
Unit 5: Configuring for Scalability

Unit Objectives

After completing this unit, you should be able to:

- Properly configure your Java Virtual Machine
- Understand the impact of setting the number of simultaneous requests
- Configure Datasources
- Create a client variable repository
- Counteract problems with web spiders
- Set up Verity to operate in a client-server mode

Unit Topics

- Effectively Configuring ColdFusion and the JVM
- Configuring Database Drivers
- Configuring settings that affect scalability
- Configuring settings that affect portability
- Defining and Maintaining Verity Collections
- Patching your system

Effectively Configuring ColdFusion and the JVM



As a J2EE-based application, ColdFusion relies on the stability and performance of several products from third-parties. Clearly the most critical of these is the underlying Java Virtual Machine (JVM) which is responsible for processing most requests.

Many of the deployment problems encountered by ColdFusion Developers and administrators center around the limited amount of memory that a JVM can access, how that memory is organized, and the frequency of garbage collection whereby previously used memory is returned to the available heap.

Maximizing the performance of your server is also highly dependent on choices that you make in the ColdFusion Administrator regarding the handling of simultaneous requests. There are also a number of critical configuration settings that should be investigated that exist in separate XML files on the ColdFusion Server.

In addition, some of the mechanisms that need to be configured properly lest they adversely impact scalability include the following:

- Datasources
- Charting
- Flex form compilation
- Verity search engine

Choosing and Deploying a JVM

While ColdFusion on Windows ships with Java 6 (version 1.60_04), other Java Virtual Machines are, in fact, supported by ColdFusion. Once you have deployed another JVM, you can point ColdFusion to use it by modifying the `jvm.config` file located in `{cf install path}\runtime\bin\` or `{jrun install path}\bin\`.

You can download alternate JVMs from SUN at the following URL:

<http://java.sun.com/products/archive/index.html>

In particular, there is a documented bug in JVM 1.60_04 related to slow class file loading on startup. This bug in the JVM, in turn, leads to slower startup times for applications that rely heavily on ColdFusion components (<http://forums.sun.com/thread.jspa?threadID=5218663>).

Administrators who encounter this situation should consider rolling back to Java 5. A fix for this problem in Java 6 has been promised by Sun for the 1.60_10 release. There are also a number of bugs in 1.60_04 that have been fixed in 1.60_06 so you may need to experiment and closely review the JVM release notes. Regardless, always test your application on a different JVM in a development environment before deploying into production.

Note that CF 8 on a Macintosh system uses Java 5 by default. Mac OS X 10.5.2 introduced JVM 1.60_05, however, you will need to change your `JVM.config` file manually to use it.

Understanding JVM Limitations

Depending on your core operating system and JVM, will be limited in the amount of RAM that ColdFusion can address as detailed in the following table:

Platform/OS	Maximum Heap Size
32-bit Windows	1.2 GB
64-bit Windows	4 GB
Macintosh (32-bit)	2 GB
32-bit Linux	2 GB
64-bit Linux	2GB+

Table: JVM memory limitations

Regardless of the amount memory you hardware/OS/JVM supports, SUN recommends that you not allocate more than 2GB to any instance due to diminishing returns related to garbage collection. Note that the default configuration of ColdFusion allocates a maximum of 512MB to the heap.

While this may seem like a large address space, it is possible to develop applications that may exceed this limitation, particularly if your application performs one of the following tasks:

- Self-generates CFM files (as many content management systems do)
- Makes aggressive use of in-memory caching
- If any of the components that comprise your application have a memory leak where used address space is never returned to the heap
- Instantiates a large default session scope and answers requests from a large number of concurrent users

Note that Garbage collection can itself be a scalability bottleneck for your application in that during the GC process, the system effectively pauses execution while memory is returned to the free heap. For this reason it is important to choose the GC algorithm that best suits your application.

Modifying JVM Settings

JVM settings for the J2EE versions of CF can be modified from directly within the J2EE administrator simply by editing the `jvm.config` file directly in a text editor. ColdFusion's single-server configuration allows you to modify these options from within the CF Administrator. Perhaps the three most important settings in the `JVM.config` are the ones that pertain to memory allocation:

- Minimum JVM Heap Size
- Maximum JVM Heap Size
- JVM Arguments

The minimum JVM heap size is the amount of memory that the JVM will consume on system startup. Any time the JVM needs to allocate more RAM, you may suffer some temporary performance degradation.

The Maximum JVM Heap size corresponds to the total addressable memory space that can be referenced. On a Microsoft Windows 32-bit system this can be set as high as 1300.

There are many command line arguments that can be passed to the JVM through the JVM Arguments block. JVM arguments are completely dependent on the specific version of ColdFusion, the JVM, and operating system that you are running. If you plan on heavily modifying the `jvm.config` file, it is best to do so by editing the file directly rather than using the GUI interfaces available.

The `JVM.config` file is located in the following directory:

- Single server: `[cfinstall path]\runtime\bin\jvm.config`
- Multi server: `[jrun install path]\bin`

Note that any error in the format of the JVM arguments will prevent the ColdFusion service from starting successfully.

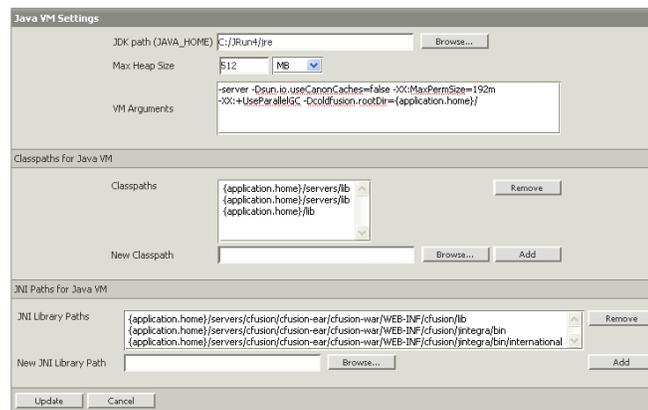


Figure: Configuring the JVM through the JRun Management Console

JVM Argument Syntax

- Boolean options are turned on with `-XX:+<option>` and turned off with `-XX:-<option>`.
- Numeric options are set with `-XX:<option>=<number>`. Numbers can include 'm' or 'M' for megabytes, 'k' or 'K' for kilobytes, and 'g' or 'G' for gigabytes (for example, 32k is the same as 32768).
- String options are set with `-XX:<option>=<string>`, are usually used to specify a file, a path, or a list of commands

Note that with each succeeding iteration of its JVM, Sun strives for *ergonomic* improvements, so that manual changes to the jvm arguments can be minimized.

Standard JVM Settings

You have probably noticed the following switches in your java.config file:

Setting	Example	Description
<code>-server</code>	<code>-server</code>	Loads the Java Hotspot server VM (as opposed to <code>-client</code> which loads the client VM)
<code>-Dproperty=value</code>	<code>-Dsun.io.useCanonCaches=false</code>	Sets a system property value

Table: Standard JVM configuration switches

JVM Memory Organization and Garbage Collection

In order to comprehend how JVM garbage collection operates, it is useful to understand that JVM memory is organized.

- The Eden space is where new objects in memory are instantiated.
- If an object in the Eden space is not immediately released, it's assigned to one of the survivor spaces.
- If an object continues to be used over a period of time it ultimately gets copied into the tenured generation space, and then finally the Permanent Generation.
- ColdFusion Application and Server variables, along with the CF template cache, live in the permanent generation space while the variables collection would exist within the Eden space.

The memory heap is depicted in the figure below:

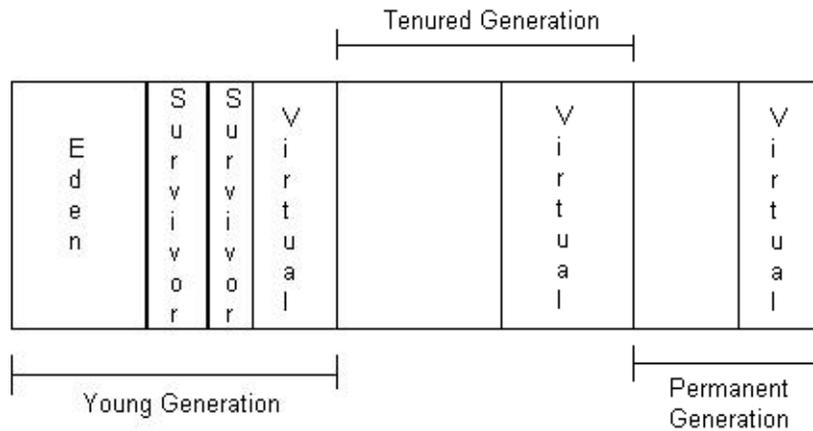


Figure: JVM memory organization

Configuring JVM Memory

The following options describe how to allocate memory to your JVM.

Setting	Example	Description
<code>XmxMemSize</code>	<code>-Xmx512m</code>	Specify the maximum size, in bytes, of the memory allocation pool. This value must be a multiple of 1024 greater than 2MB. Append the letter k or K to indicate kilobytes, or m or M to indicate megabytes. The default value is 64MB.
<code>XmsMemSize</code>	<code>-Xms512m</code>	Specify the initial size, in bytes, of the memory allocation pool. This value must be a multiple of 1024 greater than 1MB
<code>XX:MaxPermSize</code>	<code>-XX:MaxPermSize=64m</code>	Maximum size of the Permanent Generation. 64-bit VM's are 30% larger
<code>-XX:PermSize</code>	<code>-XX:Permsize=64m</code>	Initial size of the Permanent Generation
<code>-XX:NewSize</code>	<code>-XX:NewSize=48m</code>	Use this option to set the New generation Java heap size. As a general rule, set it to be 25% of the maximum heap size. Increase the value of this option for larger numbers of short-lived objects.
<code>-XX:MaxNewSize</code>	<code>-XX:MaxNewSize=256m</code>	Sets the maximum new generation heap size
<code>-Xmn</code>	<code>-Xmn128m</code>	The size of the young generation
<code>-Xss</code>	<code>-Xss256K</code>	The stack size for each thread. If the stack space is too small you will see a <code>java.lang.StackOverflowError</code>
<code>-XX:SurvivorRatio</code>	<code>-XX:SurvivorRatio=8</code>	Configure the ratio of the Eden/survivor space size. Try setting this value to 8 and then monitor your garbage collection.

Note that setting the `Xmx` and `Xms` equal to the same value results in a faster startup since new memory blocks are not being constantly allocated to the JVM.

JVM Garbage Collection Algorithms

Java supports the following garbage collection algorithms:

- The *throughput* collector: this collector uses a parallel version of the *young* generation collector. It is used if the `-XX:+UseParallelGC` option is passed on the command line. The *tenured* generation collector is the same as the serial collector. This is the default option installed for ColdFusion.
- The *concurrent* low pause collector, used if the argument `-XX:+UseConcMarkSweepGC` is specified. The concurrent collector is used to collect the *tenured* generation and does most of the collection concurrently with the execution of the application. The application is paused for short periods during the collection. A parallel version of the *young* generation copying collector is used with the concurrent collector.

Note: -XX:+UseParallelGC should not be used with -XX:+UseConcMarkSweepGC.

The Throughput collector

Use the throughput collector when you want to improve the performance of your application with larger numbers of processors. In the serial collector garbage collection is done by one thread, and therefore garbage collection adds to the serial execution time of the application. The throughput collector uses multiple threads to execute a minor collection and so reduces the serial execution time of the application. A typical situation is one in which the application has a large number of threads allocating objects. In such an application it is often the case that a large *young* generation is needed.

The throughput collector will throw an out-of-memory exception if too much time is being spent doing garbage collection. For example, if the JVM is spending more than 98% of the total time doing garbage collection and is recovering less than 2% of the heap, it will throw an out-of-memory exception.

Note that with the throughput collector, fragmentation can occur. If you suspect a problem you can reduce the number of GC threads by passing in the `-XX:ParallelGCThreads=n` option. Note that by default, the number of parallel GC threads is equal to the number of CPU's on your system.

The Concurrent Low Pause Collector

This collector attempts to reduce the pause times needed to collect the *tenured* generation. It uses a separate garbage collector thread to do parts of the major collection concurrently with the applications threads. The concurrent collector is enabled with the option `-XX:+UseConcMarkSweepGC`. For each major collection the concurrent collector will pause all the application threads for a brief period at the beginning of the collection and toward the middle of the collection. The second pause tends to be the longer of the two pauses and multiple threads are used to do the collection work during that pause. The remainder of the collection is done with a garbage collector thread that runs concurrently with the application. The minor collections are done in a manner similar to the serial collector although multiple threads are used to do the collection. This is the best option for systems with large numbers of CPU's as pauses are essentially eliminated with collection performed by separate threads.

Configuring the Garbage Collector

You can set the following options in your `jvm.config` file to configure the behavior of your Java garbage collection:

Setting	Description
<code>-XX:+UseConcMarkSweepGC</code>	Enables the low-pause collector. Typically used with <code>-XX:+UseParNewGC</code> for multi-cpu servers
<code>-XX:+UseParNewGC</code>	Option for the low pause collector. Enables multi threaded young generation collection for multi-core servers.
<code>-XX:+UseParallelGC</code>	CF default setting, enables the throughput collector.
<code>-XX:ParallelGCThreads=n</code>	Sets the number of threads used by the throughput collector. Reduce the number of fragmentation occurs.
<code>-Dsun.rmi.dgc.client.gcInterval=300000</code> <code>-Dsun.rmi.dgc.server.gcInterval=300000</code>	Tells the JVM to perform a full GC every 300000 milliseconds

Table: Garbage collection algorithm switches in `jvm.config`

Troubleshooting your memory configuration

To a certain extent, JVM memory can be monitored through the use of the ColdFusion Server Monitor (unit 9). However, a number of other tools and logging functions exist to help you analyze your Java heap including the following:

Setting	Description
<code>-XX:+HeapDumpOnOutOfMemoryError</code>	Performs a heap dump to disk when a <code>java.lang.outofmemory</code> error is encountered. Dumps heap to an <code>.hprof</code> file that can then be analyzed with <code>jmap</code> , or <code>jhat</code> , or the eclipse-based <code>mat</code> (memory analyzer tool).
<code>-XX:HeapDumpPath=[dir]</code>	Sets the directory where the HPROF file will be stored
<code>-XX:OnError=</code>	Execute an external application when an error occurs

In addition, you might consider using 3rd party Java debuggers like JProbe or Oracle JRockit Mission Control.

Programmatically Requesting GC and Memory Status

Since ColdFusion is a Java application, you can programmatically get access to the JVM and request a garbage collection event take place using the following code:

```
<cfscript>
  obj = createObject("java","java.lang.System");
  obj.gc();
  obj.runFinalization();
</cfscript>
```

Note that you will also be able to request a GC event when memory usage crosses a set threshold by using a ColdFusion Monitor Alert (covered in unit 9).

In addition, the following code will display your current memory status:

```
<cfset runtime = CreateObject("java","java.lang.Runtime")>
<cfoutput>
Max Heap:
#round(runtime.getRuntime().maxMemory() / 1000000)# <br />
Current Heap:
#round(runtime.getRuntime().totalMemory()/1000000)# <br />
Free Heap:
#round(runtime.getRuntime().freeMemory() / 1000000)#<br />
</cfoutput>
```

Additional tool for introspecting the heap

Steven Brownlee, a ColdFusion developer, created a Java .JAR file that when integrated with ColdFusion gives you programmatic introspection of the heap through ColdFusion's SERVER variables. You can download this code at the following address:

<http://www.fusioncube.net/index.php/scope-enhancer>

JVM																																																																																																																																						
ARGUMENTS	<table border="1"> <thead> <tr> <th colspan="2">array</th> </tr> </thead> <tbody> <tr><td>1</td><td>-Xms75m</td></tr> <tr><td>2</td><td>-Xmx125m</td></tr> <tr><td>3</td><td>-Djava.library.path=../lib</td></tr> <tr><td>4</td><td>-Dwrapper.key=phPUVzVVM1cu31L</td></tr> <tr><td>5</td><td>-Dwrapper.port=32000</td></tr> <tr><td>6</td><td>-Dwrapper.use_system_time=TRUE</td></tr> <tr><td>7</td><td>-Dwrapper.version=3.1.2</td></tr> <tr><td>8</td><td>-Dwrapper.native_library=wrapper</td></tr> <tr><td>9</td><td>-Dwrapper.service=TRUE</td></tr> <tr><td>10</td><td>-Dwrapper.cpu.timeout=10</td></tr> <tr><td>11</td><td>-Dwrapper.jvmid=1</td></tr> </tbody> </table>	array		1	-Xms75m	2	-Xmx125m	3	-Djava.library.path=../lib	4	-Dwrapper.key=phPUVzVVM1cu31L	5	-Dwrapper.port=32000	6	-Dwrapper.use_system_time=TRUE	7	-Dwrapper.version=3.1.2	8	-Dwrapper.native_library=wrapper	9	-Dwrapper.service=TRUE	10	-Dwrapper.cpu.timeout=10	11	-Dwrapper.jvmid=1																																																																																																													
array																																																																																																																																						
1	-Xms75m																																																																																																																																					
2	-Xmx125m																																																																																																																																					
3	-Djava.library.path=../lib																																																																																																																																					
4	-Dwrapper.key=phPUVzVVM1cu31L																																																																																																																																					
5	-Dwrapper.port=32000																																																																																																																																					
6	-Dwrapper.use_system_time=TRUE																																																																																																																																					
7	-Dwrapper.version=3.1.2																																																																																																																																					
8	-Dwrapper.native_library=wrapper																																																																																																																																					
9	-Dwrapper.service=TRUE																																																																																																																																					
10	-Dwrapper.cpu.timeout=10																																																																																																																																					
11	-Dwrapper.jvmid=1																																																																																																																																					
MEMORY	<table border="1"> <thead> <tr> <th colspan="2">struct</th> </tr> </thead> <tbody> <tr><td>FREEMEMORY</td><td>36272352</td></tr> <tr><td>HEAPUSED</td><td>41912096</td></tr> <tr><td>MAXMEMORY</td><td>131137536</td></tr> <tr><td>NONHEAPUSED</td><td>45967536</td></tr> <tr> <td>POOLS</td> <td> <table border="1"> <thead> <tr> <th colspan="7">query - Rows: 7</th> </tr> <tr> <th></th> <th>COMMITTED</th> <th>ID</th> <th>INIT</th> <th>MAX</th> <th>POOLNAME</th> <th>USED</th> </tr> </thead> <tbody> <tr><td>1</td><td>5079040</td><td>1</td><td>196608</td><td>33554432</td><td>Code Cache</td><td>5066368</td></tr> <tr><td>2</td><td>4980736</td><td>2</td><td>4849664</td><td>8192000</td><td>Eden Space</td><td>1537552</td></tr> <tr><td>3</td><td>589824</td><td>3</td><td>589824</td><td>983040</td><td>Survivor Space</td><td>458232</td></tr> <tr><td>4</td><td>72613888</td><td>4</td><td>72613888</td><td>121962496</td><td>Tenured Gen</td><td>39840456</td></tr> <tr><td>5</td><td>29884416</td><td>5</td><td>8388608</td><td>67108864</td><td>Perm Gen</td><td>29782712</td></tr> <tr><td>6</td><td>8388608</td><td>6</td><td>8388608</td><td>8388608</td><td>Perm Gen [shared-ro]</td><td>5284768</td></tr> <tr><td>7</td><td>12582912</td><td>7</td><td>12582912</td><td>12582912</td><td>Perm Gen [shared-rw]</td><td>5833688</td></tr> </tbody> </table> </td> </tr> <tr> <td>THREADS</td> <td> <table border="1"> <thead> <tr> <th colspan="7">query - Rows: 37</th> </tr> <tr> <th></th> <th>ALIVE</th> <th>DAEMON</th> <th>GROUP</th> <th>ID</th> <th>INTERRUPTED</th> <th>NAME</th> </tr> </thead> <tbody> <tr><td>1</td><td>1</td><td>1</td><td>java.lang.ThreadGroup[name=jboss,maxpri=10]</td><td>15</td><td>0</td><td>Timer-0</td></tr> <tr><td>2</td><td>1</td><td>1</td><td>java.lang.ThreadGroup[name=jboss,maxpri=10]</td><td>16</td><td>0</td><td>ScannerThread</td></tr> <tr><td>3</td><td>1</td><td>1</td><td>java.lang.ThreadGroup[name=jboss,maxpri=10]</td><td>20</td><td>0</td><td>Timer-1</td></tr> <tr><td>4</td><td>1</td><td>0</td><td>java.lang.ThreadGroup[name=jboss,maxpri=10]</td><td>25</td><td>0</td><td>PooledInvokerAcceptor#0-44</td></tr> <tr><td>5</td><td>1</td><td>1</td><td>java.lang.ThreadGroup[name=jboss,maxpri=10]</td><td>26</td><td>0</td><td>ContainerBackgroundProcess</td></tr> <tr><td>6</td><td>1</td><td>0</td><td>java.lang.ThreadGroup[name=jboss,maxpri=10]</td><td>27</td><td>0</td><td>SubscriptionWatchDog</td></tr> </tbody> </table> </td> </tr> </tbody> </table>	struct		FREEMEMORY	36272352	HEAPUSED	41912096	MAXMEMORY	131137536	NONHEAPUSED	45967536	POOLS	<table border="1"> <thead> <tr> <th colspan="7">query - Rows: 7</th> </tr> <tr> <th></th> <th>COMMITTED</th> <th>ID</th> <th>INIT</th> <th>MAX</th> <th>POOLNAME</th> <th>USED</th> </tr> </thead> <tbody> <tr><td>1</td><td>5079040</td><td>1</td><td>196608</td><td>33554432</td><td>Code Cache</td><td>5066368</td></tr> <tr><td>2</td><td>4980736</td><td>2</td><td>4849664</td><td>8192000</td><td>Eden Space</td><td>1537552</td></tr> <tr><td>3</td><td>589824</td><td>3</td><td>589824</td><td>983040</td><td>Survivor Space</td><td>458232</td></tr> <tr><td>4</td><td>72613888</td><td>4</td><td>72613888</td><td>121962496</td><td>Tenured Gen</td><td>39840456</td></tr> <tr><td>5</td><td>29884416</td><td>5</td><td>8388608</td><td>67108864</td><td>Perm Gen</td><td>29782712</td></tr> <tr><td>6</td><td>8388608</td><td>6</td><td>8388608</td><td>8388608</td><td>Perm Gen [shared-ro]</td><td>5284768</td></tr> <tr><td>7</td><td>12582912</td><td>7</td><td>12582912</td><td>12582912</td><td>Perm Gen [shared-rw]</td><td>5833688</td></tr> </tbody> </table>	query - Rows: 7								COMMITTED	ID	INIT	MAX	POOLNAME	USED	1	5079040	1	196608	33554432	Code Cache	5066368	2	4980736	2	4849664	8192000	Eden Space	1537552	3	589824	3	589824	983040	Survivor Space	458232	4	72613888	4	72613888	121962496	Tenured Gen	39840456	5	29884416	5	8388608	67108864	Perm Gen	29782712	6	8388608	6	8388608	8388608	Perm Gen [shared-ro]	5284768	7	12582912	7	12582912	12582912	Perm Gen [shared-rw]	5833688	THREADS	<table border="1"> <thead> <tr> <th colspan="7">query - Rows: 37</th> </tr> <tr> <th></th> <th>ALIVE</th> <th>DAEMON</th> <th>GROUP</th> <th>ID</th> <th>INTERRUPTED</th> <th>NAME</th> </tr> </thead> <tbody> <tr><td>1</td><td>1</td><td>1</td><td>java.lang.ThreadGroup[name=jboss,maxpri=10]</td><td>15</td><td>0</td><td>Timer-0</td></tr> <tr><td>2</td><td>1</td><td>1</td><td>java.lang.ThreadGroup[name=jboss,maxpri=10]</td><td>16</td><td>0</td><td>ScannerThread</td></tr> <tr><td>3</td><td>1</td><td>1</td><td>java.lang.ThreadGroup[name=jboss,maxpri=10]</td><td>20</td><td>0</td><td>Timer-1</td></tr> <tr><td>4</td><td>1</td><td>0</td><td>java.lang.ThreadGroup[name=jboss,maxpri=10]</td><td>25</td><td>0</td><td>PooledInvokerAcceptor#0-44</td></tr> <tr><td>5</td><td>1</td><td>1</td><td>java.lang.ThreadGroup[name=jboss,maxpri=10]</td><td>26</td><td>0</td><td>ContainerBackgroundProcess</td></tr> <tr><td>6</td><td>1</td><td>0</td><td>java.lang.ThreadGroup[name=jboss,maxpri=10]</td><td>27</td><td>0</td><td>SubscriptionWatchDog</td></tr> </tbody> </table>	query - Rows: 37								ALIVE	DAEMON	GROUP	ID	INTERRUPTED	NAME	1	1	1	java.lang.ThreadGroup[name=jboss,maxpri=10]	15	0	Timer-0	2	1	1	java.lang.ThreadGroup[name=jboss,maxpri=10]	16	0	ScannerThread	3	1	1	java.lang.ThreadGroup[name=jboss,maxpri=10]	20	0	Timer-1	4	1	0	java.lang.ThreadGroup[name=jboss,maxpri=10]	25	0	PooledInvokerAcceptor#0-44	5	1	1	java.lang.ThreadGroup[name=jboss,maxpri=10]	26	0	ContainerBackgroundProcess	6	1	0	java.lang.ThreadGroup[name=jboss,maxpri=10]	27	0	SubscriptionWatchDog
struct																																																																																																																																						
FREEMEMORY	36272352																																																																																																																																					
HEAPUSED	41912096																																																																																																																																					
MAXMEMORY	131137536																																																																																																																																					
NONHEAPUSED	45967536																																																																																																																																					
POOLS	<table border="1"> <thead> <tr> <th colspan="7">query - Rows: 7</th> </tr> <tr> <th></th> <th>COMMITTED</th> <th>ID</th> <th>INIT</th> <th>MAX</th> <th>POOLNAME</th> <th>USED</th> </tr> </thead> <tbody> <tr><td>1</td><td>5079040</td><td>1</td><td>196608</td><td>33554432</td><td>Code Cache</td><td>5066368</td></tr> <tr><td>2</td><td>4980736</td><td>2</td><td>4849664</td><td>8192000</td><td>Eden Space</td><td>1537552</td></tr> <tr><td>3</td><td>589824</td><td>3</td><td>589824</td><td>983040</td><td>Survivor Space</td><td>458232</td></tr> <tr><td>4</td><td>72613888</td><td>4</td><td>72613888</td><td>121962496</td><td>Tenured Gen</td><td>39840456</td></tr> <tr><td>5</td><td>29884416</td><td>5</td><td>8388608</td><td>67108864</td><td>Perm Gen</td><td>29782712</td></tr> <tr><td>6</td><td>8388608</td><td>6</td><td>8388608</td><td>8388608</td><td>Perm Gen [shared-ro]</td><td>5284768</td></tr> <tr><td>7</td><td>12582912</td><td>7</td><td>12582912</td><td>12582912</td><td>Perm Gen [shared-rw]</td><td>5833688</td></tr> </tbody> </table>	query - Rows: 7								COMMITTED	ID	INIT	MAX	POOLNAME	USED	1	5079040	1	196608	33554432	Code Cache	5066368	2	4980736	2	4849664	8192000	Eden Space	1537552	3	589824	3	589824	983040	Survivor Space	458232	4	72613888	4	72613888	121962496	Tenured Gen	39840456	5	29884416	5	8388608	67108864	Perm Gen	29782712	6	8388608	6	8388608	8388608	Perm Gen [shared-ro]	5284768	7	12582912	7	12582912	12582912	Perm Gen [shared-rw]	5833688																																																																						
query - Rows: 7																																																																																																																																						
	COMMITTED	ID	INIT	MAX	POOLNAME	USED																																																																																																																																
1	5079040	1	196608	33554432	Code Cache	5066368																																																																																																																																
2	4980736	2	4849664	8192000	Eden Space	1537552																																																																																																																																
3	589824	3	589824	983040	Survivor Space	458232																																																																																																																																
4	72613888	4	72613888	121962496	Tenured Gen	39840456																																																																																																																																
5	29884416	5	8388608	67108864	Perm Gen	29782712																																																																																																																																
6	8388608	6	8388608	8388608	Perm Gen [shared-ro]	5284768																																																																																																																																
7	12582912	7	12582912	12582912	Perm Gen [shared-rw]	5833688																																																																																																																																
THREADS	<table border="1"> <thead> <tr> <th colspan="7">query - Rows: 37</th> </tr> <tr> <th></th> <th>ALIVE</th> <th>DAEMON</th> <th>GROUP</th> <th>ID</th> <th>INTERRUPTED</th> <th>NAME</th> </tr> </thead> <tbody> <tr><td>1</td><td>1</td><td>1</td><td>java.lang.ThreadGroup[name=jboss,maxpri=10]</td><td>15</td><td>0</td><td>Timer-0</td></tr> <tr><td>2</td><td>1</td><td>1</td><td>java.lang.ThreadGroup[name=jboss,maxpri=10]</td><td>16</td><td>0</td><td>ScannerThread</td></tr> <tr><td>3</td><td>1</td><td>1</td><td>java.lang.ThreadGroup[name=jboss,maxpri=10]</td><td>20</td><td>0</td><td>Timer-1</td></tr> <tr><td>4</td><td>1</td><td>0</td><td>java.lang.ThreadGroup[name=jboss,maxpri=10]</td><td>25</td><td>0</td><td>PooledInvokerAcceptor#0-44</td></tr> <tr><td>5</td><td>1</td><td>1</td><td>java.lang.ThreadGroup[name=jboss,maxpri=10]</td><td>26</td><td>0</td><td>ContainerBackgroundProcess</td></tr> <tr><td>6</td><td>1</td><td>0</td><td>java.lang.ThreadGroup[name=jboss,maxpri=10]</td><td>27</td><td>0</td><td>SubscriptionWatchDog</td></tr> </tbody> </table>	query - Rows: 37								ALIVE	DAEMON	GROUP	ID	INTERRUPTED	NAME	1	1	1	java.lang.ThreadGroup[name=jboss,maxpri=10]	15	0	Timer-0	2	1	1	java.lang.ThreadGroup[name=jboss,maxpri=10]	16	0	ScannerThread	3	1	1	java.lang.ThreadGroup[name=jboss,maxpri=10]	20	0	Timer-1	4	1	0	java.lang.ThreadGroup[name=jboss,maxpri=10]	25	0	PooledInvokerAcceptor#0-44	5	1	1	java.lang.ThreadGroup[name=jboss,maxpri=10]	26	0	ContainerBackgroundProcess	6	1	0	java.lang.ThreadGroup[name=jboss,maxpri=10]	27	0	SubscriptionWatchDog																																																																													
query - Rows: 37																																																																																																																																						
	ALIVE	DAEMON	GROUP	ID	INTERRUPTED	NAME																																																																																																																																
1	1	1	java.lang.ThreadGroup[name=jboss,maxpri=10]	15	0	Timer-0																																																																																																																																
2	1	1	java.lang.ThreadGroup[name=jboss,maxpri=10]	16	0	ScannerThread																																																																																																																																
3	1	1	java.lang.ThreadGroup[name=jboss,maxpri=10]	20	0	Timer-1																																																																																																																																
4	1	0	java.lang.ThreadGroup[name=jboss,maxpri=10]	25	0	PooledInvokerAcceptor#0-44																																																																																																																																
5	1	1	java.lang.ThreadGroup[name=jboss,maxpri=10]	26	0	ContainerBackgroundProcess																																																																																																																																
6	1	0	java.lang.ThreadGroup[name=jboss,maxpri=10]	27	0	SubscriptionWatchDog																																																																																																																																

At a glance this tool gives you direct programmatic reporting on the memory pools and threads currently executing in your JVM.

More information regarding JVM Arguments and performance

Administrators running high performance sites might find the following URL's helpful:

- Frequently asked questions regarding JVM Garbage Collection:
<http://java.sun.com/docs/hotspot/gc1.4.2/faq.html>
- JVM Tuning:
http://java.sun.com/docs/hotspot/gc5.0/gc_tuning_5.html#1.1.Ergonomics%7Coutline
- Java Performance Docs:
<http://java.sun.com/docs/performance/>
- Java Performance Strategies and Tactics:
<http://java.sun.com/docs/books/performance/>
- Troubleshooting Guide for Java 6:
<http://java.sun.com/javase/6/webnotes/trouble/TSG-VM/html/clopts.html>

Walkthrough 5-1: Configuring your JVM



In this walkthrough you will activate a number of configuration options to modify the behavior of your Java Virtual Machine.

Steps

1. Using Eclipse, open the `jvm.config` file located in `C:\Jrun4\bin\`
2. Save it out to backup file named `jvm.original.config`
3. Insert the following argument after the `-Xmx512M`:
`-Xms256M` and discuss the impact with your instructor.
4. Replace the `-XX:+UseParallelGC` argument with the following and discuss the impact with your instructor:

`-XX:+UseConcMarkSweepGC -XX:+UseParNewGC`
5. Add the following argument and discuss the impact with your instructor:

`-XX:+HeapDumpOnOutOfMemoryError`
6. Add the following arguments and discuss the impact with your instructor:

`-Dsun.rmi.dgc.client.gcInterval=300000`
`-Dsun.rmi.dgc.server.gcInterval=300000`
7. Save the file
8. Copy the file `\admincf\misc\enhancer.jar` to
`C:\jrun4\servers\public1\cfusion.ear\cfusion.war\WEB-INF\lib` and to
`C:\jrun4\servers\public2\cfusion.ear\cfusion.war\WEB-INF\lib`
9. Restart the production instances
10. Open a web browser and run
<http://www.cafetownsend.com/dumpjvmstatus.cfm>
11. Review the contents with your instructor

Configuring Database Drivers



Using the ColdFusion Administrator, you will configure datasources that contain connection information to the target database. Developers, in turn, use the datasource names when defining queries or referencing stored procedures in their code. ColdFusion ships with database drivers from DataDirect. These drivers undergo frequent revisions and new versions become available from Adobe in the form of hot fixes. You can also opt to use the vendor's native JDBC driver, if available.

Supported databases

ColdFusion allows you to configure data sources for the following databases:

- Apache Derby 10.2.2.0
- MS Access 2000-2007
- MS SQL Server 7.0,2000,2005
- MySQL 3,4.1, and 5.0
- PostgreSQL 8.1
- DB2 7.x,8.x,9.1 (windows)
- DB2 7.x,8.1 (OS/390)
- DB2 for iSeries V5R1, V5R2, V5R3, V5R4
- Informix Dynamic Server 9.2x, 9.3x, 9.4x, 10.x
- Oracle 8i R2-R3 8.1.7
- Oracle 9i R1-R2 including RAC support
- Oracle 10g R1-R2 including RAC support
- Sybase Adaptive Server 11.5
- Sybase Adaptive Server Enterprise 12.0, 12.5.x, 15, and 15.5.x

In addition, ColdFusion will also allow you to connect to ODBC datasources through the `odbc-jdbc` bridge, however, performance is usually poor in this scenario.

You can also opt to use the JDBC driver that ships directly from your database vendor. Problems with database drivers generally manifest as slow query performance and/or memory leaks.

- Download the Microsoft SQL Server driver at the following URL: <http://www.microsoft.com/downloads/details.aspx?familyid=07287b11-0502-461a-b138-2aa54bfdc03a&displaylang=en>
- Download the Oracle Drivers at the following URL: http://www.oracle.com/technology/software/tech/java/sqlj_jdbc/index.html

Basic data source configuration options

Minimally, a datasource is comprised of the following:

- The selection of a driver
- A name
- A path to the database or ip address / server name / port
- A database login
- A text description of the datasource

The screenshot shows the configuration page for a Microsoft SQL Server datasource. The breadcrumb trail is "Data & Services > Datasources > Microsoft SQL Server". The title of the page is "Microsoft SQL Server : Test". The form contains the following fields:

- CF Data Source Name:
- Database:
- Server: Port:
- User name:
- Password: (16-character limit)
- Description:

At the bottom of the form, there are three buttons: "Show Advanced Settings", "Submit", and "Cancel".

Figure: SQL Server Driver Basic Options

Advanced Datasource Configuration Options

The screenshot shows the 'Advanced Datasource Configuration Options' for the SQL Server driver. Key settings include:

- Connection String:** An empty text box.
- Select Method:** A dropdown menu set to 'Direct'.
- Limit Connections:** An unchecked checkbox with an empty text box for the limit value.
- Maintain Connections:** A checked checkbox with the description '-- Maintain connections across client requests.'
- String Format:** An unchecked checkbox with the description '-- Enable High ASCII characters and Unicode for data sources configured for non-Latin characters.'
- Max Pooled Statements:** A text box containing '1000'.
- Timeout (min):** A text box containing '20'.
- Interval (min):** A text box containing '7'.
- Log Activity:** An unchecked checkbox with a text box for the log file path and a 'Browse Server' button.
- Disable Connections:** An unchecked checkbox with the description '-- Suspend all client connections.'
- Login Timeout (sec):** A text box containing '30'.
- CLOB:** An unchecked checkbox with the description '-- Enable long text retrieval (CLOB).'
- BLOB:** An unchecked checkbox with the description '-- Enable binary large object retrieval (BLOB).'
- Long Text Buffer (chr):** A text box containing '64000'.
- Blob Buffer (bytes):** A text box containing '64000'.
- Allowed SQL:** A list of SQL operations with checkboxes: SELECT, INSERT, UPDATE, DELETE, Create, DROP, ALTER, GRANT, REVOKE, and Stored Procedures. All are checked.
- Validation Query:** An empty text box.

Figure: SQL Server driver advanced configuration options

Advanced datasource options allow you to configure the following:

Setting	Description
Connection String	Passes database-specific parameters to the datasource
Timeout	The number of minutes that CF will retain a connection before closing it
Timeout Interval	The interval that CF will check to see if a connection has reached its timeout threshold.
Select Method	The Direct method provides more efficient retrieval of data when you retrieve record sets in a forward-only direction and you limit your SQL Server connection to a single open SQL statement at a time. This is typical for ColdFusion applications. The Cursor method lets you have multiple open SQL statements on a connection. This is not typical for ColdFusion applications, unless you use pooled statements.
Limit Connections	If selected, restricts ColdFusion to creating the number of connections specified. Since databases are often licensed by number of concurrent connections this option allows you to maintain compliancy.
Maintain	Improves performance by caching the database

Setting	Description
Connections	connection instead of recreating the connection for each db request
String Format	Enable this checkbox if your application must support the Unicode character set. Effectively, it changes the sql syntax for how data is passed into the database when using <cfqueryparam> and <cfpropparam> tags.
Max Pooled Statements	Enables reuse of prepared statements that use <cfqueryparam> or stored procedures. It should be set equal to the number of unique queries that you have in your application which use <cfqueryparam> or <cfstoredproc>
Log Activity	Allows you to write all database requests to a log file
Disable Connections	Suspends all connections from ColdFusion to the database. Useful for copying/overwriting file-based databases like Microsoft Access.
Login Timeout	The time (in seconds) that ColdFusion will take to login before giving up and generating an error.
CLOB	When checked, allows ColdFusion to retrieve the entire contents of a CLOB/Text column. If unchecked, CF will only retrieve the amount of text specified in the Long Text Buffer (64K by default). Should be checked if the database supports memo/text/clob data types.
BLOB	When checked, allows ColdFusion to retrieve the entire contents of an Image/Blob data column. Otherwise it only retrieves the amount of data specified in the BLOB buffer (64K) by default
Allowed SQL	Disallows any SQL statements that are not checked
Validation Query	Called when a connection from the pool is reused. Can slow query response time because an additional query is generated. You should specify this just before restarting the database to verify all connections, but remove the validation query after restarting the database to avoid continuing overhead.

Datasource Best Practices

- Each datasource should use a separate and unique username/password so that if the database account gets compromised, only one database is at risk.
- File-based databases (e.g. Microsoft Access) should be located on the same physical hardware as the ColdFusion server in order to minimize network traffic.
- File-based databases should not be http accessible.
- Client-server databases (e.g. SQL Server, ORACLE) should be installed on their own hardware, separate and distinct from the ColdFusion server.
- As a security precaution, consider configuring your database to listen on a port other than the default (e.g. 1433 for sql server)
- The Microsoft Access with Unicode driver is compatible with the <cfdbinfo> tag while the regular Microsoft Access driver is not.
- Apache Derby now ships as an embedded component of ColdFusion 8. It is therefore better supported across a wider range of operating systems than Microsoft Access if you need to ship an embedded database with your application.
- If you experience errors reporting too many open connections on your SQL/ORACLE server, consider turning off the Maintain Connections option in the database driver.
- Security of your data can be more tightly controlled by giving developers access to views and stored procedures rather than direct table access.

Walkthrough 5-2: Configuring Datasources



In this walkthrough you investigate how to configure the ColdFusion server for optimal performance.

Steps

1. Open the ColdFusion Administrator for the development instance by browsing to the following URL:
<http://dev.cafetownsend.com/cfide/administrator/>.
2. From the left-side navigation panel, select **Data & Services > Data Sources**
3. Enter a data source name of **admincf800-solution**
4. Select a driver of **Microsoft Access with Unicode**
5. Click the **Add** button
6. Click on the **Browse Server** button
7. Select the following file:
C:\ websites\devserver\cafetownsend\database\admincf800-solution.mdb
8. Click **Apply**
9. Click **Show Advanced Settings**
10. Enable **Maintain Connections**
11. Turn on the checkbox for **CLOB**
12. Click the **submit** button

-- End of Walkthrough --

Configuring settings that affect scalability



There are a number of settings within the ColdFusion Administrator that can significantly affect the performance of your application. These include the following:

- Request Limits
- Maximum number of running JRun threads
- Request Timeout
- Maximum size of post data
- Enable whitespace management
- Enable global script protection
- Maximum number of cached templates
- Trusted cache
- Save class files
- Cache web server paths
- Maximum number of cached queries
- Memory variable timeouts
- Client variable configuration

Properly configuring your CF instance through the ColdFusion Administrator can return performance gains of 30% or more.

Request Limits

This setting restricts the number of concurrent threads that ColdFusion can process. Additional requests will be queued. Essentially this setting controls how much of a time-slice each request will receive from the CPU.

Adobe generally recommends a setting of 4-5 requests per installed CPU for optimal performance. For instance, a server that contains two processors (or cores) might initially be set to handle 8 concurrent requests. Determining the optimal setting, however, involves capturing performance statistics by simulating traffic against your application varying the value of this setting which is discussed further in unit 9.

Request limits are segmented into the following areas:

- **Maximum number of simultaneous Template requests** - typically browser-based requests to CFM files.
- **Maximum number of simultaneous Flash Remoting requests** - typically requests that might come from a Flash .SWF application.
- **Maximum number of simultaneous Web Service requests** - requests that likely come from affiliate servers.
- **Maximum number of simultaneous CFC function requests** - This refers to CFC function requests that are made via HTTP – typically used by AJAX-based clients.

Additional Request Tuning Limits

Other request tuning limits involve the following:

- **Maximum number of running JRun Threads**
This is the sum of the ColdFusion requests and any additional requests handled by JRun including JSP Pages
- **Maximum number of queued JRun Threads**
This is the total number of threads that JRun will accept at any instant.
- **Maximum number of simultaneous Report threads**
This is the total number of requests to .CFR files or requests issued by the <cfreport> tag that can be handled at any instant.
- **Maximum number of threads available for CFTHREAD**
ColdFusion 8 allows developers to fork their code into concurrently processing requests through the <cfthread> tag. Consult with your developers as to how they use this tag in their applications.
- **Timeout requests in queue after X seconds**
This feature cancels any requests that have been queued for the specified time limit and should be set as long as the request timeout. While most users will abandon their requests after waiting for 8 seconds, ColdFusion will continue to process the request until it is complete or reaches a timeout point.
- **Request Queue Timeout Page**
This feature allows you to specify your own "server busy" page if a request is cancelled while it is still in the request queue. The page may not contain any CFML.

Additional Server Settings

ColdFusion has a number of other settings that can directly impact performance located under the Server Settings > Settings link in the ColdFusion Administrator.

Request Timeout

This option enables you to cancel requests that take longer than a specified number of seconds to complete execution. Since long running requests tie-up available request slots, you should take care to choose an appropriate threshold for your system. Note that you should also consider the "eight-second rule" which dictates that most users will cancel a request if they do not receive a response from the server within an eight-second threshold. Developers can specifically override this setting for known long-running processes (e.g. verity collection indexing, large file processing) by using the `<cfsetting requestTimeout="n">` tag in their code.

Disable CFC Type Check

This option, which should only be activated in a production environment, instructs ColdFusion to skip the validation of CFC arguments thereby marginally enhancing your server performance.

Maximum Size of Post Data

These settings exist to help you manage peak memory demand for your ColdFusion server. In previous versions of ColdFusion if several users decided to upload 200MB files concurrently through an HTTP Post operation, your system could theoretically run out of JVM space. Starting with ColdFusion 7.0.1+ memory intensive form posts can be queued until enough JVM space becomes available to process them.

Enable Whitespace Management

This feature reduces the size of your outgoing data stream by removing many of the extra spaces, tabs, and carriage returns that ColdFusion might otherwise persist from your CFML source files. Turning this feature on, however, is not a panacea for handling whitespace. You will still need to explicitly manage whitespace generation through the `<cfsilent>`, `<cfsetting>`, and `<cfprocessingdirective>` tags. You can also explicitly manage whitespace through the `onRequest()` handler in the `Application.cfc` file.

Enable Global Script Protection

Cross-site scripting attacks involve hackers inputting rogue HTML into your data entry forms (usually textareas or rich-text fields) that invoke JavaScript, Java Applets, or client-side ActiveX controls when form data is viewed in a browser. Due to the nature of web scripting, there are virtually an infinite number of different exploits a hacker could choose to use.

The Enable Global Script Protection feature seeks to help guard against these types of attacks by removing any `<object>`, `<applet>`, `<embed>`, `<script>`, and `<meta>` tags from the incoming data stream.

Unfortunately, however, the only way to completely guard against cross-site scripting is to disallow your users from inputting any HTML tags. Alternately you could output the contents of user-supplied data using the `HTMLEditFormat()` function which escapes all greater than/less than symbols with `>` and `<` respectively.

To further guard against cross-scripting attacks, the `<cftextarea>` tag in version 7.01+ has a new attribute, `html="true|false"` which lets you set the value as HTML or text when loading the form.

Maximum number of Cached Templates

This setting reduces hard disk activity by reading requested ColdFusion pages and then caching the compiled code in server memory. Since cached code occupies memory in the JVM you may encounter difficulties if your code base is exceptionally large.

The general recommendation is to initially configure this setting for no more than 1000 files.

As you monitor JVM memory utilization, you can increase or decrease this number as appropriate. Note, however, that in high traffic environments, ColdFusion will actually cache more than the number of files that you specify and then pare memory consumption back during a garbage collection event.

Trusted Cache

On each page request, ColdFusion will check the timestamp of the file being requested against the timestamp of the file it has in its template cache. If the timestamps are different, ColdFusion recompiles the file on disk, placing it into its template memory cache.

By enabling this feature, you essentially tell ColdFusion to assume that the source code on the server's hard disk has not changed, therefore, it should not bother performing the time/date comparison.

Activating this feature can give you a significant performance benefit, however, it should never be enabled on development servers or on production servers where the code base is subject to frequent modification.

Save Class Files

When ColdFusion MX+ reads a CFM file, it translates it into Java before executing it. This option allows you to save the compiled Java code to disk so that in the event of a system restart or template cache-miss, the CFM file would not need to be recompiled.

Compiled CFM pages are located in the following directory:

```
{CF Install Path}\wwwroot\web-inf\cfclasses\
```

You should periodically monitor the number of files in this directory. If the number of files in this directory gets too large it can actually take ColdFusion longer to locate and read the cached file than if it were to simply recompile the requested page.

We recommend that if the number of files grows to more than 5000, you might want to consider disabling this feature. Also, improvements to the compiler in ColdFusion 7 now make this feature somewhat obsolete.

Cache Web Server Paths

ColdFusion allows you to cache web server paths.

If you plan to host a single website on your ColdFusion server, then this option should be checked.

Maximum Number of Cached Queries

ColdFusion allows you to cache query results in memory using the `CachedWithin` and `CachedAfter` attributes of the `<cfquery>` tag. Caching queries can significantly improve the performance of your application; however, caching too much data can be detrimental to your system. Since this setting is not correlated with a specific memory size, you will need to carefully monitor your memory utilization to determine its appropriate value.

Configuring Memory Variables

ColdFusion enables you to configure the following options:

- Using J2EE Session Variables
- Application variable timeouts
- Session variable timeouts

Using J2EE Session Variables

As previously mentioned in unit 4, J2EE session variables must be enabled in order for session information to be transmitted between clustered server instances. In addition, this feature allows you to share variables more easily with other J2EE constructs such as JSP pages and tag libraries.

Activating this feature disables ColdFusion using cookies to represent session identifiers through two cookies named CFID and CFTOKEN.

Timing out Application Variables

ColdFusion allows you to specify the maximum timeout and default timeout for ColdFusion Application variables. These variables are shared among all users of an application (scoped through the `<cfapplication>` or `application.cfc` application name). The inactivity timer starts running once the last page request to an application has been made. A maximum and default timeout value of 2 days likely ensures that your application variables will never be timed out or deleted unless your application is used very sporadically. This is generally not a problem since Application variables are typically used to hold global constants and therefore usually consume only a small part of the permanent generation in memory.

Timing out Session Variables

Session variables, by default expire after 20 minutes of inactivity. The challenge in configuring this value is that in the stateless environment of HTTP, the countdown timer starts running immediately after the last request from a user is served. It is not uncommon, therefore, for a user's session to be timed-out while they are performing actions that do not require a server transaction such as filling out an HTML or PDF form. Developers should take care to engineer client-side solutions to explicitly manage sessions (such as creating an autosave feature or alert in JavaScript). As an Administrator, you should not significantly increase this value for the following reasons:

- Security
A user leaving their machine unattended should want their session to be terminated after a relatively short period of time
- Memory management
Timed out session variables are garbage collected and returned to the free heap
- Bots
Page scanning bots (such as the GoogleBot, Yahoo Slurp!, or CFHTTP) do not by default transmit ColdFusion's session management cookies back to the server. Therefore, each page read by a bot typically instantiates a new ColdFusion session which can lead to a huge, albeit temporary, drain on system resources particularly in the memory space where session variables reside.

The following code illustrates how to create an autosave feature for your HTML forms:

```
<html>
<head>
<script language="JavaScript">
    function fnPrompt() {
        if (confirm("Save your work?")) {
            document.myform.submit();
        } else {
            setTimeout("fnPrompt()",3000);
        }
    }
</script>
<body onLoad='setTimeout("fnPrompt()",3000) '>
<cform name="myform">
    Enter your name:<br>
    <input type="text" name="name"><br>
    <input type="submit">
</cform>
</body>
</html>
```

Configuring Client Variables

Client variables serve two distinct purposes by ColdFusion developers:

- Store data for an individual user for comparatively long periods of time between sessions
- Synchronize session data between clustered servers by writing client data to a shared database

Client variable settings are administrable through Server Settings > Client Variables. If you are hosting applications that use client variables you should take care to immediately configure ColdFusion to store them in a SQL database. Since there is a fair bit of overhead associated with using client variables, they should be stored in a client-server database like Microsoft SQL Server or ORACLE in order to minimize their performance impact.

In order to wring the best performance from your CF server, you might suggest to your developers that they disable client variables and instead create their own custom variable persistence layer.

Server Settings > Client Variables

Client variables let you store user information and preferences between sessions. The Administrator setting is only used when no ClientStorage attribute is specified in a cfapplication tag. To add a ColdFusion data source to the list of available client storage mechanisms, select the data source from the drop-down list, and click the Add button. To set the data source as the default storage mechanism, select the radio button and Click Apply.

Select Data Source to Add as Client Store

admincf800-solution

Select Default Storage Mechanism for Client Sessions

Actions	Storage Name	Description
<input type="radio"/>	Cookie	Client based text file.
<input checked="" type="radio"/>	Registry	System registry.
<input type="radio"/>	None	

Purge Interval

This option controls how often ColdFusion executes a purge operation on your client stores. If your client stores are configured to be purged, this will be rate at which the operation will be executed. It defaults to 1 hour 7 minutes and should not be less than every 30 minutes.

1 hours 7 minutes

Figure: Configuring Client Variables

Walkthrough 5-3: Configuring for Scalability



In this walkthrough you investigate how to configure ColdFusion production server instance for optimal performance.

Steps

1. Open the ColdFusion Administrator for one of your production instances by browsing to the following URL:
<http://dev.cafetownsend.com/cfide/administrator/>

Configure Performance Settings

2. In the left-side navigation, select **Server Settings > Settings**
3. Turn on the checkbox to **Timeout requests after 10 seconds**
4. Turn on the checkbox to **Enable whitespace management**
5. Turn off the checkbox to **Disable CFC Type Check**
6. Turn on the checkbox to **Enable Global Script Protection**
7. Turn off the checkbox to **Disable access to internal ColdFusion Java components**
8. Click the **Submit Changes** button
9. In the left-side navigation, select **Server Settings > Request Tuning**
10. Set the **Maximum number of simultaneous Template requests** to 8
11. Set the **Timeout requests waiting in queue after 10 seconds**
12. Click the **Submit Changes** button

Configure Caching Options

13. In the left-side navigation, select **Server Settings > Caching**
14. Turn on the setting for **Trusted Cache**
15. Turn on the setting for **Save Class Files**
16. Turn on the checkbox to **Cache Web Server Paths**
17. Click **Submit Changes**
18. Click on **Server Settings > Memory Variables**
19. Leave the settings at default

Configure a Client Variable Repository

20. Open the dev instance administrator by navigating to <http://localhost:8303/cfide/administrator>
21. Click on **Server Settings > Client Variables**
22. Under the label **Select Data Source to Add as Client Store**, select **admincf800-solution** and click **Add**
23. Note the default setting for **Purge data for clients that remain unvisited for 90 days**. Consider how many database rows might be added during this time.
24. Turn on the checkbox to **Disable global client variable updates** in order to minimize database traffic.
25. Click **Submit Changes**
26. Click on the radio button adjacent to **admincf800-solution** and click **Apply**.
27. (optional) Open the **admincf800-solution.mdb** file and note the presence of several new tables for holding client variables.

-- End of Walkthrough --

Configuring settings that affect portability



Typically, ColdFusion applications execute CFML custom tags and refer to files that might be outside the web root. Alternately, ColdFusion a child component might inherit from a parent component in a different folder.

In order to make the application more portable so that it can be easily deployed across different servers with varying drive/directory combinations, ColdFusion supports the following two administrative options:

- Mappings
- Custom Tag Paths

ColdFusion Mappings

Mappings allow the following tags allow the following tags and functions execute code that exists either outside of the web root or in a different directory from the calling page. ColdFusion mappings are separate and distinct from web server mappings.

- `<cfmodule>`
- `<cfinclude>`
- `<cfcomponent [extends]>`
- `createObject()`
- `<cfobject>`
- `<cfinvoke>`

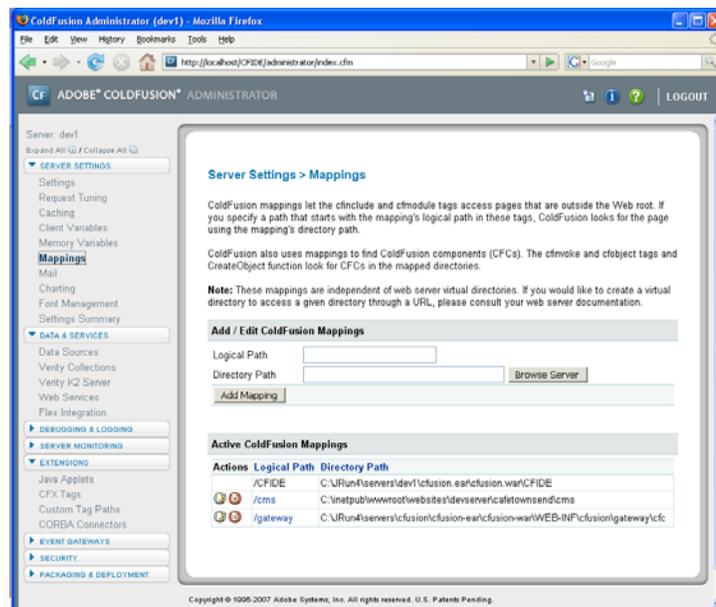


Figure: Registering ColdFusion Mappings

Note that with ColdFusion 8.0, developers can now programmatically configure their own mappings on a per-application basis. In order for this feature to operate, you must have the Enable Per App Settings checkbox turned on in Server Settings > Settings. Notwithstanding the development of this feature, you may still need to register a mapping if the developer has created an Application.cfc file inherits from a parent using the EXTENDS attribute.

Custom Tag Paths

ColdFusion custom tags are .CFM files that are invoked from calling pages using the `<cf_>` syntax. For example, in order to execute a page named `foo.cfm`, the syntax would be `<cf_foo>`.

When using this syntax, ColdFusion will look for the file in the following locations:

- The current directory
- Directories specified by the developer using per app settings
- Directories registered with the ColdFusion Administrator (depicted below)

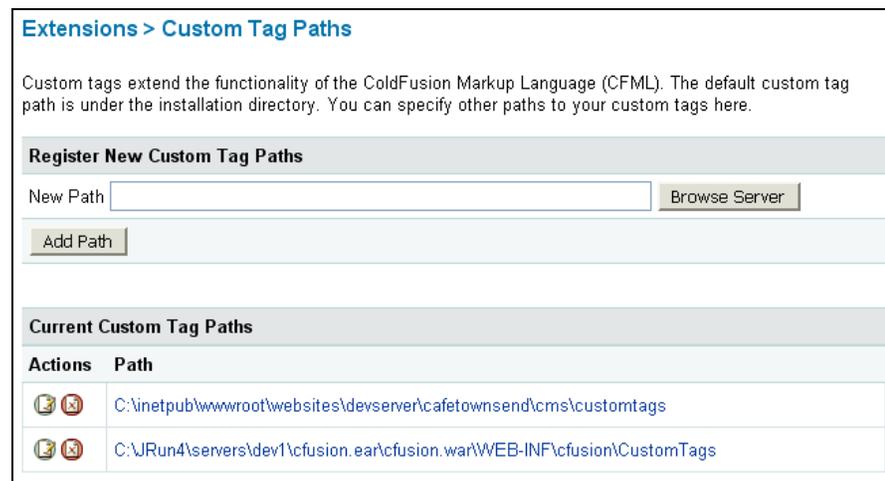


Figure: Registering Custom Tag Paths in the CF Administrator

Note: Using `<cf_>` syntax requires less overhead than using `<cfmodule>` in ColdFusion

Walkthrough 5-4: Configuring Mappings & Tags



In this walkthrough you register CF server mappings and custom tag paths to support the Café Townsend content management system in development.

Steps

1. Open the ColdFusion Administrator by browsing to the following URL: <http://dev.cafetownsend.com/cfide/administrator/>.

Add CF Server Mappings

2. Select **Server Settings > Mappings**
3. Enter a logical path of **/cms**
4. Click on the Browse Server button and select a path of **c:\websites\devserver\cafetownsend\cms**
5. Click **Add Mapping**
6. Enter a logical path of **/cafetownsend**
7. Click on the Browse Server button and select a path of **c:\websites\devserver\cafetownsend**
8. Enter a logical path of **/**
9. Click on the Browse server button and select a path of **c:\websites\devserver\cafetownsend**
10. Click **Add Mapping**

Add Web Server Mappings

11. Open the IIS Manager and add a new virtual directory with the following attributes:

alias: : **cms**
path: : **c:\websites\devserver\cafetownsend\cms**
permissions: **Read/Run Scripts**

Add custom tag paths

12. Select **Extensions > Custom Tag Paths**
13. Click on the Browse Server button and select a path of
c:\websites\devserver\cafetownsend\cms\customtags
14. Click **Add Path**
15. Launch the Café Townsend website by navigating to the following url:

<http://dev.cafetownsend.com/index.cfm>

-- End of Walkthrough --

Defining and Maintaining Verity Collections



ColdFusion 8 allows you to index disk and database-based content into full-text search collections. Verity search enables your developers to quickly perform wildcard, stem, and proximity searches at speeds that exceed the native capabilities of most relational databases.

Verity K2 is organized around a client-server model whereby multiple ColdFusion servers can communicate with a single verity host. This host can be deployed on a ColdFusion server or on its own separate hardware.

Indexed content is stored in a series of directories on disk referred to as collections. As incremental changes are made to collections over a period of time, the performance of a collection degrades to the point where re-indexing is required.

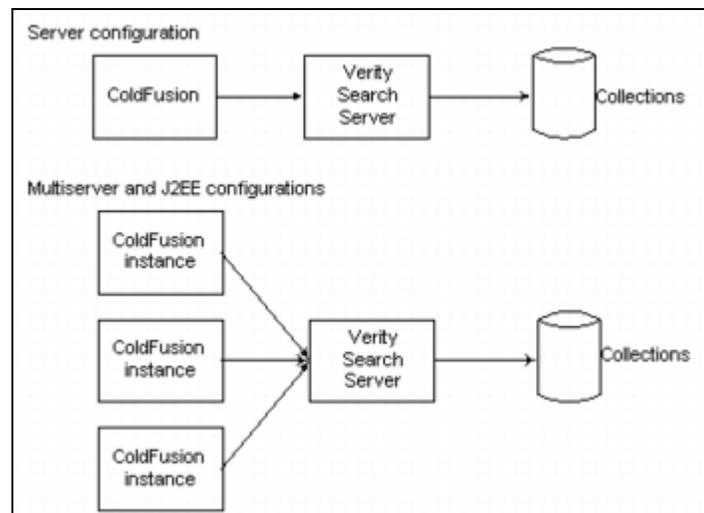


Figure: Verity's organizational architecture

Verity OEM Restrictions

- ColdFusion can only interact with one verity server at a time
- Your indexes are restricted based on document count
 - CF Developer Edition: 10K documents
 - CF Standard Edition: 125K documents
 - CF Enterprise Edition: 250K documents
- You can create a maximum of 128 collections
- The Verity Spider will index local documents only
Note that you can deploy a fully licensed version of Verity K2 to bypass these limitations.

Defining a Verity Collection

Verity collections can be defined by either a developer using the `<cfcollection>` tag or through the ColdFusion Administrator, as depicted below:

Data & Services > Verity Collections

The Verity indexing engine allows you to easily develop search capabilities for your ColdFusion applications. A Verity collection is a group of information to create and manage your Verity collections.

Add New Verity Collections

Name

Path

Language

Enable Category Support

Verity Collections

Actions	Name	Documents	Size (Kb)	Last Modified	Language	Categories	Path
	bookclub	0	28	Oct 5 2007 12:00 PM	english	Yes	C:\Run4\servers\cfusion\cfusion-ea
	figleafdotcom.207	207	4,162	Aug 13 2008 11:45 AM	english	No	d:\verity-figleaf\figleafdotcom

Figure: Administering Verity Collections

Note that when working with large data sets, Verity performs more efficiently if you spread your content among multiple collections. For instance, you could create one collection to index disk file based information and another collection to store data indexed from a relational database.

Verity Maintenance Options

The four Action icons in the Verity administration page perform the following functions:

- **Index**
Use this option only if your verity collection is configured to index a specific directory structure on the server's disk
- **Optimize**
Optimizes a collection after its efficiency has been compromised through multiple updates
- **Purge**
Empties a collection
- **Delete**
Destroys a collection

Typically, developers who use Verity within their applications create a web-based utility page to reindex and maintain collections. Depending on how you have secured your CF server instance, however, you might be responsible for the initial creation of the collection.

Configuring Remote Instances to use Verity K2

As indicated below, you can configure your ColdFusion server to act as a client for a remotely hosted Verity K2 server.

Data & Services > Verity K2 Server

You can install and configure the Verity K2 search service on a host other than the one on which ColdFusion is running. In this case, you can configure the host that ColdFusion uses when performing search operations. If you have purchased the Verity K2 Enterprise product, you may need to configure the aliases, ports and login information of the services that ColdFusion uses. Click the Advanced button to configure these values. You should not need to change the advanced values if you are running with the ColdFusion-installed version of Verity.

Configure Verity K2 Server	
Verity Host Name	<input type="text" value="localhost"/>
K2 Admin Alias	<input type="text" value="ColdFusionK2"/>
K2 Admin Port	<input type="text" value="9951"/>
K2 Server Alias	<input type="text" value="ColdFusionK2_server1"/>
K2 Server Port	<input type="text" value="9921"/>
K2 Index Alias	<input type="text" value="ColdFusionK2_indexserver1"/>
K2 Index Port	<input type="text" value="9961"/>
<small>By default, the K2 Enterprise product requires authentication for the K2 Administration server. You will need to enter a user name and password below if you have installed the K2 product so that it requires authentication.</small>	
K2 Admin User Name	<input type="text"/>
K2 Admin Password	<input type="password"/>
<input type="button" value="Hide Advanced Settings"/>	

Figure: Registering an external K2 server instance

Walkthrough 5-5: Creating a Verity Collection



In this walkthrough you create a series of Verity collections to index disk and database content.

Steps

1. Open the ColdFusion Administrator for your development instance by browsing to the following URL:

<http://dev.cafetownsend.com/cfide/administrator>

2. Click on **Data & Services > Verity Collections**
3. Enter a collection name of **admincf800_dev_articles** and click **Create Collection**
4. Enter a collection name of **admincf800_dev_pages** and click **Create Collection**
5. Click on the **Index collection** icon button next to **admincf800_dev_pages**
6. Click on the **Browse Server** button
7. Select **c:\websites\devserver\cafetownsend\site**
8. Turn on the checkbox to recursively index subdirectories
9. Enter a return URL of **/**
10. Click the Submit button

-- End of Walkthrough --

Patching your system



ColdFusion ships with database drivers from DataDirect. These drivers undergo frequent revisions and new versions become available from Adobe in the form of hot fixes.

Adobe releases hot fixes and updaters for ColdFusion on a regular basis. You can determine which patches, hot fixes, and JVM are installed by accessing the System Information page within the ColdFusion Administrator.

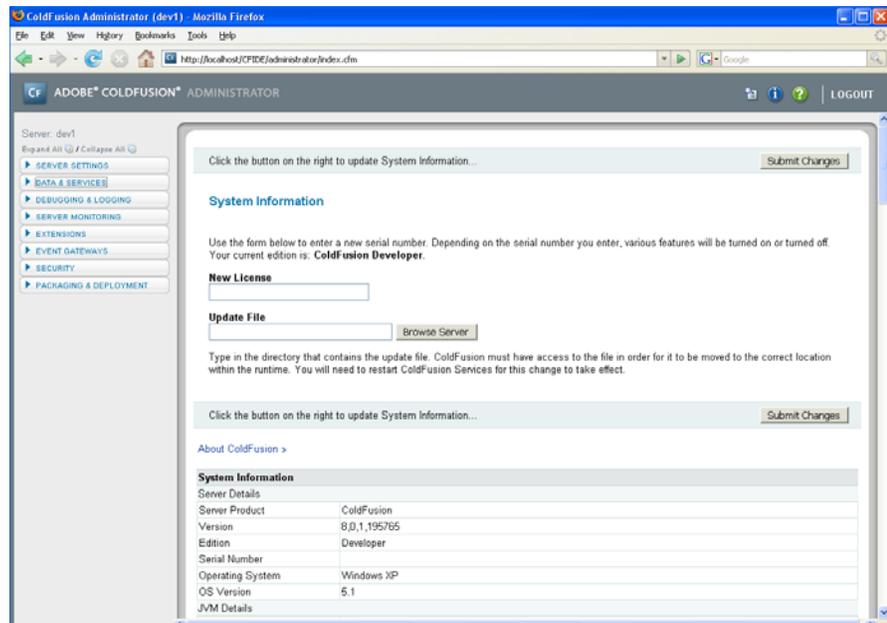


Figure: The ColdFusion Administrator System Information Page

Walkthrough 5-6: Reviewing Configuration



In this walkthrough you investigate how to configure the ColdFusion server for optimal performance.

Steps

1. Open the ColdFusion Administrator by browsing to the following URL: <http://dev.cafetownsend.com/cfide/administrator/>.
2. Click on the  button
3. Review the results with your instructor. In particular, take notice of the many JAR files used by ColdFusion 8.
4. Download and install a hotfix from Adobe.com (optional)

-- End of Walkthrough --

Enabling Interactive Debugging for Developers



ColdFusion 8 allows developers using the CF Eclipse integrated development environment to interactively step through their code in order to diagnose and troubleshoot runtime errors. In order to configure your ColdFusion instance to use this feature you must manually edit the `jvm.config` file for your specific CF instance as referenced in the screenshot below:

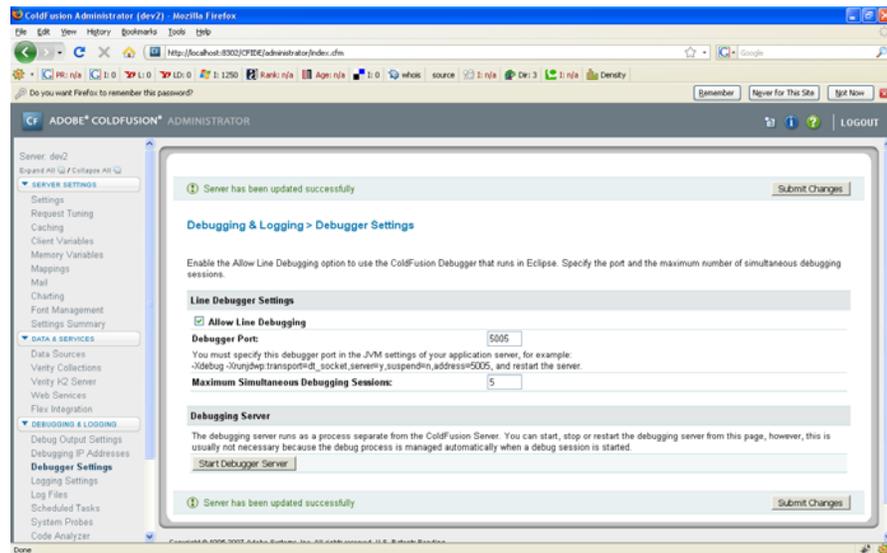


Figure: The ColdFusion Administrator Debugger Settings Page

Modifying your JVM Arguments

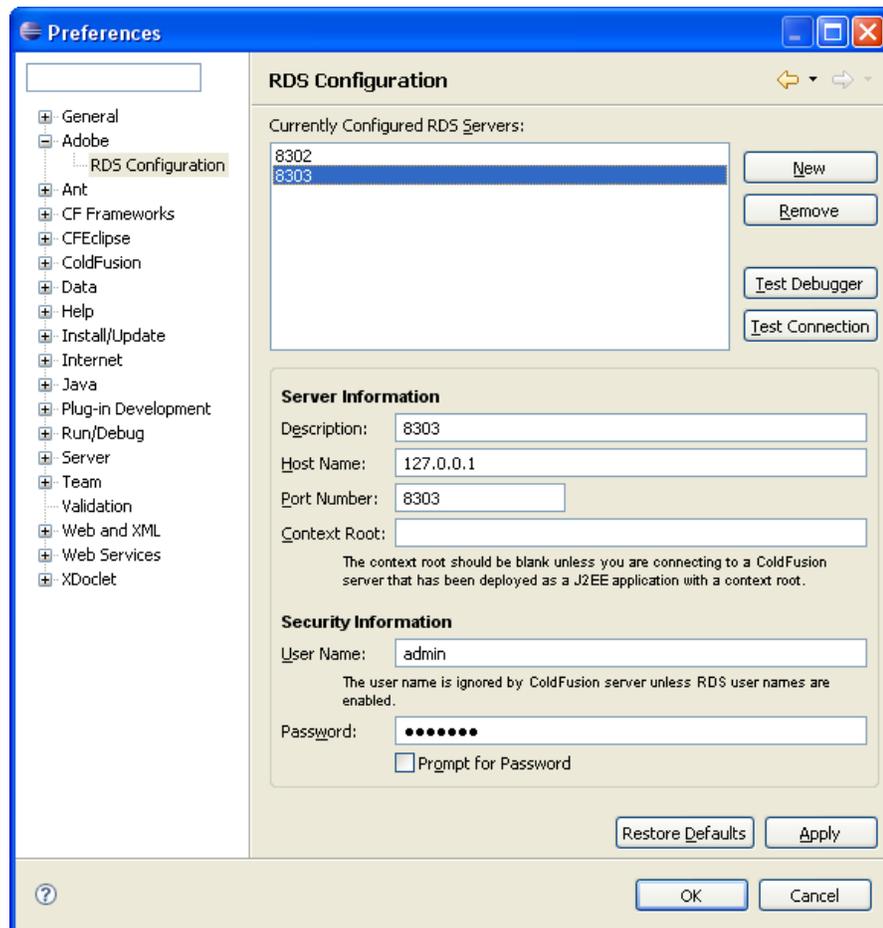
In order to enable interactive debugging you must add the following syntax to the `java.args` line of your `jvm.config`:

```
-xdebug
-Xrunjdwp:transport=dt_socket,server=y,suspend=n,address=5005
```

Where 5005 is the port number that you referenced in the ColdFusion Administrator for your instance. Note that the port must also be unblocked by whatever firewall or port restriction mechanism you might have in place.

Testing Interactive Debugging in Eclipse

In order for debugging to function properly, RDS must also be enabled for the designated instance. In CF Eclipse you configure a connection to the server through the Adobe > RDS Configuration panel as indicated below in the screenshot. Clicking on the Test Debugger connection verifies that the debugger server process is functioning correctly.



Walkthrough 5-7: Configuring for Debugging



In this walkthrough you reconfigure the development instance to use a custom JVM config file as well as enable interactive debugging for the instance via CFEclipse.

Steps

1. Using Windows Explorer, create a copy of `c:\jrun4\bin\jvm.config` named `c:\jrun4\bin\jvm.config_dev1`
2. Open a windows command prompt and type the following commands:

```
cd C:\JRun4\bin

jrunsvc -remove "Adobe ColdFusion 8 AS dev1"

jrunsvc -install dev1 "Adobe ColdFusion 8 as Dev1" "Adobe
Coldfusion 8 as Dev1" -config jvm.config_dev1
```

3. Close the command window

Enable Debugging in the CF Administrator

4. Open a web browser and go to the dev instance CF administrator at <http://127.0.0.1:8301/cfide/administrator>
5. Click on **Debugging & Logging > Debugger Settings**
6. Turn on the checkbox to **allow line debugging**
7. Copy the `jvm.args` parameters listed on the screen in the debugger port section of the page to the windows clipboard
8. Click **submit changes**
9. Using notepad, edit `c:\jrun4\bin\jvm.config_dev1`
10. Paste (append) the text in your clipboard to the `java.args` line in the `jvm.config_dev1` file
11. Save the changes to `jvm.config_dev1`

Test debugging with CFEclipse

12. Open CFEclipse
13. Select **Window > Preferences**
14. In the left panel, select **Adobe > RDS Configuration**
15. In the RDS Configuration panel, click on the **New** button

16. Enter the following information:

Description: **8301**
Host Name: **127.0.0.1**
Port Number: **8301**
User Name: **admin**
Password: **password**

17. Go to the windows control panel and restart the dev1 instance
18. Click on the Test Debugger button.
19. Enter an RDS password of password. It should report back that it was able to connect to the debugger process successfully.

-- End of Walkthrough --

Unit Summary



- ColdFusion is a Java Application running on top of a Java Application Server
- ColdFusion server stability is largely dependent on the JVM memory configuration and garbage collection algorithm
- You can allocate up to 1.2GB to a CF instance on a Win32 box
- You can allocate more than 1.2GB on 64-bit systems, however, 2GB is the practical maximum due to the overhead associated with garbage collection
- There are many JVM memory profilers available, including the CF 8 Server Monitor
- ColdFusion Datasources are integral to any ColdFusion application and can be tweaked for performance
- Setting the ColdFusion Template Cache to a large value can lead to behavior that mimics a memory leak
- CF Administrator settings can lead to performance gains of 30% or more
- Keep your system patched to help ensure secure operations

Review Questions



1. Assuming that you had 3 GB available on a Windows server, what would be the best settings for Xmx and Xms parameters to the JVM and why?
2. Name two garbage collection algorithms and explain when they should be used.
3. Of what benefit is the ColdFusion template cache? What drawbacks exist for using the cache?
4. What is the best value for the number of simultaneous CF template requests given a server with a single, dual-core CPU?