INFORMATION

Runtime metadata is available from bean deployments so you can monitor the performance of your beans.

Runtime metadata is available for the following types of beans:
• stateful session beans
• stateless session beans
• singleton beans
• message-driven beans

The bean application includes the metadata as annotations in the code or in the deployment descriptor. An application can use both options. For details about the available runtime data, see the **ejb3** subsystem in the JBoss EAP management model.

**Additional resources**

• For more information about available runtime data, see the **ejb3** subsystem in the JBoss EAP management model.

### 5.3.1. Command line options for retrieving runtime data from Jakarta Enterprise Beans

Runtime data from Jakarta Enterprise Beans is available from the management CLI so you can evaluate the performance of your Jakarta Enterprise Beans.

The command to retrieve runtime data for all types of beans uses the following pattern:

```
$ /deployment=<deployment_name>/subsystem=ejb3/<bean_type>=<bean_name>:read-resource(include-runtime)
```

Replace `<deployment_name>` with the name of the deployment .jar file for which to retrieve runtime data. Replace `<bean_type>` with the type of the bean for which to retrieve runtime data. The following options are valid for this placeholder:

• **stateless-session-bean**
• **stateful-session-bean**
• **singleton-bean**
• **message-driven-bean**

Replace `<bean_name>` with the name of the bean for which you to retrieve runtime data.

The system delivers the result to **stdout** formatted as JavaScript Object Notation (JSON) data.

**Example command to retrieve runtime data for a singleton bean named ManagedSingletonBean deployed in a file named ejb-management.jar**

```
$ /deployment=ejb-management.jar/subsystem=ejb3/singleton-bean=ManagedSingletonBean:read-resource(include-runtime)
```

**Example output runtime data for the singleton bean**

```json
{
  "outcome" => "success",
  "result" => {
```
"async-methods" => ["void async(int, int)"],
"business-local" => ["sample.ManagedSingletonBean"],
"business-remote" => ["sample.BusinessInterface"],
"component-class-name" => "sample.ManagedSingletonBean",
"concurrency-management-type" => undefined,
"declared-roles" => [
  "Role3",
  "Role2",
  "Role1"
],
"depends-on" => undefined,
"execution-time" => 156L,
"init-on-startup" => false,
"invocations" => 3L,
"jndi-names" => [
  "java:module/ManagedSingletonBean!sample.ManagedSingletonBean",
  "java:global/ejb-management/ManagedSingletonBean!sample.ManagedSingletonBean",
  "java:app/ejb-management/ManagedSingletonBean!sample.ManagedSingletonBean",
  "java:app/ejb-management/ManagedSingletonBean!sample.BusinessInterface",
  "java:global/ejb-management/ManagedSingletonBean!sample.BusinessInterface",
  "java:module/ManagedSingletonBean!sample.BusinessInterface"
],
"methods" => {"doIt" => {
  "execution-time" => 156L,
  "invocations" => 3L,
  "wait-time" => 0L
}},
"peak-concurrent-invocations" => 1L,
"run-as-role" => "Role3",
"security-domain" => "other",
"timeout-method" => "public void sample.ManagedSingletonBean.timeout(javax.ejb.Timer)",
"timers" => [{
  "time-remaining" => 4304279L,
  "next-timeout" => 1577768415000L,
  "calendar-timer" => true,
  "persistent" => false,
  "info" => "timer1",
  "schedule" => {
    "year" => ".*",
    "month" => ".*",
    "day-of-month" => ".*",
    "day-of-week" => ".*",
    "hour" => "0",
    "minute" => "0",
    "second" => "15",
    "timezone" => undefined,
    "start" => undefined,
    "end" => undefined
  }
},
"transaction-type" => "CONTAINER",
"wait-time" => 0L,
"service" => {"timer-service" => undefined}
}
Example command to retrieve runtime data for a message-driven bean named **NoTimerMDB** deployed in a file named `ejb-management.jar`

```
/deployment=ejb-management.jar/subsystem=ejb3/message-driven-bean=NoTimerMDB:read-resource(include-runtime)
```

Example output for the message-driven bean

```
{
   "outcome" => "success",
   "result" => {
      "activation-config" => [
         ("destination" => "java:/queue/NoTimerMDB-queue"),
         ("destinationType" => "javax.jms.Queue"),
         ("acknowledgeMode" => "Auto-acknowledge")
      ],
      "component-class-name" => "sample.NoTimerMDB",
      "declared-roles" => [
         "Role3",
         "Role2",
         "Role1"
      ],
      "delivery-active" => true,
      "execution-time" => 0L,
      "invocations" => 0L,
      "message-destination-link" => "queue/NoTimerMDB-queue",
      "message-destination-type" => "javax.jms.Queue",
      "messaging-type" => "javax.jms.MessageListener",
      "methods" => {},
      "peak-concurrent-invocations" => 0L,
      "pool-available-count" => 16,
      "pool-create-count" => 0,
      "pool-current-size" => 0,
      "pool-max-size" => 16,
      "pool-name" => "mdb-strict-max-pool",
      "pool-remove-count" => 0,
      "run-as-role" => "Role3",
      "security-domain" => "other",
      "timeout-method" => undefined,
      "timers" => [],
      "transaction-type" => "CONTAINER",
      "wait-time" => 0L,
      "service" => undefined
   }
}
```

5.4. EXCEPTIONS THAT INDICATE AN ENTERPRISE BEAN SUBSYSTEM TUNING MIGHT BE REQUIRED

- The Stateless Jakarta Enterprise Beans instance pool is not large enough or the timeout is too low
javax.ejb.EJBException: JBAS014516: Failed to acquire a permit within 20 SECONDS at org.jboss.as.ejb3.pool.strictmax.StrictMaxPool.get(StrictMaxPool.java:109)

See Bean Instance Pools.

- The enterprise bean thread pool is not large enough, or an enterprise bean is taking longer to process than the invocation timeout

java.util.concurrent.TimeoutException: No invocation response received in 300000 milliseconds

See Bean Thread Pools.

5.5. DEFAULT GLOBAL TIMEOUT VALUES FOR SFSBS

In the ejb3 subsystem, you can configure a default global timeout value for all stateful session beans (SFSBs) that are deployed on your server instance by using the default-stateful-bean-session-timeout attribute.

With the default-stateful-bean-session-timeout attribute, you can use the following management CLI operations on the ejb3 subsystem:

- The read-attribute operation in the management CLI to view the current global timeout value for the attribute.
- The write-attribute operation to configure the attribute by using the management CLI.

Attribute behavior varies according to the server mode. For example:

- When running in the standalone server, the configured value gets applied to all SFSBs deployed on the application server.
- When running a server in a managed domain, all SFSBs that are deployed on server instances within server groups receive concurrent timeout values.

**NOTE**

When you change the global timeout value for the attribute, the updated settings only apply to new deployments. You must reload the server to apply the new settings to current deployments.

By default, the attribute value is set at -1, which means that deployed SFSBs are configured to never time out. However, you can configure two of the following types of valid values for the attribute:

- When you set the attribute value to 0, the attribute immediately marks eligible SFSBs for removal by the ejb container.
- When you set the attribute value greater than 0, the SFSBs remain idle for the specified time in milliseconds before the ejb container removes the eligible SFSBs.
NOTE

You can still use the pre-existing `@StatefulTimeout` annotation or the `stateful-timeout` element, which is located in the `ejb-jar.xml` deployment descriptor, to configure the timeout value for an SFSB. However, setting such a configuration overrides the default global timeout value to the SFSB.

Two methods exist for verifying a new value you set for the attribute:

- Use the `read-attribute` operation in the management CLI.
- Examine the `ejb3` subsystem section of the server’s configuration file.

Additional resources

- For more information about viewing the current global timeout value for an attribute, see Display an Attribute Value in the Management CLI Guide.
- For more information about updating the current global timeout value for an attribute, see Update an Attribute in the Management CLI Guide.
CHAPTER 6. DATASOURCE AND RESOURCE ADAPTER TUNING

Connection pools are the principal tool that JBoss EAP uses to optimize performance for environments that use datasources, such as relational databases, or resource adapters.

Allocating and deallocating resources for datasource and resource adapter connections is very expensive in terms of time and system resources. Connection pooling reduces the cost of connections by creating a 'pool' of connections that are available to applications.

Before configuring your connection pool for optimal performance, you must monitor the datasource pool statistics or resource adapter statistics under load to determine the appropriate settings for your environment.

6.1. MONITORING POOL STATISTICS

6.1.1. Datasource Statistics

When statistics collection is enabled for a datasource, you can view runtime statistics for the datasource.

6.1.1.1. Enabling Datasource Statistics

By default, datasource statistics are not enabled. You can enable datasource statistics collection using the management CLI or the management console.

Enable Datasource Statistics Using the Management CLI

The following management CLI command enables the collection of statistics for the ExampleDS datasource.

```
/subsystem=datasources/data-source=ExampleDS:write-attribute(name=statistics-enabled,value=true)
```

Reload the server for the changes to take effect.

Enable Datasource Statistics Using the Management Console

Use the following steps to enable statistics collection for a datasource using the management console.

1. Navigate to Configuration → Subsystems → Datasources & Drivers → Datasources.

2. Select the datasource and click View.

3. Click Edit under the Attributes tab.

4. Set the Statistics Enabled field to ON and click Save. A popup appears indicating that the changes require a reload in order to take effect.

5. Reload the server.

   • For a standalone server, click the Reload link from the popup to reload the server.
For a managed domain, click the Topology link from the popup. From the Topology tab, select the appropriate server and select the Reload drop down option to reload the server.

6.1.1.2. Viewing Datasource Statistics

You can view runtime statistics for a datasource using the management CLI or management console.

View Datasource Statistics Using the Management CLI

The following management CLI command retrieves the core pool statistics for the ExampleDS datasource.

```
{  
  "outcome" => "success",  
  "result" => {  
    "ActiveCount" => 1,  
    "AvailableCount" => 20,  
    "AverageBlockingTime" => 0L,  
    "AverageCreationTime" => 122L,  
    "AverageGetTime" => 128L,  
    "AveragePoolTime" => 0L,  
    "AverageUsageTime" => 0L,  
    "BlockingFailureCount" => 0,  
    "CreatedCount" => 1,  
    "DestroyedCount" => 0,  
    "IdleCount" => 1,  
    "..."
  }
}
```

The following management CLI command retrieves the JDBC statistics for the ExampleDS datasource.

```
{  
  "outcome" => "success",  
  "result" => {  
    "PreparedStatementCacheAccessCount" => 0L,  
    "PreparedStatementCacheAddCount" => 0L,  
    "PreparedStatementCacheCurrentSize" => 0,  
    "PreparedStatementCacheDeleteCount" => 0L,  
    "PreparedStatementCacheHitCount" => 0L,  
    "PreparedStatementCacheMissCount" => 0L,  
    "statistics-enabled" => true
  }
}
```
NOTE
Since statistics are runtime information, be sure to specify the include-runtime=true argument.

See Datasource Statistics for a detailed list of all available statistics.

View Datasource Statistics Using the Management Console
To view datasource statistics from the management console, navigate to the Datasources subsystem from the Runtime tab, select a datasource, and click View.

See Datasource Statistics for a detailed list of all available statistics.

6.1.2. Resource Adapter Statistics
You can view core runtime statistics for deployed resource adapters. See the Resource Adapter Statistics appendix for a detailed list of all available statistics.

Enable Resource Adapter Statistics
By default, resource adapter statistics are not enabled. The following management CLI command enables the collection of statistics for a simple resource adapter myRA.rar with a connection factory bound in JNDI as java:/eis/AcmeConnectionFactory:

```
NOTE
In a managed domain, precede the command with /host=HOST_NAME/server=SERVER_NAME/.
```

```
/deployment=myRA.rar/subsystem=resource-adapters/statistics=statistics/connection-
definitions=java:/eis/AcmeConnectionFactory/write-attribute(name=statistics-enabled,value=true)
```

View Resource Adapter Statistics
Resource adapter statistics can be retrieved from the management CLI. The following management CLI command returns statistics for the resource adapter myRA.rar with a connection factory bound in JNDI as java:/eis/AcmeConnectionFactory.

```
NOTE
In a managed domain, precede the command with /host=HOST_NAME/server=SERVER_NAME/.
```

```
deployment=myRA.rar/subsystem=resource-adapters/statistics=statistics/connection-
definitions=java:/eis/AcmeConnectionFactory/read-resource(include-runtime=true)
{
    "outcome" => "success",
    "result" => {
        "ActiveCount" => "1",
        "AvailableCount" => "20",
        "AverageBlockingTime" => "0",
        "AverageCreationTime" => "0",
        "CreatedCount" => "1",
        "DestroyedCount" => "0",
        "InUseCount" => "0",
```

Red Hat JBoss Enterprise Application Platform 7.4 Performance Tuning Guide 34
6.2. POOL ATTRIBUTES

This section details advice for selected pool attributes that can be configured for optimal datasource or resource adapter performance. For instructions on how to configure each of these attributes, see:

- Configuring Datasource Pool Attributes
- Configuring Resource Adapter Pool Attributes

**Minimum Pool Size**

The `min-pool-size` attribute defines the minimum size of the connection pool. The default minimum is zero connections. With a zero `min-pool-size`, connections are created and placed in the pool when the first transactions occur.

If `min-pool-size` is too small, it results in increased latency while executing initial database commands because new connections might need to be established. If `min-pool-size` is too large, it results in wasted connections to the datasource or resource adapter.

During periods of inactivity the connection pool will shrink, possibly to the `min-pool-size` value.

Red Hat recommends that you set `min-pool-size` to the number of connections that allow for ideal on-demand throughput for your applications.

**Maximum Pool Size**

The `max-pool-size` attribute defines the maximum size of the connection pool. It is an important performance parameter because it limits the number of active connections, and thus also limits the amount of concurrent activity.

If `max-pool-size` is too small, it can result in requests being unnecessarily blocked. If `max-pool-size` is too large, it can result in your JBoss EAP environment, datasource, or resource adapter using more resources than it can handle.

Red Hat recommends that you set the `max-pool-size` to at least 15% higher than an acceptable `MaxUsedCount` observed after monitoring performance under load. This allows some buffer for higher than expected conditions.

**Prefill**

The `pool-prefill` attribute specifies whether JBoss EAP will prefill the connection pool with the minimum number of connections when JBoss EAP starts. The default value is `false`. 

---

**NOTE**

Since statistics are runtime information, be sure to specify the `include-runtime=true` argument.
When `pool-prefill` is set to `true`, JBoss EAP uses more resources at startup, but there will be less latency for initial transactions.

Red Hat recommends to set `pool-prefill` to `true` if you have optimized the `min-pool-size`.

**Strict Minimum**

The `pool-use-strict-min` attribute specifies whether JBoss EAP allows the number of connections in the pool to fall below the specified minimum.

If `pool-use-strict-min` is set to `true`, JBoss EAP will not allow the number of connections to temporarily fall below the specified minimum. The default value is `false`.

Although a minimum number of pool connections is specified, when JBoss EAP closes connections, for instance, if the connection is idle and has reached the timeout, the closure may cause the total number of connections to temporarily fall below the minimum before a new connection is created and added to the pool.

**Timeouts**

There are a number of timeout options that are configurable for a connection pool, but a significant one for performance tuning is `idle-timeout-minutes`.

The `idle-timeout-minutes` attribute specifies the maximum time, in minutes, a connection may be idle before being closed. As idle connections are closed, the number of connections in the pool will shrink down to the specified minimum.

The longer the timeout, the more resources are used but requests might be served faster. The lower the timeout, the less resources are used but requests might need to wait for a new connection to be created.

### 6.3. CONFIGURING POOL ATTRIBUTES

#### 6.3.1. Configuring Datasource Pool Attributes

**Prerequisites**

- Install a JDBC driver. See [JDBC Drivers](#) in the JBoss EAP *Configuration Guide*.
- Create a datasource. See [Creating Datasources](#) in the JBoss EAP *Configuration Guide*.

You can configure datasource pool attributes using either the management CLI or the management console:

- To use the management console, navigate to `Configuration → Subsystems → Datasources & Drivers → Datasources`, select your datasource, and click `View`. The pool options are configurable under the datasource `Pool` tab. Timeout options are configurable under the datasource `Timeouts` tab.

- To use the management CLI, execute the following command:

  ```bash
  /subsystem=datasources/data-source=DATASOURCE_NAME/:write-attribute(name=ATTRIBUTE_NAME,value=ATTRIBUTE_VALUE)
  ```

  For example, to set the `ExampleDS` datasource `min-pool-size` attribute to a value of 5 connections, use the following command:
6.3.2. Configuring Resource Adapter Pool Attributes

Prerequisites

- Deploy your resource adapter and add a connection definition. See Configuring Resource Adapters in the JBoss EAP Configuration Guide.

You can configure resource adapter pool attributes using either the management CLI or the management console:

- To use the management console, navigate to Configuration → Subsystems → Resource Adapters, select your resource adapter, click View, and select Connection Definitions in the left menu. The pool options are configurable under the Pool tab. Timeout options are configurable under the Attributes tab.

- To use the management CLI, execute the following command:

```
/subsystem=resource-adapters/resource-adapter=RESOURCE_ADAPTER_NAME/connection-definitions=CONNECTION_DEFINITION_NAME:write-attribute(name=ATTRIBUTE_NAME,value=ATTRIBUTE_VALUE)
```

For example, to set the my_RA resource adapter my_CD connection definition min-pool-size attribute to a value of 5 connections, use the following command:

```
/subsystem=resource-adapters/resource-adapter=my_RA/connection-definitions=my_CD:write-attribute(name=min-pool-size,value=5)
```
CHAPTER 7. MESSAGING SUBSYSTEM TUNING

Performance tuning advice for the messaging-activemq subsystem is covered in the Performance Tuning part of the JBoss EAP Configuring Messaging guide.
CHAPTER 8. LOGGING SUBSYSTEM TUNING

You can further improve upon JBoss EAP logging subsystem performance in production environments by disabling logging to the console, configuring appropriate logging levels, and specifying the best location to store log files.

Additional resources

- For more information on configuring the logging subsystem, see the logging chapter in the JBoss EAP Configuration Guide.

8.1. DISABLING LOGGING TO THE CONSOLE

Disabling console logging can improve JBoss EAP performance. Although outputting logs to the console can be useful in development and testing environments, for production environments, in most cases, it is not necessary.

The JBoss EAP root logger includes a console log handler for all default standalone server profiles except standalone-full-ha. The default managed domain profiles do not include a console handler.

Procedure

- To remove the default console handler from the root logger, use the following management CLI command:

  /subsystem=logging/root-logger=ROOT:remove-handler(name=CONSOLE)

8.2. CONFIGURING LOGGING LEVELS

For ideal performance, ensure that you configure the logging levels for your production environment appropriately. For example, although INFO or DEBUG levels might be appropriate for development or testing environments, in most cases you should set your production environment logging level to something higher, such as WARN or ERROR.

Additional resources

- For details on setting log levels for different logging handlers, see Configuring log handlers in the JBoss EAP Configuration Guide.

8.3. CONFIGURING THE LOCATION OF THE LOG FILES

You should consider the storage location of log files as a potential performance issue. If you save logs to a file system or disk configuration that has poor I/O throughput, it has the potential to affect the whole platform’s performance.

To prevent logging from impacting JBoss EAP performance, set your log locations to high-performance dedicated disks that have a lot of space.

Additional resources

- For details on configuring log file locations for different logging handlers, see Configuring log handlers in the JBoss EAP Configuration Guide.
CHAPTER 9. UNDERTOW SUBSYSTEM TUNING

The non-blocking I/O (NIO) undertow subsystem introduced in JBoss EAP 7 has greatly improved performance compared to the previous web subsystem in JBoss EAP 6. Opportunities for tuning the undertow subsystem for your environment include:

- Buffer cache configuration
- Byte buffer pools configuration
- Jakarta Server Pages configuration options
- Listener configuration options

9.1. BUFFER CACHE CONFIGURATION

A buffer cache stores static files handled by the undertow subsystem. This includes images, static HTML, CSS, and JavaScript files. You can specify a default buffer cache for each Undertow servlet container. Having an optimized buffer cache for your servlet container can improve Undertow performance for serving static files.

Buffers in a buffer cache are allocated in regions and are of a fixed size. There are three configurable attributes for each buffer cache:

- **buffer-size**: The size of an individual buffer, in bytes. The default is 1024 bytes. Set the buffer size to entirely store your largest static file.
- **buffers-per-region**: The number of buffers per region. The default is 1024.
- **max-regions**: The maximum number of regions, which sets a maximum amount of memory allocated to the buffer cache. The default is 10 regions.

You can calculate the maximum amount memory used by a buffer cache by multiplying the buffer size, the number of buffers per region, and the maximum number of regions. For example, the default buffer cache is 1024 bytes * 1024 buffers per region * 10 regions = 10 MB.

Configure your buffer caches based on the size of your static files, and the results from testing expected loads in a development environment. When determining the effect on performance, consider the balance of the buffer cache performance benefit versus the memory used.

Additional resources

- For instructions on using the management CLI to configure buffer caches, see Configuring buffer caches in the JBoss EAP Configuration Guide.

9.2. BYTE BUFFER POOLS CONFIGURATION

Undertow byte buffer pools are used to allocate pooled NIO ByteBuffer instances. All listeners have a byte buffer pool and you can use different buffer pools and workers for each listener. Byte buffer pools can be shared between different server instances.
The main byte buffer pool attribute that significantly affects performance is \texttt{buffer-size}. The default is calculated based on the RAM resources of your system and is sufficient in most cases. If you are configuring this attribute manually, an ideal size for most servers is 16 KB.

Additional resources

- For a full list of the attributes available for configuring byte buffer pools, see \textit{Byte Buffer Pool Attributes} in the JBoss EAP \textit{Configuration Guide}.
- See the JBoss EAP \textit{Configuration Guide} for instructions on how to create and configure byte buffer pools.

9.3. JAKARTA SERVER PAGES CONFIGURATION

The following Jakarta Server Pages configuration options for Undertow servlet containers provide optimizations for how Jakarta Server Pages are compiled into Java bytecode:

- \texttt{generate-strings-as-char-arrays}
  - If your Jakarta Server Pages contain a lot of \texttt{String} constants, enabling this option optimizes scriptlets by converting the \texttt{String} constants to \texttt{char} arrays.

- \texttt{optimize-scriptlets}
  - If your Jakarta Server Pages contain many \texttt{String} concatenations, enabling this option optimizes scriptlets by removing \texttt{String} concatenation for every Jakarta Server Pages request.

- \texttt{trim-spaces}
  - If your Jakarta Server Pages contain a lot of white space, enabling this option trims the white space from HTTP requests and reduces HTTP request payload.

9.3.1. Enabling Jakarta Server Pages options using the management console

To enable the Undertow Jakarta Server Pages configuration options using the management console, complete the following steps:

Procedure

1. Navigate to Configuration → Subsystems → Web (Undertow) → Servlet Container.
2. Select the servlet container you want to configure and click View.
3. Select Jakarta Server Pages and click Edit.
4. For each option you want to enable, set the field to ON, and then click Save.

9.3.2. Enabling Jakarta Server Pages options using the management CLI

To enable the Undertow Jakarta Server Pages configuration options using the management CLI, complete the following steps:

Procedure

- Use the following command:

  \begin{verbatim}
  /subsystem=undertow/servlet-container=SERVLET CONTAINER/setting=jsp/:write-attribute(name=OPTION_NAME,value=true)
  \end{verbatim}
For example, to enable `generate-strings-as-char-arrays` for the `default` servlet container, use the following command:

```
/subsystem=undertow/servlet-container=default/setting=jsp:/write-attribute(name=generate-strings-as-char-arrays,value=true)
```

## 9.4. LISTENER CONFIGURATION OPTIONS

Depending on your applications and environment, you can configure multiple listeners specific to certain types of traffic, for example, traffic on specific ports, and then configure options for each listener.

The following are selected performance-related options that you can configure on HTTP, HTTPS, and AJP listeners:

- **max-connections**
  The maximum number of concurrent connections that the listener can handle. By default this attribute is undefined, which results in unlimited connections. You can use this option to set a ceiling on the number of connections a listener can handle, which might be useful to cap resource usage. In configuring this value, you should consider your workload and traffic type. Also see `no-request-timeout` below.

- **no-request-timeout**
  The length of time in milliseconds that a connection is idle before it is closed. The default value is 60,000 milliseconds (1 minute). Tuning this option in your environment for optimal connection efficiency can help improve network performance. If idle connections are prematurely closed, there are overheads in re-establishing connections. If idle connections are open for too long, they unnecessarily use resources.

- **max-header-size**
  The maximum size of an HTTP request header, in bytes. The default is 1,048,576 (1024 KB). Limiting the header size can be useful to prevent certain types of denial of service attacks.

- **buffer-pool**
  Specifies the buffer pool in the `io` subsystem to use for the listener. By default, all listeners use the `default` buffer pool. You can use this option to configure each listener to use a unique buffer pool, or have multiple listeners use the same buffer pool.

- **worker**
  The `undertow` subsystem relies on the `io` subsystem to provide XNIO workers. This option specifies the XNIO worker that the listener uses. By default, a listener uses the `default` worker in the `io` subsystem. It might be useful to configure each listener to use a specific worker so you can assign different worker resources to certain types of network traffic.

### 9.4.1. Configuring listener options using the management console

To configure the listener options using the management console, complete the following steps:

**Procedure**

1. Navigate to **Configuration → Subsystems → Web (Undertow) → Server**.
2. Select the server you want to configure and click **View**.
3. In the left menu, select **Listener** then select the type of listener to configure, for example **HTTP Listener**, and select the listener in the table.
4. Click **Edit**, modify the options you want to configure, and click **Save**.

### 9.4.2. Configuring listener options using the management CLI

To configure the listener options using the management CLI, complete the following steps:

**Procedure**

- Use the following command:

  ```
  /subsystem=undertow/server=SERVER_NAME/LISTENER_TYPE=LISTENER_NAME:write-attribute(name=OPTION_NAME,value=OPTION_VALUE)
  ```

  For example, to set **max-connections** to **100000** for the **default** HTTP listener in the **default-server** Undertow server, use the following command:

  ```
  /subsystem=undertow/server=default-server/http-listener=default:write-attribute(name=max-connections,value=100000)
  ```
CHAPTER 10. IO SUBSYSTEM TUNING

The io subsystem defines XNIO workers and buffer pools that are used by other JBoss EAP subsystems, such as Undertow and Remoting.

10.1. CONFIGURING WORKERS

You can create multiple separate workers that each have their own performance configuration and which handle different I/O tasks. For example, you could create one worker to handle HTTP I/O, and another worker to handle Jakarta Enterprise Beans I/O, and then separately configure the attributes of each worker for specific load requirements.

See the IO Subsystem Attributes appendix for the list of configurable worker attributes.

Worker attributes that significantly affect performance include io-threads which sets the total number of I/O threads that a worker can use, and task-max-threads which sets the maximum number of threads that can be used for a particular task. The defaults for these two attributes are calculated based on the server's CPU count.

See the JBoss EAP Configuration Guide for instructions on how to create and configure workers.

10.1.1. Monitoring Worker Statistics

You can view a worker’s runtime statistics using the management CLI. This exposes worker statistics such as connection count, thread count, and queue size.

The following command displays runtime statistics for the default worker:

```
/subsystem=io/worker=default:read-resource(include-runtime=true,recursive=true)
```

NOTE

The number of core threads, which is tracked by the core-pool-size statistic, is currently always set to the same value as the maximum number of threads, which is tracked by the max-pool-size statistic.

10.2. CONFIGURING BUFFER POOLS

NOTE

IO buffer pools are deprecated, but they are still set as the default in the current release. For more information about configuring Undertow byte buffer pools, see the Configuring Byte Buffer Pools section of the Configuration Guide for JBoss EAP.

A buffer pool in the io subsystem is a pooled NIO buffer instance that is used specifically for I/O operations. Like workers, you can create separate buffer pools which can be dedicated to handle specific I/O tasks.

See the IO Subsystem Attributes appendix for the list of configurable buffer pool attributes.

The main buffer pool attribute that significantly affects performance is buffer-size. The default is calculated based on the RAM resources of your system, and is sufficient in most cases. If you are configuring this attribute manually, an ideal size for most servers is 16KB.
See the JBoss EAP Configuration Guide for instructions on how to create and configure buffer pools.
CHAPTER 11. JGROUPS SUBSYSTEM TUNING

For optimal network performance it is recommended that you use UDP multicast for JGroups in environments that support it.

NOTE

TCP has more overhead and is often considered slower than UDP since it handles error checking, packet ordering, and congestion control itself. JGroups handles these features for UDP, whereas TCP guarantees them itself. TCP is a good choice when using JGroups on unreliable or high congestion networks, or when multicast is not available.

This chapter assumes that you have chosen your JGroups stack transport protocol (UDP or TCP) and communications protocols that JGroups cluster communications will use. See the JBoss EAP Configuration Guide for more information about cluster communication with JGroups.

11.1. MONITORING JGROUPS STATISTICS

You can enable statistics for the jgroups subsystem to monitor JBoss EAP clustering using the management CLI or through JMX.

NOTE

Enabling statistics adversely affects performance. Only enable statistics when necessary.

1. Use the following command to enable statistics for a JGroups channel.

   NOTE

   In a managed domain, precede these commands with /profile=PROFILE_NAME.

   /subsystem=jgroups/channel=CHANNEL_NAME:write-attribute(name=statistics-enabled,value=true)

   For example, use the following command to enable statistics for the default ee channel.

   /subsystem=jgroups/channel=ee:write-attribute(name=statistics-enabled,value=true)

2. Reload the JBoss EAP server.

   reload

3. You can now see JGroups statistics using either the management CLI, or through JMX with a JVM monitoring tool:

   • To use the management CLI, use the :read-resource(include-runtime=true) command on the JGroups channel or protocol that you want to see the statistics for.
NOTE

In a managed domain, precede these commands with `/host=HOST_NAME/server=SERVER_NAME`.

For example:

- To see the statistics for the ee channel, use the following command:
  `/subsystem=jgroups/channel=ee:read-resource(include-runtime=true)`

- To see the statistics for the FD_ALL protocol in the ee channel, use the following command:
  `/subsystem=jgroups/channel=ee/protocol=FD_ALL:read-resource(include-runtime=true)`

- To connect to JBoss EAP using a JVM monitoring tool, see the Monitoring Performance chapter. You can see the statistics on JGroups MBeans through the JMX connection.

11.2. NETWORKING AND JUMBO FRAMES

Where possible, it is recommended that the network interface for JGroups traffic should be part of a dedicated Virtual Local Area Network (VLAN). This allows you to separate cluster communications from other JBoss EAP network traffic to more easily control cluster network performance, throughput, and security.

Another network configuration to consider to improve cluster performance is to enable jumbo frames. If your network environment supports it, enabling jumbo frames by increasing the Maximum Transmission Unit (MTU) can help boost network performance, especially in high throughput environments.

To use jumbo frames, all NICs and switches in your network must support it. See the Red Hat Customer Portal for instructions on enabling jumbo frames for Red Hat Enterprise Linux.

11.3. MESSAGE BUNDLING

Message bundling in JGroups improves network performance by assembling multiple small messages into larger bundles. Rather than sending out many small messages over the network to cluster nodes, instead messages are queued until the maximum bundle size is reached or there are no more messages to send. The queued messages are assembled into a larger message bundle and then sent.

This bundling reduces communications overhead, especially in TCP environments where there is a higher overhead for network communications.

Configuring Message Bundling

JGroups message bundling is configured using the `max_bundle_size` property. The default `max_bundle_size` is 64KB.

The performance improvements of tuning the bundle size depend on your environment, and whether more efficient network traffic is balanced against a possible delay of communications while the bundle is assembled.

Use the following management CLI command to configure `max_bundle_size`.

*
For example, to set `max_bundle_size` to 60K for the default `udp` stack:

```
/subsystem=jgroups/stack=udp/transport=UDP/property=max_bundle_size:add(value=60K)
```

### 11.4. JGROUPS THREAD POOLS

The `jgroups` subsystem uses its own thread pools for processing cluster communication. JGroups contains thread pools for `default`, `internal`, `oob`, and `timer` functions which you can configure individually. Each JGroups thread pool includes configurable attributes for `keepalive-time`, `max-threads`, `min-threads`, and `queue-length`.

Appropriate values for each thread pool attribute depend on your environment, but for most situations the default values should suffice.

See the JBoss EAP Configuration Guide for instructions on how to configure JGroups thread pools.

### 11.5. JGROUPS SEND AND RECEIVE BUFFERS

The `jgroups` subsystem has configurable send and receive buffers for both UDP and TCP stacks.

Appropriate values for JGroups buffers depend on your environment, but for most situations the default values should suffice. It is recommended that you test your cluster under load in a development environment to tune appropriate values for the buffer sizes.

**NOTE**

Your operating system may limit the available buffer sizes and JBoss EAP may not be able to use its configured buffer values. See Resolving Buffer Size Warnings in the JBoss EAP Configuration Guide.

See the JBoss EAP Configuration Guide for instructions on how to configure JGroups send and receive buffers.
CHAPTER 12. TRANSACTIONS SUBSYSTEM TUNING

If your environment uses XA distributed transactions, you can tune the transaction manager’s log store for better performance.

The default transaction log store uses a simple file store. For XA transactions this type of log store can be inefficient, as it creates one system file for each transaction log. Especially for XA transactions, a journal store is much more efficient as it uses a journal that consists of one file for all transactions.

For better XA transaction performance, it is recommended that you use a journal log store. For Red Hat Enterprise Linux systems, you can additionally enable asynchronous I/O (AIO) on the journal store to further improve performance.

**NOTE**

For Red Hat Enterprise Linux systems, if you are enabling asynchronous I/O (AIO) on the journal store, ensure that the `libaio` package is installed.

Enable the Journal Log Store Using the Management Console

1. Navigate to Configuration → Subsystems → Transaction → and click View.
2. In the Configuration tab, click Edit.
3. Set the Use Journal Store field to ON.
4. Optional: For Red Hat Enterprise Linux systems, set the Journal Store Enable Async IO field to ON.
5. Click Save.

Enable the Journal Log Store Using the Management CLI

1. To enable the journal log store using the management CLI, use the following command:

   ```
   /subsystem=transactions:write-attribute(name=use-journal-store,value=true)
   ```

2. Optional: For Red Hat Enterprise Linux systems, use the following command to enable journal log store asynchronous I/O:

   ```
   /subsystem=transactions:write-attribute(name=journal-store-enable-async-io, value=true)
   ```
## APPENDIX A. REFERENCE MATERIAL

### A.1. DATASOURCE STATISTICS

**Table A.1. Core Pool Statistics**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ActiveCount</strong></td>
<td>The number of active connections. Each of the connections is either in use by an application or available in the pool.</td>
</tr>
<tr>
<td><strong>AvailableCount</strong></td>
<td>The number of available connections in the pool.</td>
</tr>
<tr>
<td><strong>AverageBlockingTime</strong></td>
<td>The average time spent blocking on obtaining an exclusive lock on the pool. This value is in milliseconds.</td>
</tr>
<tr>
<td><strong>AverageCreationTime</strong></td>
<td>The average time spent creating a connection. This value is in milliseconds.</td>
</tr>
<tr>
<td><strong>AverageGetTime</strong></td>
<td>The average time spent obtaining a connection. This value is in milliseconds.</td>
</tr>
<tr>
<td><strong>AveragePoolTime</strong></td>
<td>The average time that a connection spent in the pool. This value is in milliseconds.</td>
</tr>
<tr>
<td><strong>AverageUsageTime</strong></td>
<td>The average time spent using a connection. This value is in milliseconds.</td>
</tr>
<tr>
<td><strong>BlockingFailureCount</strong></td>
<td>The number of failures trying to obtain a connection.</td>
</tr>
<tr>
<td><strong>CreatedCount</strong></td>
<td>The number of connections created.</td>
</tr>
<tr>
<td><strong>DestroyedCount</strong></td>
<td>The number of connections destroyed.</td>
</tr>
<tr>
<td><strong>IdleCount</strong></td>
<td>The number of connections that are currently idle.</td>
</tr>
<tr>
<td><strong>InUseCount</strong></td>
<td>The number of connections currently in use.</td>
</tr>
<tr>
<td><strong>MaxCreationTime</strong></td>
<td>The maximum time it took to create a connection. This value is in milliseconds.</td>
</tr>
<tr>
<td><strong>MaxGetTime</strong></td>
<td>The maximum time for obtaining a connection. This value is in milliseconds.</td>
</tr>
<tr>
<td><strong>MaxPoolTime</strong></td>
<td>The maximum time for a connection in the pool. This value is in milliseconds.</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>MaxUsageTime</td>
<td>The maximum time using a connection. This value is in milliseconds.</td>
</tr>
<tr>
<td>MaxUsedCount</td>
<td>The maximum number of connections used.</td>
</tr>
<tr>
<td>MaxWaitCount</td>
<td>The maximum number of requests waiting for a connection at the same time.</td>
</tr>
<tr>
<td>MaxWaitTime</td>
<td>The maximum time spent waiting for an exclusive lock on the pool. This value is in milliseconds.</td>
</tr>
<tr>
<td>TimedOut</td>
<td>The number of timed out connections.</td>
</tr>
<tr>
<td>TotalBlockingTime</td>
<td>The total time spent waiting for an exclusive lock on the pool. This value is in milliseconds.</td>
</tr>
<tr>
<td>TotalCreationTime</td>
<td>The total time spent creating connections. This value is in milliseconds.</td>
</tr>
<tr>
<td>TotalGetTime</td>
<td>The total time spent obtaining connections. This value is in milliseconds.</td>
</tr>
<tr>
<td>TotalPoolTime</td>
<td>The total time spent by connections in the pool. This value is in milliseconds.</td>
</tr>
<tr>
<td>TotalUsageTime</td>
<td>The total time spent using connections. This value is in milliseconds.</td>
</tr>
<tr>
<td>WaitCount</td>
<td>The number of requests that had to wait to obtain a connection.</td>
</tr>
<tr>
<td>XACommitsAverageTime</td>
<td>The average time for an XAResource commit invocation. This value is in milliseconds.</td>
</tr>
<tr>
<td>XACommitsCount</td>
<td>The number of XAResource commit invocations.</td>
</tr>
<tr>
<td>XACommitsMaxTime</td>
<td>The maximum time for an XAResource commit invocation. This value is in milliseconds.</td>
</tr>
<tr>
<td>XACommitsTotalTime</td>
<td>The total time for all XAResource commit invocations. This value is in milliseconds.</td>
</tr>
<tr>
<td>XAEndsAverageTime</td>
<td>The average time for an XAResource end invocation. This value is in milliseconds.</td>
</tr>
<tr>
<td>XAEndsCount</td>
<td>The number of XAResource end invocations.</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>XAEndMaxTime</td>
<td>The maximum time for an XAResource end invocation. This value is in milliseconds.</td>
</tr>
<tr>
<td>XAEndTotalTime</td>
<td>The total time for all XAResource end invocations. This value is in milliseconds.</td>
</tr>
<tr>
<td>XAForgetAverageTime</td>
<td>The average time for an XAResource forget invocation. This value is in milliseconds.</td>
</tr>
<tr>
<td>XAForgetCount</td>
<td>The number of XAResource forget invocations.</td>
</tr>
<tr>
<td>XAForgetMaxTime</td>
<td>The maximum time for an XAResource forget invocation. This value is in milliseconds.</td>
</tr>
<tr>
<td>XAForgetTotalTime</td>
<td>The total time for all XAResource forget invocations. This value is in milliseconds.</td>
</tr>
<tr>
<td>XAPrepareAverageTime</td>
<td>The average time for an XAResource prepare invocation. This value is in milliseconds.</td>
</tr>
<tr>
<td>XAPrepareCount</td>
<td>The number of XAResource prepare invocations.</td>
</tr>
<tr>
<td>XAPrepareMaxTime</td>
<td>The maximum time for an XAResource prepare invocation. This value is in milliseconds.</td>
</tr>
<tr>
<td>XAPrepareTotalTime</td>
<td>The total time for all XAResource prepare invocations. This value is in milliseconds.</td>
</tr>
<tr>
<td>XARollbackAverageTime</td>
<td>The average time for an XAResource rollback invocation. This value is in milliseconds.</td>
</tr>
<tr>
<td>XARollbackCount</td>
<td>The number of XAResource rollback invocations.</td>
</tr>
<tr>
<td>XARollbackMaxTime</td>
<td>The maximum time for an XAResource rollback invocation. This value is in milliseconds.</td>
</tr>
</tbody>
</table>
### Table A.2. JDBC Statistics

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PreparedStatementCacheAccessCount</td>
<td>The number of times that the statement cache was accessed.</td>
</tr>
<tr>
<td>PreparedStatementCacheAddCount</td>
<td>The number of statements added to the statement cache.</td>
</tr>
<tr>
<td>PreparedStatementCacheCurrentSize</td>
<td>The number of prepared and callable statements currently cached in the statement cache.</td>
</tr>
<tr>
<td>PreparedStatementCacheDeleteCount</td>
<td>The number of statements discarded from the cache.</td>
</tr>
<tr>
<td>PreparedStatementCacheHitCount</td>
<td>The number of times that statements from the cache were used.</td>
</tr>
<tr>
<td>PreparedStatementCacheMissCount</td>
<td>The number of times that a statement request could not be satisfied with a statement from the cache.</td>
</tr>
</tbody>
</table>

### A.2. RESOURCE ADAPTER STATISTICS

#### Table A.3. Resource Adapter Statistics

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ActiveCount</td>
<td>The number of active connections. Each of the connections is either in use by an application or available in the pool</td>
</tr>
<tr>
<td>AvailableCount</td>
<td>The number of available connections in the pool.</td>
</tr>
<tr>
<td>AverageBlockingTime</td>
<td>The average time spent blocking on obtaining an exclusive lock on the pool. The value is in milliseconds.</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>AverageCreationTime</td>
<td>The average time spent creating a connection. The value is in milliseconds.</td>
</tr>
<tr>
<td>CreatedCount</td>
<td>The number of connections created.</td>
</tr>
<tr>
<td>DestroyedCount</td>
<td>The number of connections destroyed.</td>
</tr>
<tr>
<td>InUseCount</td>
<td>The number of connections currently in use.</td>
</tr>
<tr>
<td>MaxCreationTime</td>
<td>The maximum time it took to create a connection. The value is in milliseconds.</td>
</tr>
<tr>
<td>MaxUsedCount</td>
<td>The maximum number of connections used.</td>
</tr>
<tr>
<td>MaxWaitCount</td>
<td>The maximum number of requests waiting for a connection at the same time.</td>
</tr>
<tr>
<td>MaxWaitTime</td>
<td>The maximum time spent waiting for an exclusive lock on the pool.</td>
</tr>
<tr>
<td>TimedOut</td>
<td>The number of timed out connections.</td>
</tr>
<tr>
<td>TotalBlockingTime</td>
<td>The total time spent waiting for an exclusive lock on the pool. The value is in milliseconds.</td>
</tr>
<tr>
<td>TotalCreationTime</td>
<td>The total time spent creating connections. The value is in milliseconds.</td>
</tr>
<tr>
<td>WaitCount</td>
<td>The number of requests that had to wait for a connection.</td>
</tr>
</tbody>
</table>

**A.3. IO SUBSYSTEM ATTRIBUTES**

**NOTE**

Attribute names in these tables are listed as they appear in the management model, for example, when using the management CLI. See the schema definition file located at `EAP_HOME/docs/schema/wildfly-io_2_0.xsd` to view the elements as they appear in the XML, as there may be differences from the management model.

**Table A.4. worker Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>io-threads</td>
<td></td>
<td>The number of I/O threads to create for the worker. If not specified, the number of threads is set to the number of CPUs × 2.</td>
</tr>
<tr>
<td>stack-size</td>
<td>0</td>
<td>The stack size, in bytes, to attempt to use for worker threads.</td>
</tr>
</tbody>
</table>
### Table A.5. buffer-pool Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>buffer-size</td>
<td></td>
<td>The size, in bytes, of each buffer slice. If not specified, the size is set based on the available RAM of your system:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 512 bytes for less than 64 MB RAM</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 1024 bytes (1 KB) for 64 MB - 128 MB RAM</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 16384 bytes (16 KB) for more than 128 MB RAM</td>
</tr>
<tr>
<td></td>
<td></td>
<td>For performance tuning advice on this attribute, see Configuring Buffer Pools.</td>
</tr>
<tr>
<td>buffers-per-slice</td>
<td></td>
<td>How many slices, or sections, to divide the larger buffer into. This can be more memory efficient than allocating many separate buffers. If not specified, the number of slices is set based on the available RAM of your system:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 10 for less than 128 MB RAM</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 20 for more than 128 MB RAM</td>
</tr>
</tbody>
</table>

---

**NOTE**

IO buffer pools are deprecated, but they are still set as the default in the current release. For more information about configuring Undertow byte buffer pools, see the Configuring Byte Buffer Pools section of the Configuration Guide for JBoss EAP. Additionally, see Byte Buffer Pool Attributes in the JBoss EAP Configuration Guide for the byte buffer pool attribute list.
Whether the buffer pool uses direct buffers, which are faster in many cases with NIO. Note that some platforms do not support direct buffers.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>direct-buffers</td>
<td></td>
<td>Whether the buffer pool uses direct buffers, which are faster in many cases with NIO. Note that some platforms do not support direct buffers.</td>
</tr>
</tbody>
</table>

*Revised on 2021-12-06 16:06:52 UTC*