HTTP Connectors Load Balancing Guide

HTTP load-balancing for JBoss Enterprise Application Platform
Edition 5.2.0
Abstract

Read this guide to install and configure the supported HTTP connectors for use with JBoss Enterprise Application Platform and JBoss Enterprise Web Server. This guide covers the Apache Tomcat Connector (mod_jk), JBoss HTTP Connector (mod_cluster), Internet Server API (ISAPI) and Netscape Server API (NSAPI), and discusses clustering and load-balancing with regard to each.
Table of Contents

PREFACE .................................................................................................................. 3
1. FILE NAME CONVENTIONS ............................................................................. 3

PART I. APACHE TOMCAT CONNECTOR (MOD_JK) ........................................... 4

CHAPTER 1. OVERVIEW ...................................................................................... 5

CHAPTER 2. DOWNLOAD AND INSTALL ............................................................... 6

CHAPTER 3. CONFIGURE LOAD BALANCING USING APACHE HTTP SERVER AND MOD_JK .......................... 7
  3.1. CONFIGURE WORKER NODES IN MOD_JK .............................................. 8
  3.2. CONFIGURING JBOSS TO WORK WITH MOD_JK ................................... 9

CHAPTER 4. TROUBLESHOOTING AND OPTIMIZING MOD_JK ......................... 11
  4.1. COMMON PROBLEMS .............................................................................. 11
  4.2. GENERAL DIAGNOSTICS ..................................................................... 17
  4.3. GETTING FURTHER HELP ..................................................................... 18

PART II. JBOSS HTTP CONNECTOR (MOD_CLUSTER) ....................................... 19

CHAPTER 5. OVERVIEW ..................................................................................... 20
  5.1. KEY FEATURES ...................................................................................... 20
  5.2. COMPONENTS ......................................................................................... 20
  5.3. LIMITATIONS ......................................................................................... 21

CHAPTER 6. INSTALL PROXY SERVER COMPONENTS ...................................... 23
  6.1. APACHE HTTP SERVER MODULES ....................................................... 23
    6.1.1. mod_manager.so ............................................................................. 23
    6.1.2. mod_proxy_cluster.so ................................................................... 25
    6.1.3. mod_advertise.so .......................................................................... 26
  6.2. INSTALL PROXY SERVER COMPONENTS ............................................. 27

CHAPTER 7. CONFIGURE BASIC PROXY SERVER ........................................... 29
  7.1. BASIC PROXY CONFIGURATION OVERVIEW ....................................... 29
  7.2. CONFIGURE A LOAD-BALANCING PROXY USING THE HTTP CONNECTOR ........................................ 29

CHAPTER 8. INSTALL NODE WITH BASIC CONFIGURATION ......................... 32
  8.1. WORKER NODE REQUIREMENTS .......................................................... 32
  8.2. INSTALL AND CONFIGURE A WORKER NODE ................................... 32

CHAPTER 9. ADVANCED CONFIGURATION ..................................................... 36
  9.1. STATIC PROXY CONFIGURATION ......................................................... 36
  9.2. CLUSTERED NODE OPERATION ............................................................. 37

CHAPTER 10. JAVA PROPERTIES .................................................................... 39
  10.1. CONFIGURATION PROPERTIES ......................................................... 39
    10.1.1. Proxy Discovery Configuration ...................................................... 39
    10.1.2. Proxy Configuration .................................................................... 40
    10.1.3. SSL Configuration ...................................................................... 42
    10.1.4. HA Configuration ...................................................................... 44
    10.1.5. Load Configuration .................................................................... 44

CHAPTER 11. LOAD METRICS ..................................................................... 45
  11.1. SERVER-SIDE LOAD METRICS ......................................................... 45
  11.2. WEB CONTAINER METRICS ............................................................... 46
1. FILE NAME CONVENTIONS

The following naming conventions are used in file paths for readability. Each convention is styled so that it stands out from the rest of text:

**JBOSS_EAP_DIST**

The installation root of the JBoss Enterprise Application Platform instance. This folder contains the main folders that comprise the server such as `/jboss-as`, `/seam`, and `/resteasy`.

**JBOSS_EWP_DIST**

The installation root of the JBoss Enterprise Web Platform instance. This folder contains the main folders that comprise the server such as `/jboss-as-web`, `/seam`, and `/resteasy`.

**JBOSS_EWS_DIST**

The installation root of the JBoss Enterprise Web Server instance. This folder contains the main folders that comprise the server such as `/extras`, `/httpd`, and the `/tomcat6` folders.

**NATIVE**

The installation root of the JBoss Native zip, extracted to the same directory level as `JBOSS_EAP_DIST`.

**SJWS**

The installation root of the Sun Java Web Server instance. The default file locations for this naming convention are:

- for Solaris 10 x86 or SPARC 64: `/opt/SUNWwbsrv70` /HTTPD_DIST

**HTTPD_DIST**

The installation root of the Apache HTTP Server. This folder contains the main folders that comprise the server such as `/conf`, `/webapps`, and `/bin`. The JBoss Enterprise Web Server `JBOSS_EWS_DIST` directory contains the root installation of `HTTPD_DIST`.

**PROFILE**

The name of the server profile you use as part of your testing or production configuration. The server profiles reside in `JBOSS_EAP_DIST/jboss-as/server` or `JBOSS_EWP_DIST/jboss-as-web/server`.
PART I. APACHE TOMCAT CONNECTOR (MOD_JK)
CHAPTER 1. OVERVIEW

Apache HTTP Server ("Apache") is a well-known web server which can be extended using plug-ins. The Apache Tomcat Connector mod_jk is a plug-in designed to allow request forwarding from Apache HTTP Server to a servlet container. The module also supports load-balancing HTTP calls to a set of servlet containers while maintaining sticky sessions.

HTTP session replication is used to replicate the state associated with web client sessions to other nodes in a cluster. If one node becomes unavailable, another node in the cluster continues to service the failed node's requests. This involves two distinct operations:

- Session state replication
- Load-balancing HTTP Requests

Session state replication is handled by JBoss per application, providing the application is configured to make use of this feature (refer to Section 17.1, “Enabling session replication in your application”).

Load balancing must be handled externally to JBoss, via either hardware or software. A cost-effective way of enabling load balancing is to set up a software load balancer using Apache HTTP Server and the Apache Tomcat Connector (mod_jk).
CHAPTER 2. DOWNLOAD AND INSTALL

Apache HTTP Server is included in the JBoss Enterprise Web Server binary you download from https://access.redhat.com.

Apache mod_jk is included in the native installation binaries for JBoss Enterprise Application Platform and JBoss Enterprise Web Server.

Follow the procedures in the JBoss Enterprise Application Platform or JBoss Enterprise Web Server Installation Guide to download and install the correct platform and native binaries.
CHAPTER 3. CONFIGURE LOAD BALANCING USING APACHE HTTP SERVER AND MOD_JK

Follow the tasks in this chapter to correctly configure load balancing using Apache HTTP Server and the mod_jk connector.

Task: Configure Apache HTTP Server to Load mod_jk

Complete this task to configure Apache HTTP Server to load mod_jk.

Prerequisites

- Apache HTTP Server and mod_jk installed (Refer to Chapter 2, Download and install).

1. Open \texttt{HTTPD\_DIST/conf/httpd.conf} and add the following text at the end of the file.

   ```
   # Include mod_jk's specific configuration file
   Include conf/mod-jk.conf
   ```

2. Create a new file named \texttt{HTTPD\_DIST/conf/mod-jk.conf}

3. Add the following configuration block to \texttt{mod-jk.conf}.

   ```
   # Load mod_jk module
   LoadModule jk_module modules/mod_jk.so

   # Where to find workers.properties
   JkWorkersFile conf/workers.properties

   # Where to put jk logs
   JkLogFile logs/mod_jk.log

   # Set the jk log level [debug/error/info]
   JkLogLevel info

   # Select the log format
   JkLogStampFormat "[%a %b %d %H:%M:%S %Y]"

   # JkOptions indicates to send SSK KEY SIZE
   JkOptions +ForwardKeySize +ForwardURICompat -ForwardDirectories

   # JkRequestLogFormat
   JkRequestLogFormat "%w %V %T"

   # Mount your applications
   JkMount /application/* loadbalancer

   # Add shared memory.
   # This directive is present with 1.2.10 and
   # later versions of mod_jk, and is needed for
   # for load balancing to work properly
   JkShmFile logs/jk.shm
   ```
Confirm that the `LoadModule` directive references the right path for the `mod_jk` library. If not, edit the path.

The default configuration specifies that static content is served directly by Apache HTTP Server and all requests with URL path `/application/*` are sent to the load balancer. If `mod_jk` is only to be used as a load balancer, change the directive to `/*`.

Optional: JkMountFile Directive

In addition to the `JkMount` directive, you can use the `JkMountFile` directive to specify a mount point’s configuration file. The configuration file contains multiple Tomcat forwarding URL mappings.

a. Navigate to `HTTPD_DIST/conf`.

b. Create a file named `uriworkermap.properties`.

c. Specify the URL whose requests are to be forwarded and the name of the worker node to which they are to be forwarded, using the following syntax example as a guide.

The example block will configure `mod_jk` to forward requests to `/jmx-console` and `/web-console` to Apache HTTP Server.

The syntax required takes the form `/url=worker_name`.

d. In `HTTPD_DIST/conf/mod-jk.conf`, append the following directive.

```bash
JkMountFile conf/uriworkermap.properties
```

### 3.1. CONFIGURE WORKER NODES IN MOD_JK

**Task: Configure mod_jk Worker Nodes**

Complete this task to configure two `mod_jk` worker node definitions in a weighted round-robin configuration with sticky sessions active between two servlet containers.
Prerequisites

Understand the format of the workers.properties directives, as specified in Appendix A, Reference: workers.properties.

1. Navigate to HTTPD_DIST/conf/.
2. Create a file named workers.properties.
3. Append the following information to workers.properties.

```
# Define list of workers that will be used
# for mapping requests
worker.list=loadbalancer,status

# Define Node1
# modify the host as your host IP or DNS name.
worker.node1.port=8009
worker.node1.host=node1.mydomain.com
worker.node1.type=ajp13
worker.node1.ping_mode=A
worker.node1.lbfactor=1

# Define Node2
# modify the host as your host IP or DNS name.
worker.node2.port=8009
worker.node2.host=node2.mydomain.com
worker.node2.type=ajp13
worker.node2.ping_mode=A
worker.node2.lbfactor=1

# Load-balancing behavior
worker.loadbalancer.type=lb
worker.loadbalancer.balance_workers=node1,node2
worker.loadbalancer.sticky_session=1

# Status worker for managing load balancer
worker.status.type=status
```

3.2. CONFIGURING JBOSS TO WORK WITH MOD_JK

Task: Configure JBoss Enterprise Application Platform to Operate Using mod_jk

Complete this task to correctly prepare a JBoss Enterprise Application Platform instance on a clustered node to receive forwarded requests from the mod_jk load balancer.

Repeat this task for each server instance you require, observing the warnings at each step.

Prerequisites

- Complete Task: Configure mod_jk Worker Nodes.

1. Navigate to the location of the clustered server instance.
2. Open `JBoss_EAP_DIST/jboss-as/server/PROFILE/deploy/jbossweb.sar/server.xml`.

3. Specify the node name by appending the `jvmRoute` attribute to the `<Engine>` element in `server.xml`. The `jvmRoute` attribute value is the node name defined in `HTTPD_DIST/conf/workers.properties`.

   ```xml
   <Engine name="jboss.web" defaultHost="localhost" jvmRoute="node1"> <!--Preceeding syntax removed for readability -->
   <Connector protocol="AJP/1.3" port="8009" address="${jboss.bind.address}" redirectPort="8443" /> <!--Proceeding syntax removed for readability -->
   </Engine>
   
   IMPORTANT
   If you intend to configure more than one server node in a cluster, ensure you change the `jvmRoute` attribute value to a unique name each time you repeat this step.

4. In `server.xml`, ensure the AJP protocol `<connector>` element is enabled (uncommented). The element is uncommented by default in new installations.

5. You now have a correctly configured Apache HTTP Server with `mod_jk` load balancer, which balances calls to the servlet containers in the cluster, and ensures clients will always use the same servlet container (sticky sessions).

   NOTE
CHAPTER 4. TROUBLESHOOTING AND OPTIMIZING MOD_JK

While optimizing the configuration in Apache HTTP Server, mod_jk, mod_proxy, mod_cluster and JBoss Enterprise Application Platform typically resolves any and all problems and errors in load balancing, there are exceptions (such as long running servlets that require additional optimization).

In most cases, a correctly tuned configuration is the catch all for mod_jk issues. This section discusses some problems and how the configuration can be improved to avoid them.

Optimization Considerations

- Ensure you are on the latest supported component versions.
- Ensure the relevant configurations are tuned correctly. The Red Hat Global Support Services staff can use interactive tools to assist you with tailored configuration settings. Find the appropriate contact details at https://access.redhat.com/support/.

If optimizing the configuration does not resolve the issue the problem is most likely on the JBoss/JVM side. Refer to Procedure 4.5, “JBoss/JVM Problems” for advice about these issues.

4.1. COMMON PROBLEMS

The list below outlines some common configuration problems. Ensuring your implementation is not subject to one of these may assist to resolve your issue.

Specific errors and general performance issues are discussed later in this section.

Common Configuration Issues

**JkShmFile on a NFS share**

Placing the JkShmFile on a NFS share can cause unexplained pauses in mod_jk and odd behavior. It is strongly recommended that the JkShmFile always be placed on local storage to avoid problems.

**A firewall between Apache HTTP Server and JBoss Enterprise Application Platform**

If there is a firewall between Apache HTTP Server and JBoss Enterprise Application Platform and no socket_keepalive parameter is set, the firewall can close connections unexpectedly.

**MaxClients higher than maxThreads**

Setting the MaxClients parameter in Apache HTTP Server higher than the maxThreads setting in JBoss (with a high load on the server) will result in Apache HTTP Server overwhelming the JBoss instance with threads which will cause hung and/or dropped connections.

**No connectionTimeout parameter set**

The connectionTimeout parameter set in JBoss is required for proper maintenance of old connections.

**No CPing/CPong set**

The CPing/CPong property in mod_jk is the most important worker property setting, allowing mod_jk to test and detect faulty connections. Not setting this parameter can lead to bad connections not being detected as quickly which can lead to web requests behaving as if 'hung'.
Running an old version of mod_jk.

There are known issues with sticky sessions in versions prior to mod_jk 1.2.27.

Running an older version of EAP.

There is a bug in EAP 4.2 base and EAP 4.2 CP01 which causes sockets to be left in the CLOSE_WAIT state, thus causing the appearance of hung requests again. This issue has been reported and fixed https://jira.jboss.org/jira/browse/JBPAPP-366

Unresponsive back end server

`java.lang.OutOfMemoryError` errors or high pause times can cause the back end server to become unresponsive.

All of the problems listed above are typically resolved after optimizing the configuration in Apache HTTP Server, mod_jk, and JBoss.

Common Errors

"CPing/CPong" Errors

Presents with errors like the following:

```plaintext
[info] ajp_handle_cping_cpong::jk_ajp_common.c (865): timeout in reply cpong
...
[error] ajp_connect_to_endpoint::jk_ajp_common.c (957): (nodeA) cping/cpong after connecting to the backend server failed (errno=110)
[error] ajp_send_request::jk_ajp_common.c (1507): (nodeA) connecting to backend failed. Tomcat is probably not started or is listening on the wrong port (errno=110)
```

These CPing/CPong messages do not indicate a problem with mod_jk at all, they indicate that JBoss did not respond in the defined CPing/CPong time.

This is seen many times when there is high load on the JVM JBoss is running on causing high garbage collection or potentially thread contention. It could also be that the JBoss instance is overloading, or even that a firewall is blocking the connection or there are network issues.

The following workflow may assist to correct these type of issues:

Procedure 4.1. Resolving "CPing/CPong" Errors

1. Optimize your Apache HTTP Server and JBoss Enterprise Application Platform configuration. You can contact Red Hat's Global Support Services for assistance with this.

   If this does not resolve the issue, proceed to Step 2

2. Confirm that there is no firewall blocking or dropping the AJP connections.

"Tomcat is down" Errors

Presents with errors like the following:
The above error means that JBoss did not respond in the configured reply_timeout time. The solution can be one (or both) of the following:

1. Increase the reply_timeout.

2. Verify there are no garbage collection issues/long pause times in JBoss that may prevent the request from responding thus causing that error.

The above error likely means that JBoss Enterprise Application Platform did not respond within the configured core Apache HTTP Server timeout period. Note that with these messages the [11159:3086420192] portion of the message serves as an identifier for the connection/request in question. Therefore tracing back from the point of the error in logs can help clarify the activity around the connection/request that lead to the error.

In this case, that helps clarify that the error was experienced five minutes after the response was sent to JBoss, which likely points to a five minute timeout (this is Apache HTTP Server's Timeout directive default if not specified). If the Timeout is interrupting mod_jk requests, then it should be increased from the current value to allow for the maximum acceptable response time.

Procedure 4.2. Resolving "Tomcat is down" Errors

1. Optimize your Apache HTTP Server and JBoss Enterprise Application Platform configuration. You can contact Red Hat's Global Support Services for assistance with this.

   If this does not resolve the issue, proceed to Step 2

2. Confirm that there is no firewall blocking or dropping the AJP connections.
General Performance Issues

Presents with errors like the following:

```
Invalid message received with signature 12336
```

The above exception when using mod_jk in JBoss Web typically indicates a non AJP request sent to the AJP connector.

Workflows that may assist in resolving these kinds of issues is below:

**Procedure 4.3. General Performance Problems**

1. Optimize your Apache HTTP Server and JBoss Enterprise Application Platform configuration. You can contact Red Hat’s Global Support Services for assistance with this.

    If this does not resolve the issue, proceed to Step 2

2. Gather garbage collection logs for analysis.

    If the logs show long garbage collection pause times then you should optimize the Java Virtual Machine to reduce the garbage collection pauses and gather/recheck updated logs. Refer to https://access.redhat.com/knowledge/solutions/19932 (Red Hat account required) for more information.

    If this is not the case, or did not resolve the issue, try Step 3, Step 4 and/or Step 5 until your issue is resolved.

3. Determine how long the longest request should take. Factor in transaction times. You may need to increase the reply_timeout to resolve the problem.

    If this does not resolve the issue, continue to Step 4.

4. Determine if your current environment can handle the given load. If not, you may need to upgrade or add more machines.

    If this does not resolve the issue, continue to Step 5.

5. Confirm that there is no firewall blocking or dropping the AJP connections.

**Procedure 4.4. 503 Errors**

1. Optimize your Apache HTTP Server and JBoss Enterprise Application Platform configuration. You can contact Red Hat’s Global Support Services for assistance with this.

    If this does not resolve the issue, proceed to Step 2

2. Gather garbage collection logs for analysis.

    If the logs show long garbage collection pause times then you should optimize the Java Virtual Machine to reduce the garbage collection pauses and gather/recheck updated logs. Refer to https://access.redhat.com/knowledge/solutions/19932 (Red Hat account required) for more information.

    If this is not the case, or does not resolve the issue, continue to Step 3
3. Determine how long the longest request should take. Factor in transaction times. You may need to increase the `reply_timeout` to resolve the issue.

If this does not resolve the issue, move on to **Step 4**.

4. Determine if your current environment can handle the given load. If not, you may need to upgrade or add more machines.

**JBoss/JVM-related Issues**

May present with errors like:

```
[error] service::jk_lb_worker.c (1473): All tomcat instances failed, no more workers left
```

If Apache HTTP Server and JBoss Enterprise Application Platform are optimized and you still receive "no more workers left" this typically indicates an issue on the JBoss/JVM side. A number of JVM-related problems could lead to mod_jk not being able to get a connection to JBoss in the configured timeouts, thus causing the worker to go into the error state and producing this message.

**Procedure 4.5. JBoss/JVM Problems**

1. Enable garbage collection logging.

   a. For UNIX based systems, the options should be placed in `run.conf`, not `run.sh`. The `run.conf` in the server configuration directory (e.g. `<JBoss_HOME>/server/<PROFILE>/run.conf`) takes precedence over the `run.conf` in the `<JBoss_HOME>/bin` directory (except in JBoss EAP 5.0.0 due to a regression fixed in version 5.0.1).

   ```
   .run.sh -c node1 -b 127.0.0.1 -Djboss.messaging.ServerPeerID=1
   .run.sh -c node2 -b 127.0.0.1 -Djboss.messaging.ServerPeerID=2
   -Djboss.service.binding.set=ports-01
   JAVA_OPTS="$JAVA_OPTS -verbose:gc -Xloggc:gc_node1.log -XX:+PrintGCDetails -XX:+PrintGCDateStamps"
   ```

   d. Then for the gc.log files to be properly separated you will need to make sure each `<PROFILE>` has a unique `run.conf` with the JVM_OPTS specific to that `<PROFILE>`.

   For example node1 will contain a `<JBoss_HOME>/server/node1/run.conf` with contents:

   ```
   JAVA_OPTS="$JAVA_OPTS -verbose:gc -Xloggc:gc_node1.log -XX:+PrintGCDetails -XX:+PrintGCDateStamps"
   ```

   e. And `<JBoss_HOME>/server/node2/run.conf` with contents:
1. For the time period when there are slowdowns, hangs, or errors, gather the following data:

   - High CPU data coupled with thread dumps (depending upon platform):

     The links below can assist in gathering Java thread data. A Red Hat subscription is required.

     - CPU utilization by Java threads on Linux/Solaris: https://access.redhat.com/knowledge/node/46596.
     - CPU utilization by Java threads on Windows: https://access.redhat.com/knowledge/node/46598.

   - For cases where the Java application is an application server, gather log files:

     **In JBoss:**

     - `<JBOSSES_HOME>/server/<PROFILE>/log/server.log`
     - `<JBOSSES_HOME>/server/<PROFILE>/log/boot.log`

     **In Tomcat:**

     - `catalina.out`

2. Determine if the CPU utilization is caused by the JVM (Java application). Here, you want to validate that a Java process is indeed using an unexpected amount of CPU. The Java thread data gathered in the first step should help identify this.

3. Assuming a Java process is identified as the cause of high CPU, the most common cause is Java Garbage collection. Determine if the high CPU is caused by Java garbage collection by running:

   ```
   JAVA_OPTS="-XX:+PrintGCDetails -XX:+PrintGCDateStamps"
   ```

   Alternatively, you may be able to add a timestamp to the file name depending on the OS and/or shell. For example, with OpenJDK or Oracle/Sun JDK on Linux: `-Xloggc:gc.log.`date +%Y%m%d%H%M%S`. On Windows, you can use:

   ```
   for /f "tokens=2-4 delims=/ " %%a in ("%DATE%") do (set mydate=%a%%e%%b)
   for /f "tokens=1-2 delims=/:" %%a in ("%TIME%") do (set mytime=%%a%%b)
   set "JAVA_OPTS=%JAVA_OPTS% -Xloggc:C:/log/gv.log.%mydate%_%mytime%"
   ```

2. For the time period when there are slow downs, hangs, or errors, gather the following data:

   - High CPU data coupled with thread dumps (depending upon platform):

     The links below can assist in gathering Java thread data. A Red Hat subscription is required.

     - CPU utilization by Java threads on Linux/Solaris: https://access.redhat.com/knowledge/node/46596.
     - CPU utilization by Java threads on Windows: https://access.redhat.com/knowledge/node/46598.

   - For cases where the Java application is an application server, gather log files:

     **In JBoss:**

     - `<JBOSSES_HOME>/server/<PROFILE>/log/server.log`
     - `<JBOSSES_HOME>/server/<PROFILE>/log/boot.log`

     **In Tomcat:**

     - `catalina.out`

3. Determine if the CPU utilization is caused by the JVM (Java application). Here, you want to validate that a Java process is indeed using an unexpected amount of CPU. The Java thread data gathered in the first step should help identify this.

4. Assuming a Java process is identified as the cause of high CPU, the most common cause is Java Garbage collection. Determine if the high CPU is caused by Java garbage collection by running:

   ```
   JAVA_OPTS="-XX:+PrintGCDetails -XX:+PrintGCDateStamps"
   ```

   Alternatively, you may be able to add a timestamp to the file name depending on the OS and/or shell. For example, with OpenJDK or Oracle/Sun JDK on Linux: `-Xloggc:gc.log.`date +%Y%m%d%H%M%S`. On Windows, you can use:

   ```
   for /f "tokens=2-4 delims=/ " %%a in ("%DATE%") do (set mydate=%a%%e%%b)
   for /f "tokens=1-2 delims=/:" %%a in ("%TIME%") do (set mytime=%%a%%b)
   set "JAVA_OPTS=%JAVA_OPTS% -Xloggc:C:/log/gv.log.%mydate%_%mytime%"
   ```
analyzing the garbage collection for long pause times and/or low throughput overall at the
time of the issue.

To find the garbage collection logging related to the issue, it is necessary to determine the
number of seconds after JVM startup that the issue happens (that is the typical format of
garbage collection logging timestamps). To determine the time elapsed, you can use the
first timestamp in the high CPU data gathered and the first timestamp in the console log,
boot.log (JBoss), server.log (JBoss), or catalina.out (Tomcat.)

If you see long pause times and/or low throughput overall, refer to the following
Knowledge Base article (Red Hat subscription required)
https://access.redhat.com/knowledge/node/19932.

5. If Garbage collection is not responsible for the high CPU, use the thread dump information
gathered when validating CPU information to identify the threads.

One area that is not a direct consequence of an unoptimized mod_jk configuration but can still cause
issues with mod_jk is JVM and garbage collection related problems. When there are high pause times
and the JVM is not optimized for the app server, the pause times can cause mod_jk issues even when
mod_jk is tuned.

4.2. GENERAL DIAGNOSTICS

1. Verify the back end server is responsive by making a direct request to it.

2. Monitor high load using one of the following methods:
   
   **Twiddle**

   1. Locate the appropriate Twiddle script for your environment (twiddle.sh, twiddle.bat or twiddle.jar) in the `<JBOSS_HOME>/bin/` directory.
   
   2. Run the following command:

   ```
   <TWIDDLE> -u admin -p password get "jboss.web:name=ajp-127.0.0.1-8009,type=ThreadPool"
   ```

   Use the script appropriate to your operating system and environment.

   Twiddle may need to be modified for each specific environment, but the above will work
   in a default JBoss instance where no ports have been changed and JBoss is starting on
   the localhost.

   **JMX Console**


   2. Find the `jboss.web` section.

   3. Click on `name=ajp-localhost/127.0.0.1-8009,type=ThreadPool` (or whichever AJP ThreadPool matches your environment)

   4. Investigate the `currentThreadsBusy` attribute. If this attribute is reaching the `maxThreads` there will be a problem as JBoss Web is reaching the defined ThreadPool capacity.
4.3. GETTING FURTHER HELP

If none of the above information resolves your issue you can contact the Global Support Services staff for assistance.

You can find the appropriate contact details at https://access.redhat.com/support/.

Please gather the following information prior to your call.

- JBoss EAP boot.log.
- Apache HTTP Server's httpd.conf and the httpd-mpm.conf file (if it exists).
- mod_jk's workers.properties.
- mod_jk's mod_jk.conf.
- <JBOSS_HOME>/server/<PROFILE>/deploy/JBOSSWEB/server.xml
- <JBOSS_HOME>/server/<PROFILE>/deploy/JBOSSWEB/META-INF/jboss-service.xml
- The output of running httpd -V on Apache HTTP Server (httpd -V > httpd.out, for example).
  Note the capital "V". A lowercase "v" will not produced the desired output.
- Version of Apache HTTP Server or the JBoss Enterprise Web Server.
- /etc/sysconfig/httpd
PART II. JBOSS HTTP CONNECTOR (MOD_CLUSTER)
CHAPTER 5. OVERVIEW

The JBoss HTTP Connector mod_cluster is a reduced-configuration, intelligent load-balancing solution for JBoss Enterprise Application Platform, based on technology originally developed by the JBoss mod_cluster community project.

The JBoss HTTP connector load-balances HTTP requests to JBoss Enterprise Application Platform and JBoss Enterprise Web Server worker nodes, using Apache HTTP Server as the proxy server.

5.1. KEY FEATURES

Apache HTTP Server-based

The JBoss HTTP Connector mod-cluster uses Apache HTTP Server as the proxy server.

Real-time load-balancing calculation

The JBoss HTTP Connector mod_cluster creates a feedback network between the worker nodes and the proxy server. The mod_cluster service is deployed on each of the worker nodes. This service feeds real-time load information to the proxy server. The proxy server then makes intelligent decisions about where to allocate work, based on the current load on each worker node. This real-time adaptive load distribution results in increased optimization of resources.

The information that is reported by the worker nodes and the load-balancing policy used by the proxy are both customizable.

Routing based on real-time application life-cycle

The JBoss HTTP Connector mod_cluster service deployed on the worker nodes relays application life-cycle events to the proxy server. This allows the server to dynamically update its routing table. When an application is undeployed on a node, the proxy server no longer routes traffic for that application to that node.

Automatic Proxy Discovery

The proxy server can be configured to announce its presence via UDP multicast. New worker nodes discover the proxy server and add themselves to the load-balancing cluster automatically. This greatly reduces the configuration and maintenance needed. When UDP multicast is not available or is undesirable, worker nodes are configured with a static list of proxies.

Multiple Protocol Support

The JBoss HTTP Connector mod_cluster can use HTTP, HTTPS, or Apache JServ Protocol (AJP) for communication between the proxy and the worker nodes.

5.2. COMPONENTS

Proxy Server

On the proxy server, the JBoss HTTP Connector, mod-cluster, consists of four Apache HTTP Server modules.

Shared Memory Manager module: mod_slotmem.so

The Shared Memory Manager module, mod_slotmem, makes the real-time worker node information available to multiple Apache HTTP Server server processes.
Cluster Manager module: mod_manager.so
The Cluster Manager module, mod_manager, receives and acknowledges messages from nodes, including worker node registrations, worker node load data, and worker node application life-cycle events.

Proxy Balancer module: mod_proxy_cluster.so
The Proxy Balancer module, mod_proxy_cluster, handles the routing of requests to cluster nodes. The Proxy Balancer selects the appropriate node to forward the request to, based on application location in the cluster, current state of each of the cluster nodes, and the Session ID (if a request is part of an established session).

Proxy Advertisement module: mod_advertise.so
The Proxy Advertisement module, mod_advertise.so, broadcasts the existence of the proxy server via UDP multicast messages. The server advertisement messages contain the IP address and port number on which the proxy is listening for responses from nodes that wish to join the load-balancing cluster.

NOTE
Refer to Section 6.1, “Apache HTTP Server modules” for detailed information about the available modules including user-configurable parameters.

Worker Node Components
The JBoss HTTP Connector client service, mod-cluster.sar, is deployed on each worker node.

Worker node service: mod-cluster.sar
This service provides the proxy with real-time information on the worker node's state and sends notification of application life-cycle events; as well as allowing the node to discover and register itself with any proxies running on the same network.

5.3. LIMITATIONS
The JBoss HTTP Connector mod_cluster uses shared memory to keep the nodes description, the shared memory is created at the startup of httpd and the structure of each item is fixed. Therefore, when defining proxy server and worker node properties, make sure to follow these character limits:

- Maximum Alias length: 100 characters (Alias corresponds to the network name of the respective virtual host; the name is defined in the Host element)
- Maximum context length: 40 characters (for example, if myapp.war is deployed in /myapp, then /myapp is the context)
- Maximum balancer name length: 40 characters (the balancer property in MBean)
- Maximum JVMRoute string length: 80 character (JVMRoute in the <Engine> element)
- Maximum domain name length: 20 characters (the domain property in MBean)
- Maximum hostname length for a node: 64 characters (hostname address in the <Connector> element)
- Maximum port length for a node: 7 characters (8009 is 4 characters, the port property in the <Connector> element)

- Maximum scheme length for a node: 6 characters (possible values are http, https, ajp, the protocol of the connector)

- Maximum cookie name length: 30 characters (the header cookie name for session ID default value: JSESSIONID from org.apache.catalina.Globals.SESSION_COOKIE_NAME)

- Maximum path name length: 30 characters (the parameter name for the session ID default value: JSESSIONID from org.apache.catalina.Globals.SESSION_PARAMETER_NAME)

- Maximum length of a session ID: 120 characters (session ID resembles the following: BE81FA969BF64C8EC2B6600457EAAAA.node01)
CHAPTER 6. INSTALL PROXY SERVER COMPONENTS

Read this chapter to install the JBoss HTTP Connector, mod_cluster, on a proxy server.

6.1. APACHE HTTP SERVER MODULES

Read this section for expanded definitions of the Apache HTTP Server proxy server modules discussed in Section 5.2, “Components”. You specify these modules as part of Task: Install Proxy Server Components.

6.1.1. mod_manager.so

The Cluster Manager module, mod_manager, receives and acknowledges messages from nodes, including worker node registrations, worker node load data, and worker node application life-cycle events.

```
LoadModule manager_module modules/mod_manager.so
```

You can also define the following related directives in the `<VirtualHost>` element:

**MemManagerFile**

Defines the location for the files in which mod_manager stores configuration details. mod_manager also uses this location to store generated keys for shared memory and lock files. *This must be an absolute path name.*

It is recommended that this path be set explicitly and on a local drive, not an NFS share. The default value is platform/httpd specific.

Valid paths are:

- HP-UX: `HTTPD_HOME/cache/mod_cluster`
- Red Hat Enterprise Linux: `/var/cache/mod_cluster`

**Maxcontext**

The maximum number of contexts JBoss mod_cluster will use. The default value is 100.

**Maxnode**

The maximum number of worker nodes JBoss mod_cluster will use. The default value is 20.

**Maxhost**

The maximum number of hosts (aliases) JBoss mod_cluster will use. This is also the maximum number of load balancers. The default value is 10.

**Maxsessionid**

The maximum number of active session identifiers stored. A session is considered inactive when no information is received from that session within five minutes. The default value is 0, which disables this logic.

**ManagerBalancerName**
The name of the load balancer to use when the worker node does not provide a load balancer name. The default value is mycluster.

PersistSlots
When set to on, nodes, aliases and contexts are persisted in files. The default value is off.

CheckNonce
When set to on, session identifiers are checked to ensure that they are unique, and have not occurred before. The default is on.

WARNING
Setting this directive to off can leave your server vulnerable to replay attacks.

SetHandler
Defines a handler to display information about worker nodes in the cluster. This is defined in the Location element:

```xml
<Location $LOCATION>
    SetHandler mod_cluster-manager
    Order deny,allow
    Deny from all
    Allow from 127.0.0.1
</Location>
```

When accessing the $LOCATION defined in the Location element in your browser, you will see something like the following. (In this case, $LOCATION was also defined as mod_cluster-handler.)
Figure 6.1. mod_cluster Status

*Transferred* corresponds to the POST data sent to the worker node. *Connected* corresponds to the number of requests that had been processed when this status page was requested. *Sessions* corresponds to the number of active sessions. This field is not present when *Maxsessionid* is 0.

### 6.1.2. mod_proxy_cluster.so

The Proxy Balancer module, *mod_proxy_cluster*, handles the routing of requests to cluster nodes. The Proxy Balancer selects the appropriate node to forward the request to, based on application location in the cluster, current state of each of the cluster nodes, and the Session ID (if a request is part of an established session).

```
LoadModule proxy_cluster_module modules/mod_proxy_cluster.so
```

You can also define the following related directives in the `<VirtualHost>` element to change load-balancing behavior.

#### mod_proxy_cluster directives

**CreateBalancers**

Defines how load balancers are created in the Apache HTTP Server virtual hosts. The following values are valid in *CreateBalancers*:

0

Create load balancers in all virtual hosts defined in Apache HTTP Server. Remember to configure the load balancers in the `ProxyPass` directive.

1

Do not create balancers. When using this value, you must also define the load balancer name in the `ProxyPass` or `ProxyPassMatch`.

2

Create only the main server. This is the default value for *CreateBalancers*.
UseAlias

Defines whether to check that the defined Alias corresponds to the ServerName. The following values are valid for UseAlias:

0
Ignore Alias information from worker nodes. This is the default value for UseAlias.

1
Verify that the defined alias corresponds to a worker node's server name.

LBstatusRecalTime

Defines the interval in seconds between the proxy calculating the status of a worker node. The default interval is 5 seconds.

ProxyPass; ProxyPass

ProxyPass maps remote servers into the local server namespace. If the local server has an address http://local.com/, then the following ProxyPass directive would convert a local request for http://local.com/requested/file1 into a proxy request for http://worker.local.com/file1.

ProxyPassMatch; ProxyPassMatch

ProxyPassMatch uses Regular Expressions to match local paths to which the proxied URL should apply.

For either directive, ! indicates that a specified path is local, and a request for that path should not be routed to a remote server. For example, the following directive specifies that .gif files should be served locally.

ProxyPassMatch ^/(.*\.gif)$ !

6.1.3. mod_advertise.so

The Proxy Advertisement module, mod_advertise.so, broadcasts the existence of the proxy server via UDP multicast messages. The server advertisement messages contain the IP address and port number where the proxy is listening for responses from nodes that wish to join the load-balancing cluster.

This module must be defined alongside mod_manager in the VirtualHost element. Its identifier in the following code snippet is advertise_module.

LoadModule advertise_module modules/mod_advertise.so

mod_advertise also takes the following directives:

ServerAdvertise

Enables or disables the advertising mechanism. When set to On, the advertising mechanism is used to tell worker nodes to send status information to this proxy. When set to Off, the advertising mechanism is disabled.
You can also specify a hostname and port with the following syntax: `ServerAdvertise On http://hostname:port/`. This is only required when using a name-based virtual host, or when a virtual host is not defined.

The default value is **off** but it is automatically enabled if any `Advertise` directive is specified in defining a VirtualHost.

**AdvertiseGroup**

Defines the multicast address to advertise on. The syntax is `AdvertiseGroup address:port`, where `address` should correspond to `AdvertiseGroupAddress`, and `port` should correspond to `AdvertisePort` in your worker nodes.

If your worker node is JBoss Enterprise Application Platform-based, and the `-u` switch is used at startup, the default `AdvertiseGroupAddress` is the value passed via the `-u` switch.

The default value is **224.0.1.105:23364**. If `port` is not specified, the default port used is **23364**.

**AdvertiseFrequency**

The interval (in seconds) between multicast messages advertising the IP address and port. The default value is **10**.

**AdvertiseSecurityKey**

Defines a string used to identify the JBoss HTTP Connector mod_cluster in JBoss Web. By default this directive is not set and no information is sent.

**AdvertiseManagerUrl**

Defines the URL that the worker node should use to send information to the proxy server. By default this directive is not set and no information is sent.

**AdvertiseBindAddress**

Defines the address and port over which to send multicast messages. The syntax is `AdvertiseBindAddress address:port`. This allows an address to be specified on machines with multiple IP addresses. The default value is **0.0.0.0:23364**.

### 6.2. INSTALL PROXY SERVER COMPONENTS

**Task: Install Proxy Server Components**

Follow this task to install the JBoss HTTP Connector.

The native components are specific to particular operating system and processor architectures. Refer to the JBoss Enterprise Application Platform Installation Guide to download the correct native components package for your server operating system and processor architecture.

**Prerequisites**

- An installed Web Server distribution, such as JBoss Enterprise Web Server or HPWS installed (designated by HTTPD_HOME in this documentation).
- JBoss Enterprise Application Platform 5 Native components downloaded.
1. **Extract Apache HTTP Server modules from Native Components download**
   Extract the four modules `mod_advertise.so`, `mod_manager.so`, `mod_proxy_cluster.so`, `mod_slotmem.so` from the appropriate Native Components package directory for your processor architecture: either `native/lib/httpd/modules` or `native/lib64/httpd/modules`.

2. **Copy Apache HTTP Server modules to HTTPD_HOME**
   Copy the JBoss HTTP Connector modules to the `HTTPD_HOME/httpd/modules` directory of the JBoss Enterprise Web Server.

3. **Disable the mod_proxy_balancer module**
   Edit the HTTPD configuration file `HTTPD_HOME/httpd/conf/httpd.conf` and mark the following line as a comment by adding a `#` character at the start:

   ```
   LoadModule proxy_balancer_module modules/mod_proxy_balancer.so
   ```

   This module is incompatible with the JBoss HTTP Connector.

4. **Configure the server to load the JBoss HTTP Connector modules**
   a. Create `HTTPD_HOME/httpd/conf/JBoss_HTTP.conf`.

   b. Add the following lines to `HTTPD_HOME/httpd/conf/JBoss_HTTP.conf`:

   ```
   LoadModule slotmem_module HTTPD_HOME/modules/mod_slotmem.so
   LoadModule manager_module HTTPD_HOME/modules/mod_manager.so
   LoadModule proxy_cluster_module HTTPD_HOME/modules/mod_proxy_cluster.so
   LoadModule advertise_module HTTPD_HOME/modules/mod_advertise.so
   ```

5. **Restart the HTTPD service**
   Refer to HTTPD-specific documentation for detailed instructions.
CHAPTER 7. CONFIGURE BASIC PROXY SERVER

Follow the instructions in this chapter to configure an HTTPD to use the JBoss HTTP Connector (mod_cluster).

7.1. BASIC PROXY CONFIGURATION OVERVIEW

Configuration of the proxy server consists of one mandatory and one optional portion:

1. Configure a proxy server listener to receive worker node connection requests and worker node feedback.

2. Optional: Disable server advertisement.

Server Advertisement

The proxy server can advertise itself using UDP multicast. When UDP multicast is available on the network between the proxy server and the worker nodes server advertisement allows you to add worker nodes with no further configuration required on the proxy server, and minimal configuration on the worker nodes.

If UDP multicast is not available or undesirable, configure the worker nodes with a static list of proxy servers, as detailed in Section 9.1, “Static proxy configuration”. There is no need in either case to configure the proxy server with a list of worker nodes.

7.2. CONFIGURE A LOAD-BALANCING PROXY USING THE HTTP CONNECTOR

Read this section to configure a load balancing proxy that uses the JBoss HTTP Connector.

Task: Configure a Proxy Server Listener

Follow this task to configure the Apache HTTP Server service to act as a load-balancing proxy using the JBoss HTTP Connector.

Prerequisites

- An installed Web Server distribution, such as JBoss Enterprise Web Server or HPWS installed (designated by HTTPD_HOME in this documentation).
- JBoss HTTP Connector modules. Refer to Chapter 6, Install proxy server components for details.

1. Create a listen directive for the proxy server

   Edit the configuration file `HTTPD_HOME/httpd/conf/JBoss_HTTP.conf` and add the following:

   ```
   Listen IP_ADDRESS:PORT_NUMBER
   ```

   Where `IP_ADDRESS` is the IP address of a server network interface to communicate with the worker nodes, and `PORT_NUMBER` is the port on that interface to listen on.
NOTE

The port PORT_NUMBER must be open on the server firewall for incoming TCP connections.

Example 7.1. Example Listen Directive

Listen 10.33.144.3:6666

2. Create Virtual Host

Add the following <VirtualHost> block to $HTTPD_HOME/httpd/conf/JBoss_HTTP.conf:

```xml
<VirtualHost IP_ADDRESS:PORT_NUMBER>
    <Location />
        Order deny,allow
        Deny from all
        Allow from PARTIAL_IP_ADDRESS
    </Location>
    KeepAliveTimeout 60
    MaxKeepAliveRequests 0
    ManagerBalancerName mycluster
    AdvertiseFrequency 5
</VirtualHost>
```

IP_ADDRESS and PORT_NUMBER match the values from the Listen directive.

PARTIAL_IP_ADDRESS is the first 1 to 3 bytes of an IP address, to restrict access to a specific subnet - 10.33.144, for example.

3. Configure SELinux to allow proxy traffic

Enter the following commands as a root-equivalent user to modify the SELinux configuration to allow the proxy traffic:

```bash
# semanage port -a -t http_port_t -p tcp 8079 # add port to the Apache port list to enable the next command to work
# setsebool -P httpd_can_network_relay 1 # for mod_proxy to work
```

4. Optional: Disable Server Advertisement

The presence of the AdvertiseFrequency directive, which is set to five seconds here, causes the server to periodically send server advertisement messages via UDP multicast.

These server advertisement messages contain the IP_ADDRESS and PORT_NUMBER specified in the VirtualHost definition. Worker nodes that are configured to respond to server advertisements use this information to register themselves with the proxy server.

To disable server advertisement, add the following directive to the VirtualHost definition:

```xml
ServerAdvertise Off
```
If server advertisements are disabled, or UDP multicast is not available on the network between the proxy server and the worker nodes, you must configure worker nodes with a static list of proxy servers. Refer to Section 9.1, “Static proxy configuration” for directions.

5. **Restart the JBoss Enterprise Web Server Apache service**
   Refer to HTTPD-specific documentation for detailed instructions.
CHAPTER 8. INSTALL NODE WITH BASIC CONFIGURATION

Read this chapter to install the JBoss HTTP Connector on a worker node, and implement basic configuration for the node to begin immediate operation.

8.1. WORKER NODE REQUIREMENTS

Supported Worker Node types

- JBoss Enterprise Platform 5 JBoss Web component
- JBoss Enterprise Web Server Tomcat service

NOTE

JBoss Enterprise Platform worker nodes support all JBoss HTTP Connector functionality. JBoss Enterprise Web Server Tomcat worker nodes support a subset of JBoss HTTP Connector functionality.

JBoss HTTP Connector Enterprise Web Server Node Limitations

- Non-clustered mode only.
- Only one load metric can be used at a time when calculating the load balance factor.

8.2. INSTALL AND CONFIGURE A WORKER NODE

This section contains a number of tasks. Follow the appropriate task to install and configure a worker node on JBoss Enterprise Application Platform, or JBoss Enterprise Web Server.

NOTE

Make sure your configuration definition meets the following character limits:

- Maximum JVMRoute string length: 80 character (JVMRoute in the <Engine> element)
- Maximum hostname length for a node: 64 characters (hostname address in the <Connector> element)
- Maximum port length for a node: 7 characters (8009 is 4 characters, the port property in the <Connector> element)
- Maximum scheme length for a node: 6 characters (possible values are http, https, ajp, the protocol of the connector)

Task: Install and Configure a JBoss Enterprise Application Platform Worker Node

Follow this procedure to install JBoss HTTP Connector on a JBoss Enterprise Application Platform instance and configure it for non-clustered operation.

Prerequisites
• Install a supported JBoss Enterprise Application Platform.

• Understand the Proxy Configuration parameters discussed in Appendix B, Reference: Java properties

1. Deploy the worker node service
Copy mod-cluster.sar from the JBOSS_EAP_DIST/mod_cluster directory to jboss-as/server/PROFILE/deploy.

2. Add a Listener to JBoss Web
Add the following Listener element beneath the other Listeners in JBOSS_EAP_DIST/jboss-as/server/PROFILE/deploy/jbossweb.sar/server.xml:

```xml
<Listener
    className="org.jboss.web.tomcat.service.deployers.MicrocontainerIntegrationLifecycleListener"
    delegateBeanName="ModClusterService"/>
```

3. Configure the service dependency
Add the following depends element beneath the other depends elements in JBOSS_EAP_DIST/jboss-as/server/PROFILE/deploy/jbossweb.sar/META-INF/jboss-beans.xml:

```xml
<depends>ModClusterService</depends>
```

4. Give the worker a unique identity
Edit JBOSS_EAP_DIST/jboss-as/server/PROFILE/deploy/jbossweb.sar/server.xml and add a jvmRoute attribute and value to the Engine element, as shown:

```xml
<Engine name="jboss.web" defaultHost="localhost"
    jvmRoute="worker01">
```

Use a unique jvmRoute value for each node.

5. Optional: Configure firewall to receive multicast Proxy Server advertisements
A proxy server using the JBoss HTTP Connector can advertise itself via UDP multicast. To enable the worker node to dynamically discover proxy servers, open port 23364 for UDP connections on the worker node's firewall.

Use the following command on Red Hat Enterprise Linux to achieve this:

```bash
/sbin/iptables -A INPUT -m state --state NEW -m udp -p udp --dport 23364 -j ACCEPT
-m comment --comment "receive mod_cluster proxy server advertisements"
/sbin/iptables save
```

If you are not using Automatic Proxy Discovery (see Automatic Proxy Discovery), configure worker nodes with a static list of proxies. Refer to Section 9.1, “Static proxy configuration” for directions. In this case you can safely ignore the following warning message:

```
[warning] mod_advertise: ServerAdvertise Address or Port not defined, Advertise disabled!!!
```
**IMPORTANT**

If your nodes are on different machines that run Red Hat Enterprise Linux, they may not acknowledge each other automatically. JBoss Clustering relies on the UDP (User Datagram Protocol) multicasting provided by jGroups. Red Hat Enterprise Linux blocks these packets by default.

To allow the packets, modify the iptables rules (as root). The following commands apply to an IP address that matches 192.168.1.x:

```bash
/sbin/iptables -I RH-Firewall-1-INPUT 5 -p udp -d 224.0.1.0/24 -j ACCEPT
/sbin/iptables -I RH-Firewall-1-INPUT 5 -p udp -d 224.0.0.0/4 -j ACCEPT
/sbin/iptables -I RH-Firewall-1-INPUT 9 -p udp -s 192.168.1.0/24 -j ACCEPT
/sbin/iptables -I RH-Firewall-1-INPUT 10 -p tcp -s 192.168.1.0/24 -j ACCEPT
/etc/init.d/iptables save
```

**Task: Install and Configure a JBoss Enterprise Web Server Worker Node**

Follow this procedure to install the JBoss HTTP Connector on a JBoss Enterprise Web Server node and configure it for non-clustered operation.

**Prerequisites**

- Install a supported JBoss Enterprise Web Server.
- Understand the Proxy Configuration parameters discussed in Appendix B, Reference: Java properties

1. **Deploy worker node service**
   Copy all of the library files in the `JBOSS_EAP_DIST/mod_cluster/JBossWeb-Tomcat/lib` directory. Move these files to `JBOSS_EWS_DIST/tomcat6/lib/`

2. **Add a Listener to Tomcat**
   Add the following Listener element beneath the other Listener elements in `JBOSS_EWS_DIST/tomcat6/conf/server.xml`.

   ```xml
   <Listener className="org.jboss.modcluster.ModClusterListener" advertise="true" stickySession="true" stickySessionForce="false" stickySessionRemove="true"/>
   ```

3. **Give this worker a unique identity**
   Edit `JBOSS_EWS_DIST/tomcat6/conf/server.xml` and add a `jvmRoute` attribute and value to the `<Engine>` element, as shown:

   ```xml
   <Engine name="Catalina" defaultHost="localhost" jvmRoute="worker01">
   ```

4. **Optional: Configure firewall to receive Proxy Server advertisements**
   A proxy server using the JBoss HTTP Connector can advertise itself via UDP multicast. To receive these multicast messages, open port 23364 for UDP connections on the worker node's
firewall.

For Linux users:

```
/sbin/iptables -A INPUT --state NEW -m udp -p udp --dport 23364 -j ACCEPT
-m comment -comment "receive mod_cluster proxy server advertisements"
```

If you are not using Automatic Proxy Discovery (see Automatic Proxy Discovery), configure worker nodes with a static list of proxies. Refer to Section 9.1, “Static proxy configuration” for directions. In this case you can safely ignore the following warning message:

```
[warning] mod_advertise: ServerAdvertise Address or Port not defined, Advertise disabled!!!
```
CHAPTER 9. ADVANCED CONFIGURATION

Read this chapter to configure advanced features of the JBoss HTTP Connector.

9.1. STATIC PROXY CONFIGURATION

Server advertisement allows worker nodes to dynamically discover and register themselves with proxy servers. If UDP broadcast is not available or server advertisement is disabled then worker nodes must be configured with a static list of proxy server addresses and ports.

NOTE

Make sure your configuration definition meets the following character limits:

- Maximum Alias length: 100 character (for example, if myapp.war is deployed in /myapp, then /myapp is the context)
- Maximum balancer name length: 40 (the balancer property in MBean)
- Maximum domain name length: 20 (the domain property in MBean)

Task: Configure Application Platform Worker Node with Static Proxy List

Follow this task to configure a JBoss Enterprise Application Platform worker node to operate with a static list of proxy servers.

Prerequisites

- JBoss Enterprise Application Platform worker node configured. Refer to Chapter 8, Install node with basic configuration for directions.

1. Disable dynamic proxy discovery
   Edit the file $JBOSS_EAP_DIST/jboss-as/server/PROFILE/mod-cluster.sar/META-INF/mod-cluster-jboss-beans.xml and set the advertise property to false:

   ```xml
   <property name="advertise">false</property>
   ```

2. Choose, and implement, one of the following static proxy options:

   - Option 1: Create a static proxy server list

   ```xml
   <property name="proxyList">10.33.144.3:6666,10.33.144.1:6666</property>
   ```

   - Option 2: Start the worker node with a static proxy list as a parameter
     a. Edit $JBOSS_EAP_DIST/server/PROFILE/mod-cluster.sar/META-INF/mod-
cluster-jboss-beans.xml

b. Add the following line:

```xml
<property name="domain">${jboss.modcluster.domain:}</property>
```

c. Add a comma-separated list of proxies in the form of *IP_ADDRESS:PORT* as the `jboss.modcluster.proxyList` parameter when starting the node.

**Example 9.2. Example Static Proxy List Parameter**

```bash
-Djboss.modcluster.domain=10.33.144.3:6666,10.33.144.1:6666
```

---

**Task: Configure Web Server Worker Node with Static Proxy List**

Follow this procedure to configure a JBoss Enterprise Web Server worker node to operate with a static list of proxy servers.

**Prerequisites**

- JBoss Enterprise Web Server worker node configured. Refer to *Chapter 8, Install node with basic configuration* for directions.
- Understand the Proxy Configuration parameters discussed in *Appendix B, Reference: Java properties*

1. **Disable dynamic proxy discovery**
   Edit `JBOSS_EWS_DIST/tomcat6/conf/server.xml` and set the `advertise` property of the ModClusterListener to false:

   ```xml
   <Listener className="org.jboss.modcluster.ModClusterListener" advertise="false" stickySession="true" stickySessionForce="false" stickySessionRemove="true"/>
   ```

2. **Define a mod_cluster listener**
   Add a `<Listener>` element to `server.xml`.

   ```xml
   <Listener className="org.jboss.modcluster.ModClusterListener" advertise="false" stickySession="true" stickySessionForce="false" stickySessionRemove="true" proxyList="10.33.144.3:6666,10.33.144.1:6666"/>
   ```

3. **Create a static proxy server list**
   Add a comma separated list of proxies in the form of *IP_ADDRESS:PORT* as the `proxyList` property of the ModClusterListener `<Listener>` element.

**Example 9.3. Example Static Proxy List**

```xml
<Listener className="org.jboss.modcluster.ModClusterListener" advertise="false" stickySession="true" stickySessionForce="false" stickySessionRemove="true" proxyList="10.33.144.3:6666,10.33.144.1:6666"/>
```
NOTE

Only JBoss Enterprise Application Platform nodes support clustered operation with the JBoss HTTP Connector. JBoss Enterprise Web Server nodes support non-clustered operation only.

JBoss HTTP Connector non-clustered operation

In non-clustered mode each worker node communicates directly with the proxy.

JBoss HTTP Connector clustered operation

In clustered mode multiple worker nodes form a JBoss HA (High Availability) cluster domain. A single worker node communicates with the proxy on behalf of the other nodes in the cluster domain.
10.1. CONFIGURATION PROPERTIES

The tables below enumerate the configuration properties available to an application server node. The location for these properties depends on how mod_cluster is configured.

10.1.1. Proxy Discovery Configuration

The list of proxies from which an application expects to receive AJP connections is either defined statically, via the addresses defined in the `proxyList` configuration property; or discovered dynamically via the `advertise` mechanism.

Using a special `mod_advertise` module, proxies can advertise their existence by periodically broadcasting a multicast message containing its address/port.

This functionality is enabled via the `advertise` configuration property. If configured to listen, a server can learn of the proxy's existence, then notify that proxy of its own existence, and update its configuration accordingly. This frees both the proxy and the server from having to define static, environment-specific configuration values.

**Table 10.1. Proxy Discovery**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>proxyList</code></td>
<td>None</td>
<td>Defines a comma-separated list of httpd proxies with which this node will initially communicate. Value should be of the form: <code>address1:port1,address2:port2</code> Using the default configuration, this property can be manipulated via the <code>jboss.modcluster.proxyList</code> system property.</td>
</tr>
<tr>
<td><code>excludedContexts</code></td>
<td>ROOT, admin-console, invoker, jbossws, jmx-console, juddi, web-console</td>
<td>List of contexts to exclude from httpd registration, of the form: <code>host1:context1,host2:context2,host3:context3</code> If no host is indicated, it is assumed to be the default host of the server (e.g. localhost). &quot;ROOT&quot; indicates the root context. Using the default configuration, this property can be manipulated via the <code>jboss.modcluster.excludedContexts</code> system property.</td>
</tr>
<tr>
<td>Attribute</td>
<td>Default</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------</td>
<td>---------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>autoEnableContexts</code></td>
<td>True</td>
<td>If <code>false</code>, the contexts are registered disabled in <code>httpd</code>, they need to be enabled via the <code>enable()</code> MBean method or via <code>mod_cluster_manager</code>.</td>
</tr>
<tr>
<td><code>stopContextTimeout</code></td>
<td>10</td>
<td>The number of seconds to wait for clean shutdown of a context. This could be the completion of all pending requests for a distributable context or the destruction/expiration of active sessions for a non-distributable context.</td>
</tr>
<tr>
<td><code>proxyURL</code></td>
<td>None</td>
<td>If defined, this value will be prepended to the URL of MCMP commands.</td>
</tr>
<tr>
<td><code>socketTimeout</code></td>
<td>20000</td>
<td>Number of milliseconds to wait for a response from an <code>httpd</code> proxy to MCMP commands before timing out and flagging the proxy as in error.</td>
</tr>
<tr>
<td><code>advertise</code></td>
<td>This is <code>true</code> if <code>proxyList</code> is undefined, <code>false</code> otherwise.</td>
<td>If enabled, <code>httpd</code> proxies will be auto-discovered via multicast announcements. This can be used either in concert or in place of a static <code>proxyList</code>.</td>
</tr>
<tr>
<td><code>advertiseGroupAddress</code></td>
<td>224.0.1.105</td>
<td>The UDP address on which to listen for <code>httpd</code> proxy multicast advertisements.</td>
</tr>
<tr>
<td><code>advertisePort</code></td>
<td>23364</td>
<td>The UDP port on which to listen for <code>httpd</code> proxy multicast advertisements.</td>
</tr>
<tr>
<td><code>advertiseSecurityKey</code></td>
<td>None</td>
<td>If specified, <code>httpd</code> proxy advertisements checksums will be verified using this value as a salt.</td>
</tr>
</tbody>
</table>

### 10.1.2. Proxy Configuration

The following configuration values are sent to proxies during server startup, when a proxy is detected via the advertise mechanism, or during the resetting of a proxy's configuration during error recovery.

Table 10.2. Proxy Configuration
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>stickySession</td>
<td>true</td>
<td>Indicates whether subsequent requests for a given session should, if possible, be routed to the same node.</td>
</tr>
<tr>
<td>stickySessionRemove</td>
<td>false</td>
<td>Indicates whether the httpd proxy should remove session stickiness in the event that the balancer is unable to route a request to the node to which it is stuck. This property is ignored if stickySession is false.</td>
</tr>
<tr>
<td>stickySessionForce</td>
<td>true</td>
<td>Indicates whether the httpd proxy should return an error in the event that the balancer is unable to route a request to the node to which it is stuck. This property is ignored if stickySession is false.</td>
</tr>
<tr>
<td>workerTimeout</td>
<td>-1</td>
<td>Number of seconds to wait for a worker to become available to handle a request. When all the workers of a balancer are unusable, mod_cluster will retry after a specified period (workerTimeout/100) to find a usable worker. A value of -1 indicates that the httpd will not wait for a worker to be available and will return an error if none is available.</td>
</tr>
<tr>
<td>maxAttempts</td>
<td>1</td>
<td>Number of times an httpd proxy will attempt to send a given request to a worker before giving up. The minimum value is 1, meaning try only once. Note that mod_proxy default is also 1: no retry.</td>
</tr>
<tr>
<td>flushPackets</td>
<td>false</td>
<td>Enables/disables packet flushing.</td>
</tr>
<tr>
<td>flushWait</td>
<td>-1</td>
<td>Time to wait before flushing packets. A value of -1 means wait forever.</td>
</tr>
<tr>
<td>ping</td>
<td>10 seconds</td>
<td>Time to wait for an answer to a ping.</td>
</tr>
<tr>
<td>Attribute</td>
<td>Default</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>--------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>smax</strong></td>
<td>Determined by <strong>httpd</strong> configuration.</td>
<td>Soft maximum idle connection count (that is the <strong>smax</strong> in worker mod_proxy documentation). The maximum value depends on the <strong>httpd</strong> thread configuration (<strong>ThreadsPerChild</strong> or <strong>1</strong>).</td>
</tr>
<tr>
<td><strong>ttl</strong></td>
<td><strong>60</strong> seconds</td>
<td>Time to live (in seconds) for idle connections above <strong>smax</strong>.</td>
</tr>
<tr>
<td><strong>nodeTimeout</strong></td>
<td>-1 (none)</td>
<td>Timeout (in seconds) for proxy connections to a node. That is the time mod_cluster will wait for the back-end response before returning an error.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>This corresponds to <strong>timeout</strong> in the worker mod_proxy documentation.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Note that mod_cluster always uses a CPing/CPong before forwarding a request and the connectiontimeout value used by mod_cluster is the ping value.</td>
</tr>
<tr>
<td><strong>balancer</strong></td>
<td><strong>mycluster</strong></td>
<td>The balancer name.</td>
</tr>
<tr>
<td><strong>domain</strong></td>
<td>None</td>
<td>If specified, load will be balanced among <strong>jvmRoutes</strong> with the same domain. This is primarily used in conjunction with partitioned session replication (e.g. buddy replication).</td>
</tr>
</tbody>
</table>

Note: When nodeTimeout is not defined the ProxyTimeout directive **Proxy** is used. If ProxyTimeout is not defined the server timeout (**Timeout**) is used (default 300 seconds). **nodeTimeout**, ProxyTimeout or **Timeout** is set at the socket level.

### 10.1.3. SSL Configuration

The communication channel between application servers and **httpd** proxies uses HTTP by default. This channel can be secured using HTTPS by setting the **ssl** parameter to true.

Note: This HTTP/HTTPS channel should not be confused with the processing of HTTP/HTTPS client requests.
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ssl</td>
<td>false</td>
<td>Should connection to proxy use a secure socket layer.</td>
</tr>
<tr>
<td>sslCiphers</td>
<td>The default JSSE cipher suites</td>
<td>Overrides the cipher suites used to init an SSL socket ignoring any unsupported ciphers.</td>
</tr>
<tr>
<td>sslProtocol</td>
<td>TLS</td>
<td>Overrides the default SSL socket protocol.</td>
</tr>
<tr>
<td>sslCertificateEncodingAlgorithm</td>
<td></td>
<td>The default JSSE key manager algorithm.</td>
</tr>
<tr>
<td>sslKeyStore</td>
<td>System.getProperty(&quot;user.home&quot;) + &quot;/.keystore&quot;</td>
<td>The location of the key store containing client certificates.</td>
</tr>
<tr>
<td>sslKeyStorePass</td>
<td>changeit</td>
<td>The password granting access to the key store.</td>
</tr>
<tr>
<td>sslKeyStoreType</td>
<td>JKS</td>
<td>The type of key store.</td>
</tr>
<tr>
<td>sslKeyStoreProvider</td>
<td>The default JSSE security provider.</td>
<td>The key store provider.</td>
</tr>
<tr>
<td>sslTrustAlgorithm</td>
<td></td>
<td>The algorithm of the trust manager factory.</td>
</tr>
<tr>
<td>sslKeyAlias</td>
<td></td>
<td>The alias of the key holding the client certificates in the key store.</td>
</tr>
<tr>
<td>sslCrlFile</td>
<td></td>
<td>Certificate revocation list.</td>
</tr>
<tr>
<td>sslTrustMaxCertLength</td>
<td>5</td>
<td>The maximum length of a certificate held in the trust store.</td>
</tr>
<tr>
<td>sslTrustStore</td>
<td>System.getProperty(&quot;javax.net.ssl.trustStorePassword&quot;)</td>
<td>The location of the file containing the trust store.</td>
</tr>
<tr>
<td>sslTrustStorePassword</td>
<td>System.getProperty(&quot;javax.net.ssl.trustStorePassword&quot;)</td>
<td>The password granting access to the trust store.</td>
</tr>
<tr>
<td>sslTrustStoreType</td>
<td>System.getProperty(&quot;javax.net.ssl.trustStoreType&quot;)</td>
<td>The trust store type.</td>
</tr>
<tr>
<td>sslTrustStoreProvider</td>
<td>System.getProperty(&quot;javax.net.ssl.trustStoreProvider&quot;)</td>
<td>The trust store provider.</td>
</tr>
</tbody>
</table>
10.1.4. HA Configuration

Additional configuration properties when mod_cluster is configured in clustered mode.

Table 10.4. HA Configuration

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>masterPerDomain</td>
<td>false</td>
<td>If the <em>domain</em> directive is used, should HA partition use a singleton master per domain.</td>
</tr>
</tbody>
</table>

10.1.5. Load Configuration

Additional configuration properties used when mod_cluster is configured in JBoss Web standalone or Tomcat.

Table 10.5. Load Configuration

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>loadMetricClass</td>
<td>org.jboss.modcluster.load.metric.impl.BusyConnectorsLoadMetric</td>
<td>Class name of an object implementing org.jboss.load.metric.LoadMetric.</td>
</tr>
<tr>
<td>loadMetricCapacity</td>
<td>1</td>
<td>The capacity of the load metric defined via the <em>loadMetricClass</em> property.</td>
</tr>
<tr>
<td>loadHistory</td>
<td>9</td>
<td>The number of historic load values to consider in the load balance factor computation.</td>
</tr>
<tr>
<td>loadDecayFactor</td>
<td>2</td>
<td>The factor by which a historic load values should degrade in significance.</td>
</tr>
</tbody>
</table>
CHAPTER 11. LOAD METRICS

11.1. SERVER-SIDE LOAD METRICS

A major feature of mod_cluster is the ability to use server-side load metrics to determine how best to balance requests.

The `DynamicLoadBalanceFactorProvider` bean computes the load balance factor of a node from a defined set of load metrics.

Load metrics can be configured with an associated weight and capacity:

1. The weight (default is 1) indicates the significance of a metric with respect to the other metrics. For example, a metric of weight 2 will have twice the impact on the overall load factor than a metric of weight 1.

2. The capacity of a metric serves 2 functions:

Each load metric contributes a value to the overall load factor of a node. The load factors from each metric are aggregated according to their weights.

In general, the load factor contribution of given metric is:

\[
L = \frac{\text{load}}{\text{capacity}} \times \text{weight} \div \text{total weight}
\]

The `DynamicLoadBalanceFactorProvider` applies a time decay function to the loads returned by each metric. The aggregate load, with respect to previous load values, can be expressed by the following formula:

\[
L = \left( L_0 + \frac{L_1}{D} + \frac{L_2}{D^2} + \frac{L_3}{D^3} + \ldots + \frac{L_H}{D^H} \right) \times \left( 1 + D + D^2 + D^3 + \ldots + D^H \right)
\]

... or more concisely as:

\[
L = \sum_{i=0}^{H} \frac{L_i}{D_i} \times \sum_{i=0}^{H} D_i
\]
... where \( D = \text{decayFactor} \) and \( H = \text{history} \).

Setting \( \text{history} = 0 \) effectively disables the time decay function and only the current load for each metric will be considered in the load balance factor computation.

The mod_cluster proxy module expects the load factor to be an integer between 0 and 100, where 0 indicates max load and 100 indicates zero load. Therefore, the final load balance factor sent to the proxy is:

\[
100 - (L \times 100)
\]

While you are free to write your own load metrics, the following \textit{LoadMetrics} are available out of the box:

\section*{11.2. WEB CONTAINER METRICS}

1. \texttt{ActiveSessionsLoadMetric}
   - Requires an explicit capacity.
   - Uses \texttt{SessionLoadMetricSource} to query session managers
   - Analogous to method=S in mod_jk

   For example:

   ```xml
   <bean name="ActiveSessionsLoadMetric" class="org.jboss.modcluster.load.metric.impl.ActiveSessionsLoadMetric" mode="On Demand">
     <annotation>@org.jboss.aop.microcontainer.aspects.jmx.JMX(name="jboss.web:service=ActiveSessionsLoadMetric",exposedInterface=org.jboss.modcluster.load.metric.LoadMetricMBean.class)</annotation>
     <constructor>
       <parameter><inject bean="SessionLoadMetricSource"/></parameter>
     </constructor>
     <property name="capacity">1000</property>
   </bean>

   <bean name="SessionLoadMetricSource" class="org.jboss.modcluster.load.metric.impl.SessionLoadMetricSource" mode="On Demand">
     <constructor>
       <parameter class="javax.management.MBeanServer"><inject bean="JMXKernel" property="mbeanServer"/></parameter>
     </constructor>
   </bean>
   ``

2. \texttt{BusyConnectorsLoadMetric}
   - Returns the percentage of connector threads from the thread pool that are busy servicing requests
   - Uses \texttt{ThreadPoolLoadMetricSource} to query connector thread pools
   - Analogous to method=B in mod_jk

   For example:
3. ReceiveTrafficLoadMetric

- Returns the incoming request traffic in KB/sec
- Requires an explicit capacity
- Uses RequestProcessorLoadMetricSource to query request processors
- Analogous to method=T in mod_jk

For example:

```xml
<bean name="ReceiveTrafficLoadMetric" class="org.jboss.modcluster.load.metric.impl.ReceiveTrafficLoadMetric" mode="On Demand">
  <constructor>
    <parameter class="org.jboss.modcluster.load.metric.impl.RequestProcessorLoadMetricSource" inject="RequestProcessorLoadMetricSource"/>
  </constructor>
  <property name="capacity">1024</property>
</bean>

<bean name="RequestProcessorLoadMetricSource" class="org.jboss.modcluster.load.metric.impl.RequestProcessorLoadMetricSource" mode="On Demand">
  <constructor>
    <parameter class="javax.management.MBeanServer" inject="JMXKernel" property="mbeanServer"/>
  </constructor>
</bean>
```

4. SendTrafficLoadMetric

- Returns the outgoing request traffic in KB/sec
- Requires an explicit capacity
- Uses RequestProcessorLoadMetricSource to query request processors
- Analogous to method=T in mod_jk

For example:

```xml
<bean name="SendTrafficLoadMetric" class="org.jboss.modcluster.load.metric.impl.SendTrafficLoadMetric" mode="On Demand">
  <annotation>@org.jboss.aop.microcontainer.aspects.jmx.JMX(name="jboss.web:service=SendTrafficLoadMetric",exposedInterface=org.jboss.modcluster.load.metric.LoadMetricMBean.class)</annotation>
  <constructor>
    <parameter class="org.jboss.modcluster.load.metric.impl.RequestProcessorLoadMetricSource"><inject bean="RequestProcessorLoadMetricSource"/></parameter>
  </constructor>
  <property name="capacity">512</property>
</bean>
```

- RequestCountLoadMetric
  - Returns the number of requests/sec
  - Requires an explicit capacity
  - Uses RequestProcessorLoadMetricSource to query request processors
  - Analogous to method=R in mod_jk

For example:

```xml
<bean name="RequestCountLoadMetric" class="org.jboss.modcluster.load.metric.impl.RequestCountLoadMetric" mode="On Demand">
  <annotation>@org.jboss.aop.microcontainer.aspects.jmx.JMX(name="jboss.web:service=RequestCountLoadMetric",exposedInterface=org.jboss.modcluster.load.metric.LoadMetricMBean.class)</annotation>
  <constructor>
    <parameter class="org.jboss.modcluster.load.metric.impl.RequestProcessorLoadMetricSource"><inject bean="RequestProcessorLoadMetricSource"/></parameter>
  </constructor>
  <property name="capacity">1000</property>
</bean>
```

### 11.3. SYSTEM/JVM METRICS

1. AverageSystemLoadMetric
   - Returns CPU load
   - Requires Java 1.6+
   - Uses OperatingSystemLoadMetricSource to generically read attributes

For example:
11.4. OTHER METRICS

1. ConnectionPoolUsageLoadMetric

2. SystemMemoryUsageLoadMetric

- Returns system memory usage
- Requires com.sun.management.OperatingSystemMXBean (available in Sun's JDK or OpenJDK)
- Uses OperatingSystemLoadMetricSource to generically read attributes

   For example:

   ```
   <bean name="SystemMemoryUsageLoadMetric" class="org.jboss.modcluster.load.metric.impl.SystemMemoryUsageLoadMetric" mode="On Demand">
   <annotation>@org.jboss.aop.microcontainer.aspects.jmx.JMX(name="jboss.web:service=SystemMemoryUsageLoadMetric",exposedInterface=org.jboss.modcluster.load.metric.LoadMetricMBean.class)</annotation>
   </bean>
   ```

3. HeapMemoryUsageLoadMetric

- Returns the heap memory usage as a percentage of max heap size

   For example:

   ```
   <bean name="HeapMemoryUsageLoadMetric" class="org.jboss.modcluster.load.metric.impl.HeapMemoryUsageLoadMetric" mode="On Demand">
   <annotation>@org.jboss.aop.microcontainer.aspects.jmx.JMX(name="jboss.web:service=HeapMemoryUsageLoadMetric",exposedInterface=org.jboss.modcluster.load.metric.LoadMetricMBean.class)</annotation>
   </bean>
   ```
- Returns the percentage of connections from a connection pool that are in use.
- Uses ConnectionPoolLoadMetricSource to query JCA connection pools

For example:

```xml
<bean name="ConnectionPoolUsageMetric" class="org.jboss.modcluster.load.metric.impl.ConnectionPoolUsageLoadMetric" mode="On Demand">  
  <annotation>@org.jboss.aop.microcontainer.aspects.jmx.JMX(name="jboss.web:service=ConnectionPoolUsageLoadMetric",exposedInterface=org.jboss.modcluster.load.metric.LoadMetricMBean.class)</annotation>
  
  <constructor>
    <parameter><inject bean="ConnectionPoolLoadMetricSource"/></parameter>
  </constructor>
</bean>

<bean name="ConnectionPoolLoadMetricSource" class="org.jboss.modcluster.load.metric.impl.ConnectionPoolLoadMetricSource" mode="On Demand">
  
  <constructor>
    <parameter class="javax.management.MBeanServer"><inject bean="JMXKernel" property="mbeanServer"/></parameter>
  </constructor>
</bean>
```
CHAPTER 12. LOAD BALANCING DEMONSTRATION

The JBoss HTTP Connector includes a load balancing demonstration to show how different server-side scenarios affect the client request routing performed by the load balancing proxy server. The required configuration is located in the `JBOSS_EAP_DIST/mod_cluster/demo` directory.

The application consists of two primary components:

`/server/load-demo.war`
A WAR file to be deployed in JBoss Enterprise Application Platform or JBoss Enterprise Web Server. This WAR includes a number of servlets.

`/client/lib/mod-cluster-demo.jar`
A web application that lets users launch a pool of threads, by sending requests through the load balancer to the application’s primary servlet. The application displays information about which servers are handling the requests. It can also send separate requests to the application’s load-generation servlets, allowing the user to see how certain load conditions affect request load balancing.

This application can be used to demonstrate how different worker-side scenarios impact the routing decisions of the proxy server.

**IMPORTANT**
If running the demonstration on JBoss Enterprise Web Server, the only metrics available will be System, and JVM metrics. The demonstration application is not designed to interact with any other metrics in Tomcat 6.

**NOTE**
The demonstration application does not show the maximum load a cluster configuration can handle.

12.1. SET UP THE DEMONSTRATION

The following procedure summarizes how to set up and start the demonstration. These steps will then be explained in further detail. Once the demonstration is running, refer to Section 12.3, "Interact with the demonstration".

Task: Start the Demo

Complete this task to set up the base requirements of the demonstration.

**Prerequisites**

- Install and Configure the Worker Node. Refer to Section 8.2, "Install and configure a worker node"
- Install and Configure the Proxy Server. Refer to Section 9.1, “Static proxy configuration”

1. Start the Proxy Server
   Navigate to `HTTPD_DIST/sbin` and start the proxy server.
2. **Start the Worker Node**
   In a terminal, execute the following command:
   - For JBoss Enterprise Web Server:
     
     ```bash
     [home]$ ./JOBS_EWS_DIST/tomcat6/bin/startup.sh
     ```
   - For JBoss Enterprise Application Platform:
     
     ```bash
     [home]$ ./JOBS_EAP_DIST/bin/run.sh
     ```

3. **On JBoss Enterprise Web Server, specify the Catalina Service Name**
   Tomcat 6 only: In `$JOBS_EWS_DIST/mod_cluster/src/demo/resources/web.xml`, under the `<web-app>` element, append a `<context-param>` directive, which specifies Catalina as a service.

   ```xml
   <context-param>
   <param-name>service-name</param-name>
   <param-value>Catalina</param-value>
   </context-param>
   ```

4. **Deploy Demo Web Archive to Worker Node**
   Copy `load-demo.war` from `$JOBS_EAP_DIST/mod_cluster/demo/server` into one of the following directories:
   - For JBoss Enterprise Web Server:
     
     `$JOBS_EWS_DIST/tomcat6/webapps`
   - For JBoss Enterprise Application Platform:
     
     `$JOBS_EAP_DIST/jboss-as/server/PROFILE/deploy`

5. **Start the Demonstration**
   Navigate to `$JOBS_EAP_DIST/mod_cluster/demo/client/`, and start the demonstration.

   ```bash
   [client]$ ./run-demo.sh
   ```

**Result**

The demonstration starts, and the Load Balancing Demonstration window opens. Proceed to Task: **Configure Client Control Tab Fields**
12.2. CONFIGURE THE DEMO CLIENT

You must configure the demonstration's Client Control parameters to ensure the client operates as expected throughout the demonstration.

**Task: Configure Client Control Tab Fields**

Complete this task to configure the **Client Control** tab of the Load Balancing Demonstration.

**Terms**

**Proxy Hostname**

Hostname of the load-balancing proxy server, or the IP address on which the proxy server is listening for requests. The default value for this field is **localhost**, or determined by the `mod_cluster.proxy.host` system property, if set.

Edit the `-Dmod_cluster.proxy.host=localhost` value in `run-demo.sh` to avoid re-setting this value each time you use the demo.

**Proxy Port**

Port on which the load-balancing proxy server listens for requests. The default value is **8000**, or determined by the `mod_cluster.proxy.port` property, if set.

Edit the `-Dmod_cluster.proxy.port=8000` value in `run-demo.sh` to avoid re-setting this value each time you use the demo.

**Context Path**

The part of the request URL that specifies the request is for **load-demo.war**.

**Session Life**

<table>
<thead>
<tr>
<th>Target Hostname</th>
<th>Target Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>localhost</td>
<td>8000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Load Creation Action</th>
<th>Number of Connections</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connection Pool Use</td>
<td>50</td>
<td>15</td>
</tr>
</tbody>
</table>
Number of seconds a client thread should use a session before invalidating or abandoning it. This should be a small value, or session stickiness can prevent changes in server load from affecting the proxy server’s routing decisions. When sticky sessions are enabled (strongly recommended), the creation of new sessions allows the proxy to balance the workload.

invalidate
When checked, specifies that a session is invalidated by the client thread when the thread stops using a session. When unchecked, the session is abandoned, and exists on the worker node until the session timeout expires.

session timeout
The number of seconds a session can remain unused before the worker node can expire and remove the session.

Deselecting invalidate and setting a high value relative to session life causes a significant number of unused sessions to accumulate on the server.

num threads
Number of client threads to launch. Each thread repeatedly makes requests until the stop button is pressed, or a request receives a response other than HTTP 200.

sleep time
Number of milliseconds that client threads should sleep between requests.

startup time
Number of seconds over which the application should stagger client thread start-up. Staggering the start time of sessions avoids the unrealistic situation where all sessions start and end almost simultaneously. Staggering the start time allows the proxy to continually see new sessions and decide how to route them.

prerequisites
Complete Task: Start the Demo before continuing with this task.

1. Click the Client Control tab.
2. Supply values for all fields on the Client Control tab, referring to the list of terms above.
3. Once you have specified the values, proceed to Task: Interact with the Demonstration.

12.3. INTERACT WITH THE DEMONSTRATION

Terms
Active Sessions
A session is considered active if a client thread will ever send a request associated with the session. When client threads stop using a session, they can either send a request to invalidate it, or abandon it by no longer including its session cookie in requests.

Once a session has been abandoned, it is no longer reflected in the Session Balancing chart, but will continue to exist on the worker node until it is removed based on session timeout values.
Total Clients
The number of client threads created since the last time the Start button was clicked.

Live Clients
The number of client threads currently running.

Failed Clients
The number of clients threads that terminated abnormally, for example, a request that resulted in something other than a HTTP 200 response.

This section shows you how to configure and start using the demo.

Task: Interact with the Demonstration
Complete this task to test the effects of changing load-balancing parameters.

Prerequisites
- Complete Task: Start the Demo.
- Complete Task: Configure Client Control Tab Fields.

1. Click on the Request Balancing tab to see how many requests are going to each of the worker nodes.

2. Click on the Session Balancing tab to see how many active sessions are being hosted by each of the worker nodes.

3. Stop some of the worker nodes, or undeploy load-demo.war, and observe the effect that this has on request and session balancing.

4. Restart some of the worker nodes, or re-deploy the load-demo.war to some of the workers, and observe the effect that this has on request and session balancing.

5. Experiment with adding artificial load to one or more worker nodes and observe the effects on load and session balancing. (See Section 12.3.1, “Generate artificial load” for details.)

12.3.1. Generate artificial load
You can use the Load Balancing Demonstration to instruct your worker nodes to generate various types of load, and then track how that load affects request and session balancing. Load generation is controlled in the Server Load Control tab:

Target Hostname, Target Port
The hostname and port number of the server on which to generate load. There are two strategies for setting these values:

1. Use the hostname and port of the proxy server. The proxy will route the load to a worker node. However, the client does not maintain a session cookie for these requests, so subsequent generated load will not necessarily be routed to the same worker.
2. If the worker nodes are running the HttpConnector and the AJP connector, you can specify the IP address and port on which a worker's HttpConnector is listening. (The default is 8080.)

Load Creation Action

Specifies the type of load the worker node should generate.

Available Actions

Active Sessions
Generates server load by causing session creation on the target server.

Datasource Use
Generates server load by taking connections from the `java:DefaultDS` datasource for a set time.

Connection Pool Use
Generates server load by blocking threads in the webserver connections pool for a set time.

Heap Memory Pool Use
Generates server load by filling 50% of free heap memory for a set time.

CPU Use
Generates server CPU load by initiating a tight loop in a thread.

Server Receive Traffic
Generates server traffic receipt load by POSTing a large byte array to the server once per second for a set time.

Server Send Traffic
Generates server traffic send load by making a request once per second, to which the server responds with a large byte array.

Request Count
Generates server load by making numerous requests, increasing the request count on the target server.

Params
Zero or more parameters to pass to the specified load creation servlet, for example, Number of Connections and Duration, as seen in the screenshot. The parameters displayed, their name, and their meaning depend on the selected Load Creation Action. The label for each parameter includes a tooltip that explains its use.
CHAPTER 13. OVERVIEW

Read this chapter for a brief introduction about the Internet Server Application Programming Interface (ISAPI).

13.1. WHAT IS INTERNET SERVER API

Internet Server Application Programming Interface (ISAPI) is a multi-tier application programming interface for Microsoft Internet Information Services (IIS) web servers, and other compatible third-party web servers.

Two application types exist for ISAPI applications:

- Extensions (full applications that run on IIS); and
- Filters (applications that modify or enhance IIS functionality by constantly filtering for requests relevant to their functionality).

ISAPI applications are implemented by compiling Extensions or Filters into Dynamic Link Library (DLL) files. These DLLs must then be registered with the web server before they are available for use.
CHAPTER 14. CONFIGURING THE ISAPI CONNECTOR ON WINDOWS

Read this chapter to learn how to configure the ISAPI connector to use JBoss Enterprise Application Platform as a worker node for a Windows Server 2003 or 2008 master node.

14.1. PREREQUISITES AND CONFIGURATION ASSUMPTIONS

Complete the following prerequisites before continuing with the tasks that follow:

- On the master node install one of the following technology combinations, and the appropriate Native binary for its operating system and architecture.
  - Windows Server 2003 (32-bit) with Microsoft IIS 6
  - Windows Server 2003 (64-bit) with Microsoft IIS 6
  - Windows Server 2008 (32-bit) with Microsoft IIS 7.0
  - Windows Server 2008 (64-bit) with Microsoft IIS 7.0
- On the worker nodes install JBoss Enterprise Application Platform 5.2 or later. The Native components are optional for worker nodes.

Refer to the *Installation Guide* for assistance with these installation prerequisites.

14.2. CONFIGURE SERVER INSTANCE AS A WORKER NODE

Task: Configure Server Instance as a Worker Node

Complete this task to correctly configure your JBoss Enterprise Application Platform instance as a worker node for use with Microsoft Internet Information Services (IIS).

Prerequisites

- Section 14.1, “Prerequisites and configuration assumptions”

1. Create a server profile for each worker node
   Make a copy of the server profile you want to configure as a worker node, and rename it - worker-01 for example.

2. Give each instance a unique name
   Edit the following line in the `<PROFILE>\deploy\jbossweb.sar\server.xml` file of each new worker instance:

   ```xml
   <Engine name="jboss.web" defaultHost="localhost">
   
   Add a unique `jvmRoute` value, as shown. This value is the identifier for this node in the cluster.

   ```xml
   <Engine name="jboss.web" defaultHost="localhost" jvmRoute="worker-01">
Enable session handling
Uncomment the following line in the `PROFILE\deployers\jbossweb.deployer\META-INF\war-deployers-jboss-beans.xml` file of each worker node:

```
<property name="useJK">false</property>
```

This property controls whether special session handling is used to coordinate with mod_jk and other connector variants. Set this property to `true` in both worker nodes:

```
<property name="useJK">true</property>
```

Start worker nodes
Start each worker node in a separate command line interface. Ensure that each node is bound to a different IP address with the `--host` switch and that the profile is specified with the `-c` switch:

```
JBoss_EAP_DIST\bin\run.bat --host=127.0.0.1 -c worker-01
```

14.3. MICROSOFT IIS 6 INITIAL CLUSTERING CONFIGURATION

Microsoft IIS 6 contains basic ISAPI filters and ISAPI mapping as part of the default installation. Here we create a filter to direct web requests to JBoss Enterprise Application Platform.

Task: Define ISAPI Filter

Complete this task to define the ISAPI Filter on the Master server using the management console.

1. On the master server, open IIS Manager:
   - Start → Run, then type `inetmgr` and hit Enter.
   - Start → Control Panel → Administrative Tools → Internet Information Services (IIS) Manager

   The IIS 6 Manager window opens.

2. In the tree view pane, expand `Web Sites`

3. Right click on Default Web Site, and then click `Properties`

   The Properties window opens.

4. Click the ISAPI Filters tab.

5. Click the Add button, and specify the following values in the Add/Edit Filter Properties window:

   **Filter name:**
   - Specify `jboss` (exactly as written).

   **Executable:**
   - Specify `NATIVE\sbin\isapi_redirect.dll`
Click OK to save the values, and close the Add/Edit Filter Properties dialog.

**NOTE**

The ISAPI Filters tab now displays the jboss filter status and priority as Unknown because IIS has not yet received requests for the resource. The status and priority will change to Loaded and High respectively once a request is executed.

**Task: Define ISAPI Virtual Directory**

Complete this task to define the ISAPI virtual directory using the IIS management console.

1. Right click on Default Web Site, and then click New → Add Virtual Directory.
   
   The Add Virtual Directory window opens.

2. Specify the following values in the Add Virtual Directory window:
   
   **Alias:**
   
   Specify jboss (exactly as written).

   **Physical path:**
   
   Specify NATIVE\sbin\.

3. Click OK to save the values and close the Add Virtual Directory window.

4. In the tree view pane, expand Web Sites → Default Web Site.

5. Right click on the jboss virtual directory, and then click Properties.

6. Click the Virtual Directory tab, and make the following changes:
   
   **Execute Permissions:**
   
   Select Scripts and Executables.

   **Read check box:**
   
   Select to activate Read access.

7. Click OK to save the values and close the jboss Properties window.

**Task: Define ISAPI Web Service Extension**

Complete this task to define the ISAPI web service extension using the management console.

1. Click Web Service Extensions, and in the Tasks group select Add a new Web service extension.
   
   The New Web Service Extension window opens.

2. Add the following values to the New Web Service Extension window:
Extension name:
Specify jboss (exactly as written).

Required files:
Specify the path NATIVE\sbin\isapi_redirect.dll.

Set extension status to Allowed:
Select the check box.

3. Click OK to save the values and close the New Web Service Extension window.

4. Confirm the jboss Web Service Extension displays in the list.

14.4. MICROSOFT IIS 7 INITIAL CLUSTERING CONFIGURATION

Microsoft IIS 7 can be managed using the Management Console, or through the command prompt using the APPCMD.EXE command tool.

Terms

ISAPI and CGI Restrictions

ISAPI and CGI restrictions are request handlers that allow dynamic content to execute on a server. These restrictions are either CGI files (.exe) or ISAPI extensions (.dll). You can add custom ISAPI or CGI restrictions if the IIS configuration system allows this. [Configuring ISAPI and CGI Restrictions in IIS 7].

Task: Define a JBoss Native ISAPI Restriction

Complete this task to define an ISAPI Restriction using the management console.

1. On the master server, open IIS Manager:
   - Start → Run, then type inetmgr and hit Enter.
   - Start → Control Panel → Administrative Tools → Internet Information Services (IIS) Manager

   The IIS 7 Manager window opens.

2. In the tree view pane, select IIS 7 (referred to as Server Home).

   The IIS 7 Home Features View opens.

3. Double-click ISAPI and CGI Restrictions.

   The ISAPI and CGI Restrictions Features View opens.

4. In the Actions pane, click Add.

   The Add ISAPI or CGI Restriction window opens.

5. Specify the following values in the Add ISAPI or CGI Restriction window:
ISAPI or CGI Path:
   Specify \NATIVE\sbin\isapi_redirect.dll.

Description:
   Specify jboss (exactly as written).

Allow extension path to execute
   Select the check box.

6. Click OK to save the values, and close the Add ISAPI or CGI Restriction window.

NOTE
   The ISAPI and CGI Restrictions Features View now displays the jboss restriction and path.

Task: Define a JBoss Native Virtual Directory

Complete this task to define a virtual directory for the JBoss Native binary using the management console.

1. Right click on Default Web Site, and then click Add Virtual Directory.

   The Add Virtual Directory window opens.

2. Specify the following values in the Add Virtual Directory window:

   Alias:
      Specify jboss (exactly as written).

   Physical path:
      Specify \NATIVE\sbin\ .

3. Click OK to save the values and close the Add Virtual Directory window.

Task: Define a JBoss Native ISAPI Redirect Filter

Complete this task to define a JBoss Native ISAPI Redirect Filter using the management console.

1. In the tree view pane, expand Sites → Default Web Site.

2. Double-click ISAPI Filters.

   The ISAPI Filters Features View opens.

3. In the Actions pane, click Add.

   The Add ISAPI Filter window opens.

4. Specify the following values in the Add ISAPI Filter window:
Filter name:
  Specify `jboss` (exactly as written).

Executable:
  Specify `NATIVE\sbin\isapi_redirect.dll`.

5. Click **OK** to save the values and close the **Add ISAPI Filters** window.

**Task: Enable the ISAPI-dll Handler**

Complete this task to enable the ISAPI-dll handler using the management console.

1. In the tree view pane, select **IIS 7** (referred to as Server Home). 
   The IIS 7 Home Features View opens.

2. Double-click **Handler Mappings**.
   The **Handler Mappings** Features View opens.

3. In the **Group by** drop down box, select **State**.
   The Handler Mappings are displayed in Enabled and Disabled groups.

4. If **ISAPI-dll** is in the Disabled group, right mouse click and select **Edit Feature Permissions**.

5. Ensure the **Read**, **Script**, and **Execute** check boxes are selected.

6. Click **OK** to save the values and close the **Edit Feature Permissions** window.

14.5. **CONFIGURE A BASIC CLUSTER WITH ISAPI**

**Task: Configure ISAPI to serve a Basic Cluster**

Complete this task to configure ISAPI to manage applications common to all servers on a single IP address range, and route application requests to the correct server instance.

Use the configuration as an example when configuring your ISAPI cluster.

**NOTE**

This task does not provide instructions for load-balancing or server outage fail over. Refer to Section 14.6, “Configure a load-balancing cluster with ISAPI” for configuration instructions.

**Prerequisites**

- Complete the relevant Microsoft IIS clustering setup procedure. Refer to Section 14.3, “Microsoft IIS 6 initial clustering configuration” or Section 14.4, “Microsoft IIS 7 initial clustering configuration” for more information.
- The following steps assume that the `C:\connectors` directory is used to store logs,
properties files, and NSAPI locks.

1. **Create isapi_redirect.properties file**
   Create a new file named `isapi_redirect.properties` in the `NATIVE\sbin\` directory.

   **IMPORTANT**
   
   The `isapi_redirect.properties` file must be in the same directory as the `isapi_redirect.dll` file.

   Append the following configuration block to `isapi_redirect.properties`:

   ```
   # Configuration file for the ISAPI Redirector
   # Extension uri definition
   extension_uri=/jboss/isapi_redirect.dll

   # Full path to the log file for the ISAPI Redirector
   log_file=c:\connectors\isapi_redirect.log

   # Log level (debug, info, warn, error or trace)
   # Use debug only testing phase, for production switch to info
   log_level=debug

   # Full path to the workers.properties file
   worker_file=c:\connectors\workers.properties

   # Full path to the uriworkermap.properties file
   worker_mount_file=c:\connectors\uriworkermap.properties

   # Full path to the rewrite.properties file
   rewrite_rule_file=c:\connectors\rewrite.properties
   ```

2. **Optional: Create rewrite.properties file**
   The `rewrite.properties` file allows you to specify simple URL rewrites specific to an application. This configuration file is optional, and can be excluded from the `isapi_redirect.properties` file if URL rewrites are not required.

   The functionality offered is similar to Apache HTTP Server's mod_rewrite, but is less powerful. You specify the rewrite path using name-value pairs. A simple example is where the app_01 application has an abstract directory name, containing images, and you want to remap that directory to something more intuitive.

   ```
   #Simple example, images are accessible under abc path
   /app-01/abc/=/app-01/images/
   ```

3. **Create uriworkermap.properties file**
   The `uriworkermap.properties` file contains deployed application mapping configuration information. Append the following lines to the file.

   ```
   # images and css files for path /status are provided by worker01
   /status=worker01
   /images/*=worker01
   /css/*=worker01
   ```
Create workers.properties file

The `worker.properties` file contains mapping definitions between worker labels and server instances. Append the following lines to the file.

```ini
# An entry that lists all the workers defined
worker.list=worker01, worker02

# Entries that define the host and port associated with these workers

# First EAP server definition, port 8009 is standard port for AJP in EAP
worker.worker01.host=127.0.0.1
worker.worker01.port=8009
worker.worker01.type=ajp13

# Second EAP server definition
worker.worker02.host=127.0.0.100
worker.worker02.port=8009
worker.worker02.type=ajp13
```

5. **Restart IIS**

Restart your IIS server to implement the changes. Execute the following commands for the IIS version you are running:

**IIS 6**

```
C:\> net stop iisadmin /Y
C:\> net start w3svc
```

**IIS 7**

```
C:\> net stop was /Y
C:\> net start w3svc
```
6. Verify the Logs
Ensure you check the ISAPI logs once IIS has restarted. The logs are saved to the file location specified in the log_file property in isapi_redirect.properties. You should also check IIS logs and the Event Viewer for other events of type Warning or Error.

14.6. CONFIGURE A LOAD-BALANCING CLUSTER WITH ISAPI

Task: Configure ISAPI to serve a Load-Balancing Cluster

Complete this task to configure ISAPI to manage applications common to all servers, route requests to JBoss Enterprise Application Platform instances, and redirect requests to live nodes when some nodes are not online or experiencing connectivity issues.

Use the configuration as an example when configuring your ISAPI cluster.

Prerequisites

- Complete the relevant Microsoft IIS clustering setup procedure. Refer to Section 14.3, “Microsoft IIS 6 initial clustering configuration” or Section 14.4, “Microsoft IIS 7 initial clustering configuration” for more information.

- The following steps assume that the C:\connectors directory is used to store logs, properties files, and NSAPI locks.

1. Create isapi_redirect.properties file
Create a new file named isapi_redirect.properties in the NATIVE\sbin\ directory.

IMPORTANT

The isapi_redirect.properties file must be in the same directory as the isapi_redirect.dll file.

Append the following configuration block to the file:

```properties
# Configuration file for the ISAPI Redirector
# Extension uri definition
extension_uri=/jboss/isapi_redirect.dll

# Full path to the log file for the ISAPI Redirector
log_file=c:\connectors\isapi_redirect.log

# Log level (debug, info, warn, error or trace)
# Use debug only testing phase, for production switch to info
log_level=debug

# Full path to the workers.properties file
worker_file=c:\connectors\workers.properties

# Full path to the uriworkermap.properties file
worker_mount_file=c:\connectors\uriworkermap.properties

#OPTIONAL: Full path to the rewrite.properties file
rewrite_rule_file=c:\connectors\rewrite.properties
```
2. **Optional: Create rewrite.properties file**
The `rewrite.properties` file allows you to specify simple URL rewrites specific to an application. This configuration file is optional, and can be excluded from the `isapi_redirect.properties` file if URL rewrites are not required.

The functionality offered is similar to Apache HTTP Server's mod_rewrite, but is less powerful. You specify the rewrite path using name-value pairs. A simple example is where the app_01 application has an abstract directory name containing images, and you want to remap that directory to something more intuitive.

```plaintext
#Simple example, images are accessible under abc path
/app-01/abc/=/app-01/images/
```

3. **Create uriworkermap.properties file**
The `uriworkermap.properties` file contains deployed application mapping configuration information. Append the following lines to the file.

```plaintext
# images, css files, path /status and /web-console will provided by nodes defined in load-balancer
/css/*/=router
/images/*/=router
/status=router
/web-console|/*=router

# Example of exclusion from mapping, logo.gif will not be displayed
!/web-console/images/logo.gif=*  

# Requests to /app-01 and /app-02 will be routed to nodes defined in load-balancer
/app-01|/*=router
/app-02|/*=router

# mapping for management console, nodes in cluster can be enabled or disabled here
/jkmanager|/*=status
```

4. **Create workers.properties file**
The `workers.properties` file contains mapping definitions between worker labels and server instances. Append the following lines to the file.

```plaintext
# The advanced router LB worker
worker.list=router,status

# First EAP server definition, port 8009 is standard port for AJP in EAP
#  
# lbfactor defines how much the worker will be used.
# The higher the number, the more requests are served
# lbfactor is useful when one machine is more powerful
# ping_mode=A – all possible probes will be used to determine that
# connections are still working

worker.worker01.port=8009
worker.worker01.host=127.0.0.1
```
5. **Restart IIS**
   Restart your IIS server to implement the changes. Execute the following commands for the IIS version you are running:

   **IIS 6**
   
   C:\> net stop iisadmin /Y  
   C:\> net start w3svc

   **IIS 7**
   
   C:\> net stop was /Y  
   C:\> net start w3svc

6. **Verify the Logs**
   Ensure you check the ISAPI logs once IIS has restarted. The logs are saved to the file location specified in the log_file property in `isapi_redirect.properties`. You should also check IIS logs and the Event Viewer for other events of type Warning or Error.
Read this chapter to gain a basic understanding of the Netscape Server API (NSAPI).

NSAPI is a programming interface that allows developers to extend the functionality of web server software by creating applications (referred to as plug-ins) that run inside the server process itself.

The goal of NSAPI, and its plug-ins, is to provide a method of creating different functional interfaces between the web server and the back-end applications which run on it.

The NSAPI plug-ins are designed to implement Server Application Functions (SAFs). SAFs consume a HTTP request and take input from a server configuration database, and return a response to the client based on the inputs. Each SAF is linked to a particular class, which directly relates to the request-response step it helps implement.

The request-response steps (classes) are summarized in the following list:

1. Authorization translation;
2. Name translation;
3. Path checks;
4. Object type;
5. Request response;

You are not required to provide a SAF for each request-response step: NSAPI allows you to substitute your own custom functionality to the core request-response steps. You also have the choice of applying the SAF globally, or constraining the SAF to a directory, file group, or individual file.
CHAPTER 16. CONFIGURING THE NSAPI CONNECTOR ON SOLARIS

The following tasks describe how to configure the NSAPI connector to use a JBoss Enterprise Application Platform instance as a worker node for a Sun Java System Web Server (SJWS) master node. Sun Java System Web Server has been renamed to the Oracle iPlanet Web Server. The old name is used throughout this guide for clarity.

In this section, all of the server instances are on the same machine. To use different machines for each instance, use the -b switch to bind your instance of JBoss Enterprise Application Platform to a public IP address. Remember to edit the workers.properties file on the SJWS machine to reflect these changes in IP address.

16.1. PREREQUISITES AND CONFIGURATION ASSUMPTIONS

- Worker nodes are already installed with JBoss Enterprise Application Platform 5.2 or later. The Native components are not a requirement of the NSAPI connector. Refer to the Installation Guide for assistance with this prerequisite.

- The master node is already installed with one of the following technology combinations, and the appropriate Native binary for its operating system and architecture. Refer to the Installation Guide for assistance with this installation prerequisite.
  - Solaris 10 x86 with Sun Java System Web Server 7.0 U8
  - Solaris 10 SPARC 64 with Sun Java System Web Server 7.0 U8

16.2. CONFIGURE SERVER INSTANCE AS A WORKER NODE

Task: Configure a JBoss Enterprise Application Platform Worker Node

Follow this task to correctly configure a JBoss Enterprise Application Platform instance as a SJWS worker node.

Prerequisites

- Section 16.1, “Prerequisites and configuration assumptions”

1. Create a server profile for each worker node
   - Make a copy of the server profile that you wish to configure as a worker node. (This procedure uses the default server profile.)

   ```bash
   [user@workstation jboss-ep-5.2]$ cd jboss-as/server
   [user@workstation server]$ cp -r default/ default-01
   [user@workstation server]$ cp -r default/ default-02
   ```

2. Give each instance a unique name
   - Edit the following line in the deploy/jbossweb.sar/server.xml file of each new worker instance:

   ```xml
   <Engine name="jboss.web" defaultHost="localhost">
   ```
Add a unique `jvmRoute` value, as shown. This value is the identifier for this node in the cluster.

For the `default-01` server profile:

```xml
<Engine name="jboss.web" defaultHost="localhost"
jvmRoute="worker01">
```

For the `default-02` server profile:

```xml
<Engine name="jboss.web" defaultHost="localhost"
jvmRoute="worker02">
```

### 3. Enable session handling
Uncomment the following line in the `deployers/jbossweb.deployer/META-INF/war-deployers-jboss-beans.xml` file of each worker node:

```xml
<property name="useJK">false</property>
```

This property controls whether special session handling is used to coordinate with mod_jk and other connector variants. Set this property to `true` in both worker nodes:

```xml
<property name="useJK">true</property>
```

### 4. Start your worker nodes
Start each worker node in a separate command line interface. Ensure that each node is bound to a different IP address with the `-b` switch.

```
[user@workstation jboss-eap-5.2]$ ./jboss-as/bin/run.sh -b 127.0.0.1 -c default-01
```

```
[user@workstation jboss-eap-5.2]$ ./jboss-as/bin/run.sh -b 127.0.0.100 -c default-02
```

---

### 16.3. INITIAL CLUSTERING CONFIGURATION

#### Task: Configure Required Elements

Complete this task to configure the basic elements required for clustering using Sun Java Web Server (SJWS) and NSAPI.

#### Prerequisites

- **Task: Configure a JBoss Enterprise Application Platform Worker Node**

- Native zip archive extracted to the directory `/tmp/connectors/`. The directory `/tmp/connectors/jboss-ep-5.2/native/` is referred to as `NATIVE` in this procedure.

- The directory `/tmp/connectors` is used as the storage location for logs, properties files, and NSAPI locks.
SJWS is installed in one of the locations specified in the SJWS file path abbreviation in Section 1, “File Name Conventions”.

1. Disable servlet mappings

Under Built In Servlet Mappings in the SJWS/PROFILE/config/default-web.xml file, disable the mappings for the following servlets, by commenting them out as shown:

- default
- invoker
- jsp

```xml
<!-- The built-in servlet mappings defined above. -->
<servlet-mapping>
  <servlet-name>default</servlet-name>
  <url-pattern>/</url-pattern>
</servlet-mapping>

<servlet-mapping>
  <servlet-name>invoker</servlet-name>
  <url-pattern>/servlet/*</url-pattern>
</servlet-mapping>

<servlet-mapping>
  <servlet-name>jsp</servlet-name>
  <url-pattern>*.jsp</url-pattern>
</servlet-mapping>
```

2. Load the required modules and properties

Append the following lines to the SJWS/PROFILE/config/magnus.conf file:

```ini
Init fn="load-modules" func="jk_init,jk_service"
shlib="NATIVE/lib/nsapi_redirector.so" shlib_flags="(global|now)"
Init fn="jk_init" worker_file="/tmp/connectors/workers.properties"
log_level="debug" log_file="/tmp/connectors/nsapi.log"
shm_file="/tmp/connectors/jk_shm"
```

These lines define the location of the nsapi_redirector.so module used by the jk_init and jk_service functions, and the location of the workers.properties file, which defines the worker nodes and their attributes.

**NOTE**

The lib directory in the NATIVE/lib/nsapi_redirector.so path applies only to 32-bit machines. On 64-bit machines, this directory is called lib64.
16.4. CONFIGURE A BASIC CLUSTER WITH NSAPI

Task: Configure a Basic Cluster with NSAPI

Complete this task to configure a basic cluster, where requests for particular paths are forwarded to particular worker nodes. The procedure specifies that worker02 serves the /nc path, while worker01 serves /status and all other paths defined in the first part of the obj.conf file.

Prerequisites

- Task: Configure Required Elements

- SJWS is installed in one of the locations specified in the SJWS file path abbreviation in Section 1, “File Name Conventions”.

1. Define the paths to serve via NSAPI

   Edit the SJWS/PROFILE/config/obj.conf file. Define paths that should be served via NSAPI at the end of the default Object definition, as shown:

   ```
   <Object name="default">
      [...]
      NameTrans fn="assign-name" from="/status" name="jknsapi"
      NameTrans fn="assign-name" from="/images(|/*)" name="jknsapi"
      NameTrans fn="assign-name" from="/css(|/*)" name="jknsapi"
      NameTrans fn="assign-name" from="/nc(|/*)" name="jknsapi"
      NameTrans fn="assign-name" from="/jmx-console(|/*)"
         name="jknsapi"
   </Object>
   ```

   You can map the path of any application deployed on your JBoss Enterprise Platform instance in this obj.conf file. In the example code, the /nc path is mapped to an application deployed under the name nc.

2. Define the worker that serves each path

   Edit the SJWS/PROFILE/config/obj.conf file and add the following jknsapi Object definition after the default Object definition.

   ```
   <Object name="jknsapi">
      ObjectType fn=force-type type=text/plain
      Service fn="jk_service" worker="worker01" path="/status"
      Service fn="jk_service" worker="worker02" path="/nc(|*)"
      Service fn="jk_service" worker="worker01"
   </Object>
   ```

   This jknsapi Object defines the worker nodes used to serve each path that was assigned to name="jknsapi" in the default Object.

   In the example code, the third Service definition does not specify a path value, so the worker node defined (worker01) serves all of the paths assigned to jknsapi by default. In this case, the first Service definition in the example code, which assigns the /status path to worker01, is superfluous.

3. Define the workers and their attributes

   Create a workers.properties file in the location you defined in Step 2.
Define the list of worker nodes and each worker node's properties in this file:

```plaintext
# An entry that lists all the workers defined
worker.list=worker01, worker02

# Entries that define the host and port associated with these workers
worker.worker01.host=127.0.0.1
worker.worker01.port=8009
worker.worker01.type=ajp13

worker.worker02.host=127.0.0.100
worker.worker02.port=8009
worker.worker02.type=ajp13
```

4. **Restart the server**
   Once your Sun Java System Web Server instance is configured, restart it so that your changes take effect.

   For Sun Java System Web Server 6.1:
   ```
   SJWS/PROFILE/stop
   SJWS/PROFILE/start
   ```

   For Sun Java System Web Server 7.0:
   ```
   SJWS/PROFILE/bin/stopserv
   SJWS/PROFILE/bin/startserv
   ```

### 16.5. CONFIGURE A LOAD-BALANCED CLUSTER WITH NSAPI

**Task: Configure a Load-balanced Cluster with NSAPI**

Complete this task to configure a load-balanced cluster consisting of two worker nodes.

**Prerequisites**

- **Task: Configure Required Elements**

  - SJWS is installed in one of the locations specified in the `SJWS` file path abbreviation in Section 1, “File Name Conventions”.

1. **Define the paths to serve via NSAPI**
   Open `SJWS/PROFILE/config/obj.conf` and define paths that should be served through NSAPI at the end of the `default` Object definition, as shown:

   ```xml
   <Object name="default">
     [...]
     NameTrans fn="assign-name" from="/status" name="jknsapi"
     NameTrans fn="assign-name" from="/images(|/*)" name="jknsapi"
     NameTrans fn="assign-name" from="/css(|/*)" name="jknsapi"
     NameTrans fn="assign-name" from="/nc(|/*)" name="jknsapi"
     NameTrans fn="assign-name" from="/jmx-console(|/*)"
   </Object>
   ```
You can map the path of any application deployed on your JBoss Enterprise Platform instance in this obj.conf file. In the example code, the /nc path is mapped to an application deployed under the name nc.

2. Define the worker that serves each path

Open SJWS/PROFILE/config/obj.conf and add the following jknsapi Object definition after the default Object definition.

```xml
<Object name="jknsapi">
  ObjectType fn=force-type type=text/plain
  Service fn="jk_service" worker="status" path="/jkmanager/*"
  Service fn="jk_service" worker="router"
</Object>
```

This jknsapi Object defines the worker nodes used to serve each path that was assigned to name="jknsapi" in the default Object.

3. Define the workers and their attributes

Create SJWS/PROFILE/config/workers.properties.

Define the list of worker nodes and each worker node's properties in this file:

```
# The advanced router LB worker
worker.list=router,status

#First EAP server definition, port 8009 is standard port for AJP in EAP
#lbfactor defines how much the worker will be used.
#The higher the number, the more requests are served
#lbfactor is useful when one machine is more powerful
#ping_mode=A – all possible probes will be used to determine that
#connections are still working
worker.worker01.port=8009
worker.worker01.host=127.0.0.1
worker.worker01.type=ajp13
worker.worker01.ping_mode=A
worker.worker01.socket_timeout=10
worker.worker01.lbfactor=3

#Second EAP server definition
worker.worker02.port=8009
worker.worker02.host=127.0.0.100
worker.worker02.type=ajp13
worker.worker02.ping_mode=A
```

NOTE

For an explanation of workers.properties directives, refer to Appendix A, Reference: workers.properties
worker.worker02.socket_timeout=10
worker.worker02.lb_factor=1

# Define the LB worker
worker.router.type=lb
worker.router.balance_workers=worker01,worker02

# Define the status worker
worker.status.type=status

4. **Restart the server**
   Once your Sun Java System Web Server instance is configured, restart it so that your changes take effect.

For Sun Java System Web Server 6.1:

    SJWS/PROFILE/stop
    SJWS/PROFILE/start

For Sun Java System Web Server 7.0:

    SJWS/PROFILE/bin/stopserv
    SJWS/PROFILE/bin/startserv
PART V. COMMON LOAD BALANCING TASKS
CHAPTER 17. HTTP SESSION STATE REPLICATION

HTTP session state replication is a means of distributing clients’ state across multiple servers. The following terms are important in understanding load balancing. Refer to the Clustering Guide part of the Administration and Configuration Guide for details of associated JBoss Enterprise Application Platform high availability options.

Software Load Balancer

A dedicated software-based service designed to distribute HTTP client session requests across multiple computer servers (cluster). The primary directive of a software load balancer is to maximize resource utilization, reduce request response times, and prevent server overload. The load balancer forwards client session requests to a server cluster, based on server load and availability.

Client Session

A semi-permanent connection between the client (an application) and the server. The load balancer determines whether the client session is created with persistence, or whether a client session is redistributed based on server load and availability.

Session Persistence

Session persistence is a feature where information about a client's session is stored by the server so that if the client's connection is interrupted temporarily, the session can continue at the time the connection error occurred. A persistent session is also commonly known as a sticky session.

Sticky Session

See Session Persistence.

Section 3.1, “Configure worker nodes in mod_jk” describes how to configure session state persistence in the load balancer to ensure a client in a session is always routed to the same server node.

Session persistence on its own is not a best-practice solution because if a server fails, all session state data is lost. For example, if a customer is about to make a purchase on a web site, and the server hosting the shopping cart instance fails, session state data associated with the cart is lost permanently.

One way of preventing client session data loss is to replicate session data across the servers in the cluster. If a server node fails or is shut down, the load balancer can fail over the next client request to any server node and obtain the same session state.

Using a load balancer that supports session persistence, but not configuring web applications for session replication, allows you to scale your implementation by avoiding the cost of session state replication: each request for a session will always be handled by the same node.

Session state replication is more expensive than basic session persistence, but the reliability it provides for session state data makes it important when creating a load balanced cluster.

17.1. ENABLING SESSION REPLICATION IN YOUR APPLICATION

To enable replication of your web application you must tag the application as distributable in the web.xml descriptor. Here's an example:

```xml
<?xml version="1.0"?>
<web-app
  xmlns="http://java.sun.com/xml/ns/j2ee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee"/>
```

HTTP Connectors Load Balancing Guide
You can further configure session replication using the **replication-config** element in the `jboss-web.xml` file. However, the **replication-config** element only needs to be set if one or more of the default values described below is unacceptable. All of the configuration elements are optional, and can be omitted if the default value is acceptable.

Here is an example:

```xml
<jboss-web>
    <replication-config>
        <cache-name>custom-session-cache</cache-name>
        <replication-trigger>SET</replication-trigger>
        <replication-granularity>ATTRIBUTE</replication-granularity>
        <replication-field-batch-mode>true</replication-field-batch-mode>
        <use-jk>false</use-jk>
        <max-unreplicated-interval>30</max-unreplicated-interval>
        <snapshot-mode>INSTANT</snapshot-mode>
        <snapshot-interval>1000</snapshot-interval>
        <session-notification-policy>com.example.CustomSessionNotificationPolicy</session-notification-policy>
    </replication-config>
</jboss-web>

**<replication-trigger>**

The **<replication-trigger>** element determines when the container should consider that session data must be replicated across the cluster. The rationale for this setting is that after a mutable object stored as a session attribute is accessed from the session, in the absence of a `setAttribute` call, the container has no clear way to know if the object (and hence the session state) has been modified and needs to be replicated. This element has 3 valid values:

- **SET** is conservative but not optimal (performance-wise): it will always replicate session data even if its content has only been accessed and not modified. This setting made (a little) sense in JBoss Enterprise Application Platform 4 since using it was a way to ensure that every request triggered replication of the session's timestamp. Since setting `max_unreplicated_interval` to 0 accomplishes the same thing at much lower cost, using **SET** makes no sense with JBoss Enterprise Application Platform 5 or JBoss Enterprise Web Platform 5.

- **SET_AND_NON_PRIMITIVE_GET** is conservative but will only replicate if an object of a non-primitive type has been accessed (in effect, the object is not of a well-known immutable JDK type such as `Integer`, `Long`, `String`, etc.) This is the default value.
SET assumes that the developer will explicitly call `setAttribute` on the session if the data needs to be replicated. This setting prevents unnecessary replication and can have a major beneficial impact on performance, but requires very good coding practices to ensure `setAttribute` is always called whenever a mutable object stored in the session is modified.

In all cases, calling `setAttribute` marks the session as needing replication.

```
<cacheName>
Specifies the name of the JBoss Cache configuration that should be used for storing distributable sessions and replicating them around the cluster. This element lets web applications that require different caching characteristics specify the use of separate, differently configured, JBoss Cache instances. In JBoss Enterprise Application Platform 4 the cache to use was a server-wide configuration that could not be changed per web application. The default value is `standard-session-cache`. See Section 17.3, “Configure the JBoss Cache instance used for session state replication” for more details on JBoss Cache configuration for web-tier clustering.

<replication-field-batch-mode>
Specifies whether all replication messages associated with a request will be batched into one message. This is applicable only if `replication-granularity` is `FIELD`. If `replication-field-batch-mode` is set to `true`, fine-grained changes made to objects stored in the session attribute map will replicate only when the HTTP request is finished; otherwise they replicate as they occur. Setting this to `false` is not advised because it increases the number of replication requests and the chance of session state being out of sync. Default is `true`.

**IMPORTANT**

The `FIELD` granularity option is now deprecated as JBoss Cache, which provides this feature, is to be substituted by Infinispan (Infinispan does not support this granularity).

<useJK>
Specifies whether the container should assume that a JK-based software load balancer (for example,. `mod_jk`, `mod_proxy`, `mod_cluster`) is being used for load balancing for this web application. If set to `true`, the container will examine the session ID associated with every request and replace the `jvmRoute` portion of the session ID if it detects a failover.

You need only set this to `false` for web applications whose URL cannot be handled by the JK load balancer.

<max-unreplicated-interval>
Specifies the maximum interval between requests, in seconds, after which a request will trigger replication of the session's timestamp regardless of whether the request has otherwise made the session dirty. Such replication ensures that other nodes in the cluster are aware of the most recent value for the session's timestamp and will not incorrectly expire an unreplicated session upon failover. It also results in correct values for `HttpSession.getLastAccessedTime()` calls following failover.

The default value is `null` (in effect, unspecified). In this case the session manager will use the presence or absence of a `jvmRoute` configuration on its enclosing JBoss Web Engine (see Section 3.2, “Configuring JBoss to work with mod_jk”) to determine whether JK is used.
A value of 0 means the timestamp will be replicated whenever the session is accessed. A value of 1 means the timestamp will be replicated only if some other activity during the request (for example, modifying an attribute) has resulted in other replication work involving the session. A positive value greater than the HttpSession.getMaxInactiveInterval() value will be treated as probable misconfiguration and converted to 0; (in effect, replicate the metadata on every request). Default value is 60.

<snapshot-mode>
Specifications when sessions are replicated to the other nodes. Possible values are INSTANT (the default) and INTERVAL.

The typical value, INSTANT, replicates changes to the other nodes at the end of requests, using the request processing thread to perform the replication. In this case, the snapshot-interval property is ignored.

With INTERVAL mode, a background task is created that runs every snapshot-interval milliseconds, checking for modified sessions and replicating them.

Note that this property has no effect if replication-granularity is set to FIELD. If it is FIELD, INSTANT mode will be used.

<snapshot-interval>
Specifies how often (in milliseconds) the background task that replicates modified sessions should be started for this web application. Only meaningful if snapshot-mode is set to INTERVAL.

<session-notification-policy>
Specifies the fully qualified class name of the implementation of the ClusteredSessionNotificationPolicy interface that should be used to govern whether servlet specification notifications should be emitted to any registered HttpSessionListener, HttpSessionAttributeListener and/or HttpSessionBindingListener.

IMPORTANT

Event notifications that may be appropriate in non-clustered environment may not necessarily be appropriate in a clustered environment; see https://jira.jboss.org/jira/browse/JBAS-5778 for an example of why a notification may not be desired. Configuring an appropriate ClusteredSessionNotificationPolicy gives the application author fine-grained control over what notifications are issued.

17.2. HTTPSESSION PASSIVATION AND ACTIVATION

Passivation

The process of controlling memory usage by removing relatively unused sessions from memory while storing them in persistent storage.

If a passivated session is requested by a client, it can be "activated" back into memory and removed from the persistent store. JBoss Enterprise Application Platform 5 supports HttpSession passivation from clustered web applications where the web.xml file includes the distributable directive.
Passivation occurs at three points during the life cycle of a web application:

- When the container requests the creation of a new session. If the number of currently active sessions exceeds a configurable limit, an attempt is made to passivate sessions to make room in memory.

- Periodically (by default every ten seconds) as the JBoss Web background task thread runs.

- When the web application is deployed and a backup copy of sessions active on other servers is acquired by the newly deploying web application’s session manager.

A session is passivated if one of the following conditions is true:

- The session has not been in use for longer than a configurable maximum idle time.

- The number of active sessions exceeds a configurable maximum and the session has not been in use for longer than a configurable minimum idle time.

In both cases, sessions are passivated on a Least Recently Used (LRU) basis.

### 17.2.1. Configuring HttpSession passivation

Session passivation behavior is configured in the `jboss-web.xml` deployment descriptor in your web application's `WEB-INF` directory.

```xml
<max-active-sessions>20</max-active-sessions>
<passivation-config>
  <use-session-passivation>true</use-session-passivation>
  <passivation-min-idle-time>60</passivation-min-idle-time>
  <passivation-max-idle-time>600</passivation-max-idle-time>
</passivation-config>
```

- **max-active-sessions**
  
  Determines the maximum number of active sessions allowed. If the number of sessions managed by the session manager exceeds this value and passivation is enabled, the excess will be passivated based on the configured `passivation-min-idle-time`. If after passivation is completed (or if passivation is disabled), the number of active sessions still exceeds this limit, attempts to create new sessions will be rejected. If set to `-1` (the default), there is no limit.

The total number of sessions in memory includes sessions replicated from other cluster nodes that are not being accessed on this node. Take this into account when setting `max-active-sessions`. Whether or not *buddy replication* is enabled will also affect the number of sessions replicated from other nodes.

Say, for example, that you have an eight node cluster, and each node handles requests from 100 users. With *total replication*, each node would store 800 sessions in memory. With *buddy*
replication enabled, and the default numBuddies setting (1), each node will store 200 sessions in memory.

- **use-session-passivation**
  
  Determines whether session passivation will be enabled for the web application. Default is false.

- **passivation-min-idle-time**
  
  Determines the minimum time (in seconds) that a session must have been inactive before the container will consider passivating it in order to reduce the active session count to obey the value defined by max-active-sessions. A value of -1 (the default) disables passivating sessions before passivation-max-idle-time. Neither a value of -1 nor a high value are recommended if max-active-sessions is set.

- **passivation-max-idle-time**
  
  Determines the maximum time (in seconds) that a session can be inactive before the container should attempt to passivate it to save memory. Passivation of such sessions will take place regardless of whether the active session count exceeds max-active-sessions. Should be less than the session-timeout setting in web.xml. A value of -1 (the default) disables passivation based on maximum inactivity.

### 17.3. CONFIGURE THE JBOSS CACHE INSTANCE USED FOR SESSION STATE REPLICATION

The container for a distributable web application makes use of JBoss Cache to provide HTTP session replication services around the cluster. It integrates with the CacheManager service to obtain a reference to a JBoss Cache instance. For more information, refer to the Distributed Caching with JBoss Cache and JBoss Cache Configuration and Deployment chapters in the Administration and Configuration Guide.

The name of the JBoss Cache configuration to use is controlled by the cacheName element in the application's jboss-web.xml (see Section 17.1, “Enabling session replication in your application”). In most cases this does not need to be set because the default value of standard-session-cache is appropriate.

The JBoss Cache configurations in the CacheManager service expose a number of options. The standard-session-cache configuration is already optimized for the web session replication use case, and most of the settings should not be altered. Administrators may be interested in altering the following settings:

- **cacheMode**
  
  The default is REPL_ASYNC, which specifies that a session replication message sent to the cluster does not wait for responses from other cluster nodes confirming that the message has been received and processed. The alternative mode, REPL_SYNC, offers a greater degree of confirmation that session state has been received, but reduces performance significantly.

- **enabled** property in the buddyReplicationConfig section

  Set to true to enable buddy replication. Default is false.
- **numBuddies** property in the **buddyReplicationConfig** section

  Set to a value greater than the default (1) to increase the number of backup nodes onto which sessions are replicated. Only relevant if buddy replication is enabled.

- **buddyPoolName** property in the **buddyReplicationConfig** section

  A way to specify a preferred replication group when buddy replication is enabled. JBoss Cache tries to pick a buddy who shares the same pool name (falling back to other buddies if not available). Only relevant if buddy replication is enabled.

- **multiplexerStack**

  Name of the JGroups protocol stack the cache should use.

- **clusterName**

  Identifying name JGroups will use for this cache's channel. Only change this if you create a new cache configuration, in which case this property should have a different value from all other cache configurations.

If you wish to use a completely new JBoss Cache configuration rather than editing one of the existing ones, refer to *Deployment via the CacheManager Service* section in the *Administration and Configuration Guide*. 
CHAPTER 18. HIGH-AVAILABILITY WEB SESSIONS

JBoss Enterprise Application Server allows you to make web sessions highly available by storing them in a database table.

IMPORTANT

HTTP session replication with JBoss Cache is the preferred approach to securing web session failover. It is strongly recommended to use this approach if possible (refer to Chapter 17, HTTP session state replication).

To provide high availability web sessions, you can configure JBoss Application Server to store the web session state in a database table. If the server then becomes unavailable, the web session state is still preserved in the database table and can be used by failover servers, while if using session replication, the web session is available on the server and the respective failover nodes. The high availability web session setup can be useful in a WAN with several application server instance or in combination with session replication.

To make web sessions highly available, you need to do the following:

- configure the server to use the session manager set on your web application (JBoss Application Server by default ignores the web application session manager and switches to JBossCacheManager automatically);
- configure your web applications to use DataSourcePersistentManager as their session manager (the manager handles the storing of web sessions to the defined database table);
- create the web session table in the target database and deploy the datasource, which will provide the connection between the session manager and the database table.

Configuring JBoss Enterprise Application Server

To configure JBoss Enterprise Application Server to allow storing of sessions in a database, disable overriding of the session manager set on your web application (overridden to JBossCacheManager; this allows the web application to use its own session manager):

1. Open the JBOSS_HOME/server/PROFILE/deployers/jbossweb.deployer/META-INF/war-deployers-jboss-beans.xml file for editing.

2. Set the overrideDistributableManager property of the WarDeployer bean to false:

```xml
<bean name="WarDeployer"
class="org.jboss.web.tomcat.service.deployers.TomcatDeployer">
  
  <!-- "False" disables overriding the session manager for distributable webapps -->
  <property name="overrideDistributableManager">false</property>
</bean>
```

Configuring Web Application
Configure your web application to use the database persistent session manager:

1. In the application's **WEB-INF** directory, create the **context.xml** file, which will define what session manager is to be used as well as the manager's attributes.

   **IMPORTANT**

   Note that it is not recommended to add the manager definition to the **jboss-web.xml** file, although it is the standard web application deployment descriptor. The goal of using the **context.xml** file instead is to avoid any unnecessary changes to the existing JBoss Application Server code.

2. In the **context.xml** file, add the **Manager** element and its attributes:

   **className**
   
   fully-qualified class name of the session manager implementation

   **dataSourceJndiName**
   
   datasource that allows the session manager to communicate with the database that stores the web sessions

   ```xml
   <Context cookies="true" crossContext="true">
     <Manager className="org.jboss.web.tomcat.service.session.persistent.DataSourcePersistentManager"
              dataSourceJndiName="java:HttpSessionDS"/>
   </Context>
   ``

   **NOTE**

   The className and dataSourceJndiName are compulsory attributes. You can also define further Context and Manager attributes (refer to Section 18.1, "DataSourcePersistentManager Configuration Attributes").

**Configuring Database and Datasource**

Create the database session table, which will hold the web session data and then create the respective datasource:

1. Create the web session (**httpsessions**) database table:

   You can change the name of the table and of the columns; however, make sure to configure the DataSourcePersistentManager attributes appropriately.

   Individual columns must be able to store values of particular datatypes:

   - creationtime and lastaccess: java long values
   - maxinactive and version: java int value
- metadata: serialized java objects (currently not used)
- attributes: serialized java objects (stores the session attributes map; should be large enough to store your largest sessions)
- primary key: synthetic primary key (optional, make sure there is a UNIQUE INDEX on app + id).

The following command creates the table with default settings in the most common databases (MySQL, IBM DB2, Oracle Database):

```sql
CREATE TABLE httpsessions (app VARCHAR(255) NOT NULL, id VARCHAR(255) NOT NULL,
fullId VARCHAR(255) NOT NULL, creationtime BIGINT NOT NULL,
maxinactive BIGINT NOT NULL, version INT NOT NULL, lastaccess BIGINT NOT NULL,
isnew CHAR(1) NOT NULL, valid CHAR(1) NOT NULL, metadata VARBINARY NULL,
attributes LONGVARBINARY NOT NULL,
CONSTRAINT app_id PRIMARY KEY (app, id))
```

2. Deploy an appropriate datasource to allow the DataSourcePersistentManager to communicate with the database (refer to the chapter on Datasource Configuration in the Administration and Configuration Guide. Make sure the datasource is set up as local-tx-datasource (xa-datascources are not supported).

3. Add the dataSourceJndiName with the jndi-name of the created datasource to DataSourcePersistentManager element in the context.xml file.

### 18.1. DATASOURCEPERSISTENTMANAGER CONFIGURATION ATTRIBUTES

The DataSourcePersistentManager element must define the className and dataSourceJndiName attributes. Apart from these, it can define other properties to specify manager's behavior and the way it interacts with the database.

**Compulsory Properties**

**className**

fully-qualified class name of the org.apache.catalina.Manager implementation (that is, org.jboss.web.tomcat.service.session.persistent.DataSourcePersistentManager)

**dataSourceJndiName**

JNDI name of the data source, which defines the database connection to the httpsessions

**Properties Defining the Database Connection Properties**

**connectionName**

value of the username parameter to pass to the `DataSource.getConnection` method (if null, the getConnection with no arguments is called)
connectionPassword
password to pass to DataSource.getConnection

sessionTable
name of the database session table in which sessions are stored (by default httpsessions)

sessionAppCol
name of the column with the web application name associated with the session (by default app)

sessionIdCol
name of the column with the core, immutable part of a session ID (by default id; this and the sessionAppCol columns form the unique index for the table)

sessionFullIdCol
name of the column with the full session ID (including any mutable element added to the core session ID, for example a jvmRoute; by default fullid)

sessionCreationTimeCol
name of the column with the time when the session was created (of the long datatype, by default creationtime)

sessionMaxInactiveCol
name of the column with the maximum number of milliseconds the session can remain unaccessed before expiring (by default maxinactive)

sessionVersionCol
name of the column which stores the session's "version" (session version is incremented each time the session is persisted; by default version)

sessionLastAccessedCol
name of the column with the timestamp of the last session access (take the long data type value; by default lastaccess)

sessionNewCol
Name of the column with the flag indicating whether the session is new (session is considered new if it was not yet joined by the client; by default isNew)

sessionValidCol
name of the column with the flag indicating whether the session is valid (by default isValid)

sessionMetadataCol
name of the column which can store serialized metadata about the session (currently unused; by default metadata)

sessionAttributeCol
name of the column with the serialized session attribute map (by default attributes)
Properties Defining the Manager's Behavior

**cleanupInterval**
- minimum period of time in seconds that must lapse from the last cleaning of old "abandoned" sessions from the database before the next cleaning (by default 14400 seconds, that is, 4 hours)
- A session is abandoned if the only server that was handling the session requests was shut down before the session expired, no further requests for the session were received so that the session did not fail over to another server. Such session do not expire on the normal session expiration checks. Therefore a special process runs periodically to clean the session from the database.

**replicationTriggerString**
- activity executed during a request that makes the session database persistent (the activity with the replication-trigger property SET)

**maxUnreplicatedInterval**
- maximum interval between requests, in seconds, after which the session's timestamp is persisted regardless of whether the request caused the session to become dirty in any other way

**useJK**
- flag determining whether the container assumes a JK-based software load balancer is used for load balancing (if set to true, the container examines the session ID associated with every request and replace the jvmRoute portion of the session ID if it detects a failover)

**maxActiveAllowed**
- maximum number of active sessions

**useSessionPassivation**
- flag for enabling/disabling session passivation (session is removed from memory but remains always in the persistent store)

**passivationMinIdleTime**
- minimum time, in seconds, a session must be inactive before passivated

**passivationMaxIdleTime**
- maximum time, in seconds, a session can be inactive before passivated

**processExpiresFrequency**
- frequency at which the background process thread calls the session manager to perform background processes (for example, expire or passivate sessions; (by default, every 10 seconds)
- This configuration is defined as value N with the background cleanup process called 1 in N callings to the session manager. Default is 1, that is, the cleanup process is performed every time the manager is called by the background process, that is cleanup is performed every 10 seconds. For example, if set to 6, the manager performs the cleanup once a minute (1/6, that is once in 60 seconds).

**sessionNotificationPolicy**
fully qualified class name of the implementation of the ClusteredSessionNotificationPolicy interface that is used to govern whether servlet specification notifications is emitted to any registered HttpSessionListener, HttpSessionAttributeListener or HttpSessionBindingListener.
CHAPTER 19. USING CLUSTERED SINGLE SIGN-ON (SSO)

JBoss supports clustered single sign-on (SSO), allowing a user to authenticate to one web application and to be recognized on all web applications that are deployed on the same virtual host, regardless of whether they are deployed on that same machine or on another node in the cluster.

Authentication replication is handled by JBoss Cache. Clustered single sign-on support is a JBoss-specific extension of the non-clustered org.apache.catalina.authenticator.SingleSignOn valve that is a standard part of Tomcat and JBoss Web.

19.1. CONFIGURATION

To enable clustered single sign-on, you must add the ClusteredSingleSignOn valve to the appropriate Host elements of the
JBoss_HOME/server/PROFILE/deploy/jbossweb.sar/server.xml file. The valve element is already included in the standard file; you just need to uncomment it. The valve configuration is shown here:

Valve className="org.jboss.web.tomcat.service.sso.ClusteredSingleSignOn" /

The element supports the following attributes:

- **class Name** is a required attribute to set the Java class name of the valve implementation to use. This must be set to org.jboss.web.tomcat.service.sso.ClusteredSingleSign.

- **cacheConfig** is the name of the cache configuration to use for the clustered SSO cache. Default is clustered-sso.

  NOTE

  For more information about cache configuration, refer to The JBoss Enterprise Application Platform CacheManager Service section in the Administration and Configuration Guide.

- **treeCacheName** is deprecated; use cacheConfig. Specifies a JMX ObjectName of the JBoss Cache MBean to use for the clustered SSO cache. If no cache can be located from the CacheManager service using the value of cacheConfig, an attempt to locate an MBean registered in JMX under this ObjectName will be made. Default value is jboss.cache:service=TomcatClusteringCache.

- **cookieDomain** is used to set the host domain to be used for SSO cookies. See Section 19.4, “Configuring the cookie domain” for more. Default is "/".

- **maxEmptyLife** is the maximum number of seconds an SSO with no active sessions will be usable by a request. The clustered SSO valve tracks which cluster nodes are managing sessions related to an SSO. When a node is shutdown, all local copies of a session are invalidated. If a further user request is made within the time specified by maxEmptyLife, the request will fail over to another cluster node, activating the backup copy of the session. If maxEmptyLife is set to 0, the SSO valve terminates together with the local session copies. Default is 1800, (30 minutes).

- **processExpiresInterval** is the minimum number of seconds between efforts by the valve to find and invalidate SSOs that have exceeded their maxEmptyLife. Does not imply effort will
be spent on such cleanup every `processExpiresInterval`, just that it will not occur more frequently than that. Default is 60.

- `requireReauthentication` is a flag to determine whether each request needs to be reauthenticated to the security `Realm`. If `true`, this valve uses cached security credentials (username and password) to reauthenticate to the JBoss Web security `Realm` for each request associated with an SSO session. If `false`, the valve can itself authenticate requests based on the presence of a valid SSO cookie, without rechecking with the `Realm`. Setting to `true` can allow web applications with different `security-domain` configurations to share an SSO. Default is `false`.

### 19.2. SSO BEHAVIOR

The user will not be challenged as long as they access only unprotected resources in any of the web applications on the virtual host.

Upon access to a protected resource in any web app, the user will be challenged to authenticate, using the log in method defined for the web app.

Once authenticated, the roles associated with this user will be utilized for access control decisions across all of the associated web applications, without challenging the user to authenticate themselves to each application individually.

If the web application invalidates a session (by invoking the `javax.servlet.http.HttpServletRequest.invalidate()` method), the user's sessions in all web applications will be invalidated.

A session timeout does not invalidate the SSO if other sessions are still valid.

### 19.3. LIMITATIONS

There are a number of known limitations to this Tomcat valve-based SSO implementation:

- Only useful within a cluster of EAP instances; SSO does not propagate to other resources.
- Requires use of container-managed authentication (via `login-config` element in `web.xml`).
- Requires cookies. SSO is maintained via a cookie and URL rewriting is not supported.
- Unless `requireReauthentication` is set to `true`, all web applications configured for the same SSO valve must share the same JBoss Web `Realm` and JBoss Security `security-domain`. This means:
  - In `server.xml` you can nest the `Realm` element inside the `Host` element (or the surrounding `Engine` element), but not inside a `context.xml` packaged with one of the involved web applications.
  - The `security-domain` configured in `jboss-web.xml` or `jboss-app.xml` must be consistent for all of the web applications.
  - Even if you set `requireReauthentication` to `true` and use a different `security-domain` (or, less likely, a different `Realm`) for different webapps, the varying security integrations must all accept the same credentials (for example, username and password).
19.4. CONFIGURING THE COOKIE DOMAIN

The SSO valve supports a cookieDomain configuration attribute. This attribute allows configuration of the SSO cookie's domain (the set of hosts to which the browser will present the cookie). By default the domain is "/", meaning the browser will only present the cookie to the host that issued it. The cookieDomain attribute allows the cookie to be scoped to a wider domain.

For example, suppose we have a case where two apps, with URLs http://app1.xyz.com and http://app2.xyz.com, that wish to share an SSO context. These apps could be running on different servers in a cluster or the virtual host with which they are associated could have multiple aliases. This can be supported with the following configuration:

```java
Valve className="org.jboss.web.tomcat.service.sso.ClusteredSingleSignOn"
   cookieDomain="xyz.com" /
```
CHAPTER 20. COMPLETE WORKING EXAMPLE

Following are a set of example configuration files for a complete working example.

Proxy Server

A proxy server listening on localhost:

LoadModule slotmem_module modules/mod_slotmem.so
LoadModule manager_module modules/mod_manager.so
LoadModule proxy_cluster_module modules/mod_proxy_cluster.so
LoadModule advertise_module modules/mod_advertise.so

Listen 127.0.0.1:6666
<VirtualHost 127.0.0.1:6666>
  <Directory />
    Order deny,allow
    Deny from all
    Allow from 127.0.0.1
  </Directory>
  KeepAliveTimeout 60
  MaxKeepAliveRequests 0
  ManagerBalancerName mycluster
  ServerAdvertise On
  AdvertiseFrequency 5
</VirtualHost>

<Location /mod_cluster-manager>
  SetHandler mod_cluster-manager
  Order deny,allow
  Deny from all
  Allow from 127.0.0.1
</Location>

JBoss Web Client Listener

Following are the listener definitions for
JBOSS_EAP_DIST/server/PROFILE/deploy/jbossweb.sar/server.xml.

<!-- Non-clustered mode -->
<Listener
  className="org.jboss.web.tomcat.service.deployers.MicrocontainerIntegrationLifecycleListener"
  delegateBeanName="ModClusterService"/>

<!-- Clustered mode -->
<Listener
  className="org.jboss.web.tomcat.service.deployers.MicrocontainerIntegrationLifecycleListener"
  delegateBeanName="HAModClusterService"/>
Following are the required dependencies for the WebServer bean in  
`JBOSS_EAP_DIST/server/PROFILE/deploy/jbossweb.sar/META-INF/jboss-beans.xml`.  
Add them to the existing dependencies.

```xml
<bean name="WebServer"  
class="org.jboss.web.tomcat.service.deployers.TomcatService">  
  <!-- ... -->  
  <depends>ModClusterService</depends><!-- Non-clustered mode -->  
  <!--depends>HAModClusterService</depends--><!-- Clustered mode -->  
  <!-- ... -->  
</bean>
```

**Example iptables Firewall Rules**

Following are a set of example firewall rules using `iptables`, for a cluster node on the 192.168.1.0/24 subnet.

```
/sbin/iptables -I INPUT 5 -p udp -d 224.0.1.0/24 -j ACCEPT -m comment --comment "mod_cluster traffic"  
/sbin/iptables -I INPUT 6 -p udp -d 224.0.0.0/4 -j ACCEPT -m comment --comment "JBoss Cluster traffic"  
/sbin/iptables -I INPUT 9 -p udp -s 192.168.1.0/24 -j ACCEPT -m comment --comment "cluster subnet for inter-node communication"  
/sbin/iptables -I INPUT 10 -p tcp -s 192.168.1.0/24 -j ACCEPT -m comment --comment "cluster subnet for inter-node communication"  
/etc/init.d/iptables save
```
Apache HTTP Server worker nodes are Servlet containers that are mapped to the mod_jk load balancer. The worker nodes are defined in `HTTPD_DIST/conf/workers.properties`. This file specifies where the different Servlet containers are located, and how calls should be load-balanced across them.

The `workers.properties` file contains two sections:

**Global Properties**

This section contains directives that apply to all workers.

**Worker Properties**

This section contains directives that apply to each individual worker.

Each node is defined using the Worker Properties naming convention. The worker name can only contain alphanumeric characters, limited to `[a-z][A-Z][0-9]`\ [-] `.`

The structure of a Worker Property is `worker.worker_name.directive`.

**worker**

The constant prefix for all worker properties.

**worker_name**

The arbitrary name given to the worker. For example: node1, node_01, Node_1.

**directive**

The specific directive required.

The main directives required to configure worker nodes are described below.

**worker.properties Global Directives**

**worker.list**

Specifies the list of worker names used by mod_jk. The workers in this list are available to map requests to.

**NOTE**

A single node configuration, which is not managed by a load balancer, must be set to `worker.list=[worker name]`.

**worker.properties Mandatory Directives**
type
Specifies the type of worker, which determines the directives applicable to the worker. The default value is **ajp13**, which is the preferred worker type to select for communication between the web server and Apache HTTP Server.

Other values include **ajp14**, **lb**, **status**.

For detailed information about ajp13, refer to The Apache Tomcat Connector - AJP Protocol Reference

workers.properties Connection Directives

host
The hostname or IP address of the worker. The worker node must support the ajp13 protocol stack. The default value is **localhost**.

You can specify the *port* directive as part of the host directive by appending the port number after the hostname or IP address. For example: `worker.node1.host=192.168.2.1:8009` or `worker.node1.host=node1.example.com:8009`

port
The port number of the remote server instance listening for defined protocol requests. The default value is **8009**, which is the default listen port for AJP13 workers. If you are using AJP14 workers, this value must be set to **8011**.

ping_mode
Specifies the conditions under which connections are probed for their current network health.

The probe uses an empty AJP13 packet for the CPing, and expects a CPong in return, within a specified timeout.

You specify the conditions by using a combination of the directive flags. The flags are not comma-separated. For example, a correct directive flag set is `worker.node1.ping_mode=C1`, which specifies that the connection will be pinged on connecting to the server and at regular intervals afterward.

C (connect)
Specifies the connection is probed once after connecting to the server. You specify the timeout using the *connect_timeout* directive, otherwise the value for *ping_timeout* is used.

P (prepost)
Specifies the connection is probed before sending each request to the server. You specify the timeout using the *prepost_timeout* directive, otherwise the value for *ping_timeout* is used.

I (interval)
Specifies the connection is probed during regular internal maintenance cycles. You specify the idle time between each interval using the *connection_ping_interval* directive, otherwise the value for *ping_timeout* is used.

A (all)
The most common setting, which specifies all directive flags are applied. For information about the `_timeout` advanced directives, refer directly to Apache Tomcat Connector - Reference Guide.

**ping_timeout**

Specifies the time to wait for CPong answers to a CPing connection probe (refer to `ping_mode`). The default value is 10000 (milliseconds).

**worker.properties Load Balancing Directives**

**lbfactor**

Specifies the load-balancing factor for an individual worker, and is only specified for a member worker of a load balancer.

This directive defines the relative amount of HTTP request load distributed to the worker compared to other workers in the cluster.

A common example where this directive applies is where you want to differentiate servers with greater processing power than others in the cluster. For example, if you require a worker to take three times the load of other workers, specify `worker.worker_name.lbfactor=3`.

**balance_workers**

Specifies the worker nodes that the load balancer must manage. The directive can be used multiple times for the same load balancer, and consists of a comma-separated list of worker names as specified in the workers.properties file.

**sticky_session**

Specifies whether requests for workers with SESSION IDs are routed back to the same worker. The default is 0 (false). When set to 1 (true), load balancer persistence is enabled.

For example, if you specify `worker.loadbalancer.sticky_session=0`, each request is load balanced between each node in the cluster. In other words, different requests for the same session will go to different servers based on server load.

If `worker.loadbalancer.sticky_session=1`, each session is persisted (locked) to one server until the session is terminated, providing that server is available.
APPENDIX B. REFERENCE: JAVA PROPERTIES

Read this appendix to learn about the JBoss HTTP Connector (mod_cluster) configuration properties that apply to either a JBoss Enterprise Application Platform or JBoss Enterprise Application Platform server node.

B.1. PROXY CONFIGURATION

The configuration values are sent to proxies under the following conditions:

- During server startup;
- When a proxy is detected through the advertise mechanism;
- During error recovery, when a proxy's configuration is reset.

Proxy Configuration Attributes

**stickySession**

Specifies whether subsequent requests for a given session should be routed to the same node, if possible. Default is true.

**stickySessionRemove**

Specifies whether the httpd proxy should remove session stickiness if the balancer is unable to route a request to the node to which it is stuck. This property is ignored if stickySession is false. Default is false.

**stickySessionForce**

Specifies whether the httpd proxy should return an error if the balancer is unable to route a request to the node to which it is stuck. This property is ignored if stickySession is false. Default is true.

**workerTimeout**

Specifies the number of seconds to wait for a worker to become available to handle a request. When all the workers of a balancer are usable, mod_cluster will retry after a while (workerTimeout/100) to find a usable worker.

A value of -1 indicates that the httpd will not wait for a worker to be available and will return an error if no workers are available. Default is -1.

**maxAttempts**

Specifies the number of times the httpd proxy will attempt to send a given request to a worker before aborting. The minimum value is 1: try once before aborting. Default is 1.

**flushPackets**

Specifies whether packet flushing is enabled or disabled. Default is false.

**flushWait**

Specifies the time to wait before flushing packets. A value of -1 means wait forever. Default is -1.
ping
Time to wait (in seconds) for a pong answer to a ping. Default is 10.

smax
Specifies the soft maximum idle connection count. The maximum value is determined by the httpd thread configuration (ThreadsPerChild or 1).

ttl
Specifies the time (in seconds) idle connections persist, above the smax threshold. Default is 60.
	nodeTimeout
Specifies the time (in seconds) mod_cluster waits for the back-end server response before returning an error. The connectiontimeout value used by mod_cluster is the ping value. Default is -1.

balancer
Specifies the name of the load-balancer. Default is mycluster.

domain
Optional parameter, which specifies how load is balanced across jvmRoutes within the same domain. domain is used in conjunction with partitioned session replication (for example, buddy replication).
APPENDIX C. REVISION HISTORY

Revision 5.2.0-103.400 2013-10-30 Rüdiger Landmann
Rebuild with publican 4.0.0

Revision 5.2.0-103 Thu Jul 12 2013 Russell Dickenson
Incorporated changes for JBoss Enterprise Application Platform 5.2.0 GA. For information about documentation changes to this guide, refer to Release Notes 5.2.0.

Revision 5.1.2-100 Thu Dec 8 2011 Jared Morgan
Incorporated changes for JBoss Enterprise Application Platform 5.1.2 GA. For information about documentation changes to this guide, refer to Release Notes 5.1.2.

Revision 5.1.1-100 Mon Jul 18 2011 Jared Morgan
Incorporated changes for JBoss Enterprise Application Platform 5.1.1 GA. For information about documentation changes to this guide, refer to Release Notes 5.1.1.