JBoss Enterprise Web Server 1.0

HTTP Connectors Load Balancing Guide

HTTP load balancing for the JBoss Enterprise Application Platform and Enterprise Web Server

Joshua Wulf

Samuel Mendenhall

James Livingston

Jim Tyrell

Laura Bailey

Jared Morgan

JBoss Enterprise Web Server 1.0 HTTP Connectors Load Balancing Guide
HTTP load balancing for the JBoss Enterprise Application Platform and Enterprise Web Server
Edition 1.0.2

Author	Joshua Wulf	jwulf@redhat.com
Author	Samuel Mendenhall	
Author	James Livingston	
Author	Jim Tyrell	
Author	Laura Bailey	lbailey@redhat.com
Author	Jared Morgan	jmorgan@redhat.com

Read this guide to install and configure the JBoss Enterprise Application Platform and Enterprise Web Server HTTP connectors: mod_jk, mod_cluster, ISAPI, and NSAPI. This guide also discusses clustering and load-balancing using these connectors.

Preface

1. File Name Conventions

The following naming conventions are used in file paths for readability. Each convention is styled differently to help you identify them in context:

`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 9 x86 or SPARC 64: **`/opt/SUNWwbsrv61/`**
- for Solaris 10 x86 or SPARC 64: **`/opt/SUNWwbsrv70/`**

`HTTPD_DIST`

The installation root of the Apache httpd 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 JBoss 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_EWS_DIST`*`/jboss-as-web/server`**.

2. Document Conventions

This manual uses several conventions to highlight certain words and phrases and draw attention to specific pieces of information.

In PDF and paper editions, this manual uses typefaces drawn from the *Liberation Fonts*[1] set. The Liberation Fonts set is also used in HTML editions if the set is installed on your system. If not, alternative but equivalent typefaces are displayed. Note: Red Hat Enterprise Linux 5 and later includes the Liberation Fonts set by default.

[1] https://fedorahosted.org/liberation-fonts/

2.1. Typographic Conventions

Four typographic conventions are used to call attention to specific words and phrases. These conventions, and the circumstances they apply to, are as follows.

`Mono-spaced Bold`

Used to highlight system input, including shell commands, file names and paths. Also used to highlight keycaps and key combinations. For example:

> To see the contents of the file **`my_next_bestselling_novel`** in your current working directory, enter the **`cat my_next_bestselling_novel`** command at the shell prompt and press **`Enter`** to execute the command.

The above includes a file name, a shell command and a keycap, all presented in mono-spaced bold and all distinguishable thanks to context.

Key combinations can be distinguished from keycaps by the hyphen connecting each part of a key combination. For example:

> Press **`Enter`** to execute the command.
>
> Press **`Ctrl`**+**`Alt`**+**`F2`** to switch to the first virtual terminal. Press **`Ctrl`**+**`Alt`**+**`F1`** to return to your X-Windows session.

The first paragraph highlights the particular keycap to press. The second highlights two key combinations (each a set of three keycaps with each set pressed simultaneously).

If source code is discussed, class names, methods, functions, variable names and returned values mentioned within a paragraph will be presented as above, in **`mono-spaced bold`**. For example:

> File-related classes include **`filesystem`** for file systems, **`file`** for files, and **`dir`** for directories. Each class has its own associated set of permissions.

Proportional Bold

This denotes words or phrases encountered on a system, including application names; dialog box text; labeled buttons; check-box and radio button labels; menu titles and sub-menu titles. For example:

> Choose **System** → **Preferences** → **Mouse** from the main menu bar to launch **Mouse Preferences**. In the **Buttons** tab, click the **Left-handed mouse** check box and click **Close** to switch the primary mouse button from the left to the right (making the mouse suitable for use in the left hand).
>
> To insert a special character into a **gedit** file, choose **Applications** → **Accessories** → **Character Map** from the main menu bar. Next, choose **Search** → **Find…** from the **Character Map** menu bar, type the name of the character in the **Search** field and click **Next**. The character you sought will be highlighted in the **Character Table**. Double-click this highlighted character to place it in the **Text to copy** field and then click the **Copy** button. Now switch back to your document and choose **Edit** → **Paste** from the **gedit** menu bar.

The above text includes application names; system-wide menu names and items; application-specific menu names; and buttons and text found within a GUI interface, all presented in proportional bold and all distinguishable by context.

`Mono-spaced Bold Italic` or ***`Proportional Bold Italic`***

Whether mono-spaced bold or proportional bold, the addition of italics indicates replaceable or variable text. Italics denotes text you do not input literally or displayed text that changes depending on circumstance. For example:

> To connect to a remote machine using ssh, type **ssh *username@domain.name*** at a shell prompt. If the remote machine is **example.com** and your username on that machine is john, type **ssh john@example.com**.
>
> The **mount -o remount *file-system*** command remounts the named file system. For example, to remount the **/home** file system, the command is **mount -o remount /home**.
>
> To see the version of a currently installed package, use the **rpm -q *package*** command. It will return a result as follows: ***package-version-release***.

Note the words in bold italics above — username, domain.name, file-system, package, version and release. Each word is a placeholder, either for text you enter when issuing a command or for text displayed by the system.

Aside from standard usage for presenting the title of a work, italics denotes the first use of a new and important term. For example:

> Publican is a *DocBook* publishing system.

2.2. Pull-quote Conventions

Terminal output and source code listings are set off visually from the surrounding text.

Output sent to a terminal is set in **mono-spaced roman** and presented thus:

```
books        Desktop   documentation  drafts  mss    photos   stuff  svn
books_tests  Desktop1  downloads      images  notes  scripts  svgs
```

Source-code listings are also set in **mono-spaced roman** but add syntax highlighting as follows:

```
package org.jboss.book.jca.ex1;

import javax.naming.InitialContext;

public class ExClient
{
   public static void main(String args[])
       throws Exception
   {
      InitialContext iniCtx = new InitialContext();
      Object         ref    = iniCtx.lookup("EchoBean");
      EchoHome       home   = (EchoHome) ref;
      Echo           echo   = home.create();

      System.out.println("Created Echo");

      System.out.println("Echo.echo('Hello') = " + echo.echo("Hello"));
   }
}
```

2.3. Notes and Warnings

Finally, we use three visual styles to draw attention to information that might otherwise be overlooked.

Note

Notes are tips, shortcuts or alternative approaches to the task at hand. Ignoring a note should have no negative consequences, but you might miss out on a trick that makes your life easier.

Important

Important boxes detail things that are easily missed: configuration changes that only apply to the current session, or services that need restarting before an update will apply. Ignoring a box labeled 'Important' will not cause data loss but may cause irritation and frustration.

Warning

Warnings should not be ignored. Ignoring warnings will most likely cause data loss.

3. We Need Feedback!

If you find a typographical error in this manual, or if you have thought of a way to make this manual better, we would love to hear from you! Please submit a report in Bugzilla: *http://bugzilla.redhat.com/bugzilla/* against the product **JBoss Enterprise Web Server.**

When submitting a bug report, be sure to mention the manual's identifier:
doc_HTTP_Connectors_Guide

If you have a suggestion for improving the documentation, try to be as specific as possible when describing it. If you have found an error, please include the section number and some of the surrounding text so we can find it easily.

Part I. Apache Tomcat Connector

Overview

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 httpd 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 takes the load of the disabled node. Two distinct functions must be performed:

- Session state replication
- Load-balancing HTTP Requests

Sessions state replication is handled by JBoss at the application level (refer to *Section 15.1, “Enabling session replication in your application”*.

Load balancing, however, requires an external load balancer. A cost effective way of managing load-balancing is to set up a software load balancer using Apache httpd and mod_jk.

Download and Install

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

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.

Configure load balancing using Apache and mod_jk

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

Task: Configure Apache to Load mod_jk

Task: Configure Apache to Load mod_jk

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, workers.properties Reference*.
- *Task: Configure Apache to Load mod_jk*

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

```
# 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**.

```
<!--Preceeding syntax removed for readability -->
<Engine name="jboss.web" defaultHost="localhost" jvmRoute="node1">
<!--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.

```
<Connector protocol="AJP/1.3" port="8009" address="${jboss.bind.address}"
   redirectPort="8443" />
```

5. You now have a correctly configured Apache httpd 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

For supplementary information about using mod_jk with JBoss, refer to the JBoss wiki page at *http://www.jboss.org/community/wiki/UsingModjk12WithJBoss*.

Part II. JBoss HTTP Connector

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, utilizing Apache as the proxy server.

4.1. Key Features

Apache HTTP Server-based

> The JBoss HTTP Connector **mod-cluster** uses Apache 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 on 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 lifecycle 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.

4.2. Components

Proxy Server

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

Shared Memory Manager: **`mod_slotmem.so`**

> The Shared Memory Manager module, **mod_slotmem**, makes the real-time worker node information available to multiple Apache 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 where the proxy is listening for responses from nodes that wish to join the load-balancing cluster.

Note

Refer to *Section 5.1, “Apache Modules”* for detailed information about the available modules including user-configurable parameters.

Worker Node Components

Worker node service: **`mod-cluster.sar`**

The JBoss HTTP Connector client service **`mod-cluster.sar`** is deployed on each worker node. 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.

Install Proxy Server Components

Read this chapter to install the JBoss HTTP Connector **mod-cluster** on a JBoss Enterprise Web Server proxy server.

5.1. Apache Modules

Read this section for expanded definitions of the Apache proxy server modules discussed in *Section 4.2, “Components”*. You specify these modules as part of *Task: Install Proxy Server Components*.

5.1.1. 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 for generated keys for shared memory and lock files. *This must be an absolute path name.* It is recommended that this path be on a local drive, and not a NFS share. The default value is **`/logs/`**.

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

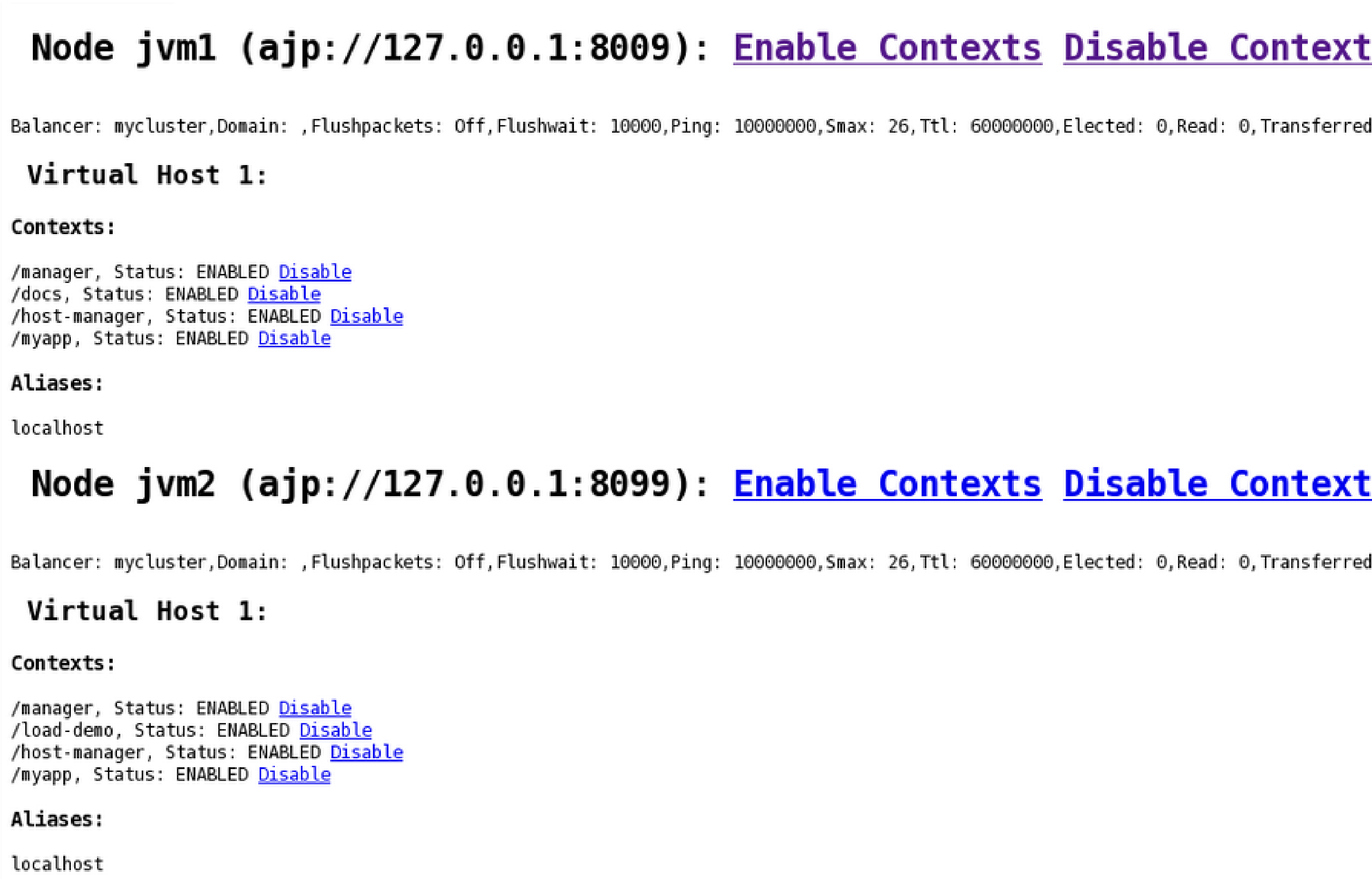
Setting this directive to **`off`** can leave your server vulnerable to replay attacks.

SetHandler

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

```
<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`**.)



Node jvm1 (ajp://127.0.0.1:8009): Enable Contexts Disable Context

Balancer: mycluster,Domain: ,Flushpackets: Off,Flushwait: 10000,Ping: 10000000,Smax: 26,Ttl: 60000000,Elected: 0,Read: 0,Transferred

Virtual Host 1:

Contexts:

```
/manager, Status: ENABLED Disable
/docs, Status: ENABLED Disable
/host-manager, Status: ENABLED Disable
/myapp, Status: ENABLED Disable
```

Aliases:

localhost

Node jvm2 (ajp://127.0.0.1:8099): Enable Contexts Disable Context

Balancer: mycluster,Domain: ,Flushpackets: Off,Flushwait: 10000,Ping: 10000000,Smax: 26,Ttl: 60000000,Elected: 0,Read: 0,Transferred

Virtual Host 1:

Contexts:

```
/manager, Status: ENABLED Disable
/load-demo, Status: ENABLED Disable
/host-manager, Status: ENABLED Disable
/myapp, Status: ENABLED Disable
```

Aliases:

localhost

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`**.

5.1.2. 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.

ProxyPassMatch; 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`**.

```
ProxyPass /requested/ http://worker.local.com/
```

`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)$ !
```

5.1.3. 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

> Defines how the advertising mechanism is used.
>
> When set to **`On`**, the advertising mechanism is used to tell worker nodes to send status information to this proxy. 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`**. When set to **`off`**, the proxy does not advertise its location.

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 specified 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`**.

5.2. Install Proxy Server Components

Task: Install Proxy Server Components

Follow this task to install the JBoss HTTP Connector on a JBoss Enterprise Web Server.

The JBoss HTTP Connector is supported in production only with JBoss Enterprise Web Server as the proxy server. Refer to the JBoss Enterprise Web Server *Installation Guide* to download and install the JBoss Enterprise Web Server.

The Native components are Operating System and processor architecture specific. 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

- JBoss Enterprise Web Server v1.0.1 or later installed.
- JBoss Enterprise Application Platform 5 Native components downloaded.

1. **Extract Apache 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 modules to JBoss Enterprise Web Server**
   Copy the JBoss HTTP Connector modules to the ***JBOSS_EWS_DIST*/httpd/modules** directory of the JBoss Enterprise Web Server.

3. **Disable the mod_proxy_balancer module**
   Edit the JBoss Enterprise Web Server Apache configuration file ***JBOSS_EWS_DIST*/httpd/conf/httpd.conf** and comment out the following line by adding an initial **#**:

   ```
   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 the file ***JBOSS_EWS_DIST*/httpd/conf.d/JBoss_HTTP.conf**.

   b. Add the following lines to the file ***JBOSS_EWS_DIST*/httpd/conf.d/JBoss_HTTP.conf**:

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

5. **Restart the JBoss Enterprise Web Server Apache service**
   Refer to the JBoss Enterprise Web Server documentation for detailed instructions.

Configure Basic Proxy Server

Follow the instructions in this chapter to configure a JBoss Enterprise Web Server to use the JBoss HTTP connector mod_cluster.

6.1. 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.

6.2. Configure a Load-balancing Proxy Using the HTTP Connector

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

proxy using the JBoss HTTP Connector.

details.

- Install JBoss HTTP Connector modules. Refer to *Chapter 5, Install Proxy Server Components* for details.

Chap... Configure Basic Proxy Server

Note

Task

1. Listen *IP_ADDRESS*:*PORT_NUMBER*

   wh... *PORT_NUMBER* is the port on that interface to listen on

   fol...

   Example 6.1. Example Listen Directive

```
Listen 10.33.144.3:6666
```

2.

```
<VirtualHost IP_ADDRESS:PORT_NUMBER>

   <Directory>
      Order deny,allow
      Deny from all
      Allow from 10.33.144.
   </Directory>

   KeepAliveTimeout 60
   MaxKeepAliveRequests 0

   ManagerBalancerName mycluster
   AdvertiseFrequency 5
```

Where *IP_ADDRESS* and *PORT_NUMBER* are the values from the Listen directive.

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.

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.

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

```
ServerAdvertise Off
```

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.

4. **Restart the JBoss Enterprise Web Server Apache service.**

   Refer to the JBoss Enterprise Web Server documentation for detailed directions.

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.

7.1. 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.

7.2. 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.

2.

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

Important

Chap

3.

4.

Task

5.

se

A p le

```
/sbin/iptables -I RH-Firewall-1-INPUT 5 -p udp -D 224.0.1.0/24 -j ACCEPT
/etc/init.d/iptables save
/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
```

the worker node to dynamically discover proxy servers, open port 23364 for UDP connections on the worker node's firewall.

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, Java Properties Reference*

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**.

   ```
   <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:

   ```
   <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 -m state --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!!!
   ```

Further Server Configuration

Read this chapter to implement further server configuration.

8.1. Apache Server Directives

Add the following Apache server directives to the Apache server configuration for server-wide effect.

8.1.1. CreateBalancers

The CreateBalancers directive determines how HTTP balancers are created in VirtualHosts.

CreateBalancers values

0

Create a balancer in all VirtualHosts.

1

Do not create balancers.

2

Create a balancer for the main server only.

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.

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 7, 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:

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

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

   Option 1: Create a static proxy server list
   Edit the file ***JBOSS_EAP_DIST*/jboss-as/server/*PROFILE*/mod-cluster.sar/META-INF/mod-cluster-jboss-beans.xml** and add a comma separated list of proxies in the form of *IP_ADDRESS:PORT* in the **proxyList** property.

   Example 9.1. Example Static Proxy List

   ```
   <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:

   ```
   <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

   ```
   -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 7, Install Node with Basic Configuration* for directions.
- Understand the Proxy Configuration parameters discussed in *Appendix B, Java Properties Reference*

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

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

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

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

   ```
   <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"/>
   ```

9.2. Clustered Node Operation

The JBoss HTTP Connector can operate in non-clustered or clustered mode.

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.

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-5.1/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, which send requests through the load balancer to **`load-demo.war`**'s primary servlet. The application displays information about which servers are handling the requests. It can also send separate requests to **`load-demo.war`**'s load generation servlets, allowing the user to see how certain load conditions affect request load balancing.

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

The demo is not a load testing tool

The demo does not show the maximum load a cluster configuration can handle.

10.1. Set up the Demonstration

The following procedure summarizes how set up and start the demonstration. These steps will then be explained in further detail. Once the demo is running, refer to *Section 10.3, “Interact with the Demonstration”*.

In a terminal, execute the following command:

- For JBoss Enterprise Application Platform:

```
[home]$ ./JBOSS_EAP_DIST/bin/run.sh
```

3. **Specify the Catalina Service Name**

Task: Start the Demo

AWAITING CONFIRMATON ON JBPAPP-6550

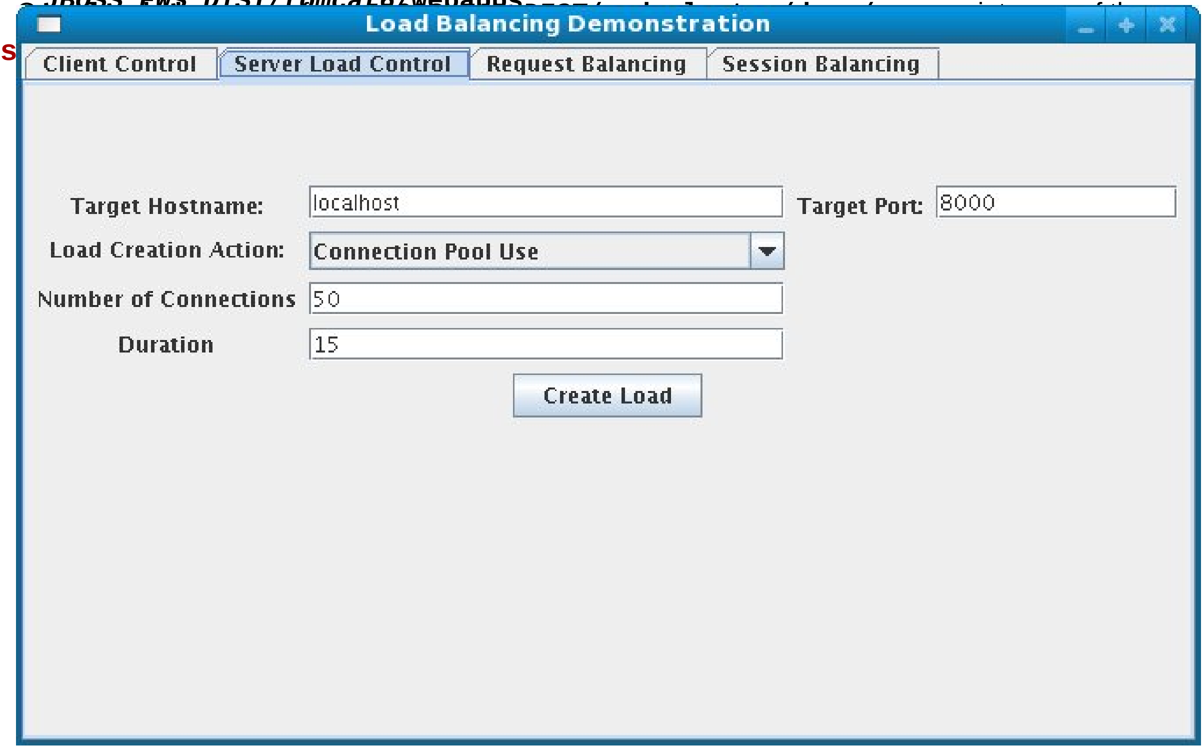
- For JBoss Enterprise Web Server:

4. **Deploy Demo Web Archive to Worker Node**

JBOSS_EWS_DIST/tomcat6/webapps

Res

5.



Load Balancing Demonstration

Target Hostname: localhost

Target Port: 8000

Load Creation Action: Connection Pool Use

Number of Connections 50

Duration 15

10.2. 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 learn the values you must supply in the **Client Control** tab of the Load Balancing Demonstration.

Prerequisites

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

1. Click the **Client Control** tab.

2. Based on the following field definitions, supply values for all fields on the **Client Control** tab.

   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

   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.

3. Once you have specified the values, proceed to *Task: Interact with the Demonstration*.

10.3. 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 experiment with load balancing in the demonstration.

Task Prerequesites

- 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 your worker nodes.
2. Click on the **Session Balancing** tab to see how many active sessions are being hosted by each of your 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 your worker nodes, or re-deploy the **`load-demo.war`** to some of your 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 10.3.1, “Generating Artificial Load”* for details.)

10.3.1. Generating 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 your 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.

Part III. Internet Server API

Overview

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

11.1. 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)
- Filters (applications that modify or enhance IIS functionality by constantly filtering for requests relevant to their functionality).

You implement the ISAPI applications by compiling Extensions or Filters into Dynamic Link Library (DLL) files. The DLLs are registered with the web server which makes them available to use.

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.

12.1. Prerequisites and Configuration Assumptions

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

> **Important**
>
> The tasks in this chapter assume all 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 Platform to a public IP address. Ensure you edit the **`workers.properties`** file on the IIS machine to reflect the IP address changes.

- Configure the master node by installing any of the 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
- Configure worker nodes by installing JBoss Enterprise Application Platform 5.1.0 or later. The Native components are optional for worker nodes.

> **Note**
>
> Refer to the *Installation Guide* for installation instructions, and to learn more about native components.

12.2. 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 12.1, “Prerequisites and Configuration Assumptions”*

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

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

   ```
   <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:

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

   For the **`default-02`** server profile:

   ```
   <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:

   ```
   <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>
   ```

4. **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.

   ```
   JBOSS_EAP_DIST\bin\run.bat --host=127.0.0.1 -c default-01
   ```

   ```
   JBOSS_EAP_DIST\bin\run.bat --host=127.0.0.100 -c default
   ```

12.3. Microsoft IIS 6 Initial Clustering Configuration

Microsoft IIS 6 contains basic ISAPI filters and ISAPI mapping as part of the default installation.

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`**.
   - **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 **`C:\connectors\jboss-ep-5.1\native\sbin\isapi_redirect.dll`**

6. 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 **`C:\connectors\jboss-ep-5.1\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 **`Scrips 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`**

   Required files:
   Specify the path **`C:\connectors\jboss-ep-5.1\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.

12.4. 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. [http://technet.microsoft.com/en-us/library/cc730912(WS.10).aspx].

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`**.

   - **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 in the

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 **`c:\connectors\jboss-ep-5.1\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 an 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 `C:\connectors\jboss-ep-5.1\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 `C:\connectors\jboss-ep-5.1\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 in the

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.

12.5. Configure a Basic Cluster with ISAPI

12.6. Configure a Load-balancing Cluster with ISAPI

Part IV. Netscape Server API

Chapter 13.

What Is Netscape Server API

Read this chapter to gain a basic understand about 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 HTTP 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

6. Log transaction.

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.

Configuring the NSAPI Connector on Solaris

The following tasks describe how to configure the NSAPI connector to use a JBoss Enterprise Platform as a worker node for a Sun Java System Web Server (SJWS) master node.

Note

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 Platform to a public IP address. Remember to edit the **`workers.properties`** file on the SJWS machine to reflect these changes in IP address.

14.1. Prerequisites and Configuration Assumptions

- Worker nodes are already installed with a JBoss Enterprise Platform 5.1 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 9 x86 with Sun Java System Web Server 6.1 SP12

  - Solaris 9 SPARC 64 with Sun Java System Web Server 6.1 SP12

  - 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

14.2. 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 14.1, “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.)

   ```
   [user@workstation jboss-ep-5.1]$ 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:

   ```
   <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:

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

   For the **`default-02`** server profile:

   ```
   <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:

   ```
   <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>
   ```

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-5.1]$ ./jboss-as/bin/run.sh -b 127.0.0.1 -c default-01
   ```

   ```
   [user@workstation jboss-eap-5.1]$ ./jboss-as/bin/run.sh -b 127.0.0.100 -c default-02
   ```

14.3. Initial Clustering Configuration

Task: Configure Initial Clustering Behavior

Complete this task to configure the basic elements required for clustering using Sun Java Web Server

Prerequisites

- *Task: Configure a JBoss Enterprise Application Platform Worker Node*
- Native Zip extracted to **/tmp/connectors/jboss-ep-native-5.1/**. This path 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, as shown in the code sample:

   - default
   - invoker
   - jsp

```
<!-- ==================== Built In Servlet Mappings ===================== -->

<!-- The servlet mappings for the built in servlets defined above. -->

<!-- The mapping for the default servlet -->
<!--servlet-mapping>
 <servlet-name>default</servlet-name>
 <url-pattern>/</url-pattern>
</servlet-mapping-->

<!-- The mapping for the invoker servlet -->
<!--servlet-mapping>
 <servlet-name>invoker</servlet-name>
 <url-pattern>/servlet/*</url-pattern>
</servlet-mapping-->

<!-- The mapping for the JSP servlet -->
<!--servlet-mapping>
 <servlet-name>jsp</servlet-name>
 <url-pattern>*.jsp</url-pattern>
</servlet-mapping-->
```

2. **Load the required modules and properties**

```
Init fn="load-modules" funcs="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

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**.

14.4. Configure a Basic Cluster with NSAPI

Use the following procedure

```
SJWS/PROFILE/bin/stopserv
SJWS/PROFILE/bin/startserv
```

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

2. For Sun Java System Web Server 7.0:

Task

3.

```
SJWS/PROFILE/stop
SJWS/PROFILE/start
```

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

For Sun Java System Web Server 6.1:

4. **Restart the server**

   Once your Sun Java System Web Server instance is configured, restart it so that your changes take effect.

```
SJWS/PROFILE/stop
SJWS/PROFILE/start
```

```
worker.router.balance_workers=worker01,worker02

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

For Sun Java System Web Server 6.1:

14.5. Configure a Load-balanced Cluster with NSAPI

4. **Restart the server**

   Once your Sun Java System Web Server instance is configured, restart it so that your changes take effect.

Part V. Common Load Balancing Tasks

Chapter 15.

HTTP Session State Replication

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

A client session that is exclusively allocated to a single server instance. The load balancer routes all HTTP requests associated with the client session to the allocated server instance only. Session persistence is commonly referred to 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.

15.1. 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 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://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd"
          version="2.4">
```

```
    <distributable/>

</web-app>
```

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. Here is an example:

```
<!DOCTYPE jboss-web PUBLIC
    -//JBoss//DTD Web Application 5.0//EN
    http://www.jboss.org/j2ee/dtd/jboss-web_5_0.dtd>

<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>
```

All of the configuration elements are optional, and can be omitted if the default value is acceptable. A couple are commonly used; the rest are very infrequently changed from the defaults. We'll cover the commonly used ones first.

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_AND_GET** is conservative but not optimal (performance-wise): it will always replicate session data even if its content has not been modified but simply accessed. 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_AND_GET`** makes no sense with Enterprise Application Platform 5.

- **SET_AND_NON_PRIMITIVE_GET** is conservative but will only replicate if an object of a non-primitive type has been accessed (i.e. 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.

The <replication-granularity> element determines the granularity of what gets replicated if the container determines session replication is needed. The supported values are:

SESSION

Specifies the entire session attribute map should be replicated when any attribute is considered modified. Replication occurs at request end. This option replicates the most data and thus incurs the highest replication cost, but since all attributes values are always replicated together it ensures that any references between attribute values will not be broken when the session is deserialized. For this reason it is the default setting.

ATTRIBUTE

Specifies only attributes that the session considers to be potentially modified are replicated. Replication occurs at request end. For sessions carrying large amounts of data, parts of which are infrequently updated, this option can significantly increase replication performance. However, it is not suitable for applications that store objects in different attributes that share references with each other (e.g. a **`Person`** object in the "husband" attribute sharing with another **`Person`** in the "wife" attribute a reference to an **`Address`** object). This is because if the attributes are separately replicated, when the session is deserialized on remote nodes the shared references will be broken.

The other elements under the **`replication-config`** element are much less frequently used.

<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 15.3, “Configuring 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. Default is **`true`**.

<useJK>

Specifies whether the container should assume that a JK-based software load balancer (e.g. 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 won't incorrectly expire an unreplicated session upon failover. It also results in correct values for **`HttpSession.getLastAccessedTime()`** calls following failover.

The default value is **`null`** (i.e. 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 (e.g. 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**; i.e. replicate the metadata on every request. Default value is **60**.

<snapshot-mode>

Specifies 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.

15.2. 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 HttpSessions passivation from clustered web applications where the **web.xml** file includes the **distributable** directive.

Passivation occurs at three points during the lifecycle 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.

15.2.1. Configuring HttpSession Passivation

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

```
<!DOCTYPE jboss-web PUBLIC
    -//JBoss//DTD Web Application 5.0//EN
    http://www.jboss.org/j2ee/dtd/jboss-web_5_0.dtd>

<jboss-web>

   <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>


</jboss-web>
```

- **max-active-session**

  Determines the maximum number of active sessions allowed. If the number of sessions managed by the 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.

- **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 **`web.xml`** **`session-timeout`** setting. A value of **`-1`** (the default) disables passivation based on maximum inactivity.

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`**. The number of sessions replicated from other nodes will also depend on whether *buddy replication* is enabled.

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.

15.3. Configuring 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. The container 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* Chapter 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 15.1, "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.

> **Note**
>
> For more information, refer to the *JBoss Cache* chapter in the *Administration and Configuration Guide*

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.

> **Note**
>
> For more information about the other available parameters for cacheMode, refer to the *Cache Mode* section in the *Administration and Configuration Guide*.

- **enabled** property in the **buddyReplicationConfig** section

  Set to **`true`** to enable buddy replication. Default is **`false`**.

  > **Note**
  >
  > For more information about the other available parameters for cacheMode, refer to the *Buddy Replication* section in the *Administration and Configuration Guide*.

- **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.

  > **Note**
  >
  > For more information about the other available parameters for cacheMode, refer to the *Buddy Replication* section in the *Administration and Configuration Guide*.

- **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.

  > **Note**
  >
  > For more information about the other available parameters for cacheMode, refer to the *Buddy Replication* section in the *Administration and Configuration Guide*.

- **multiplexerStack**

  Name of the JGroups protocol stack the cache should use.

Note

For more information about the other available parameters for cacheMode, refer to *The Channel Factory Service* section in the *Administration and Configuration Guide*.

- **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*.

Using Clustered Single Sign-on

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, whether or not 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.

16.1. 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:

- **className** 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 16.4, “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 what cluster nodes are managing sessions related to an SSO. A positive value for this attribute allows proper handling of shutdown of a node that is the only one that had handled any of the sessions associated with an SSO. The shutdown invalidates the local copy of the sessions, eliminating all sessions from the SSO. If maxEmptyLife were zero, the SSO would terminate along with the local session copies. But, backup copies of the sessions (if they are from clustered webapps) are available on other cluster nodes. Allowing the SSO to live beyond the life of its managed sessions gives the user time to make another request which can fail over to a different cluster node, where it activates the the backup copy of the session. Default is **`1800`**, i.e. 30 minutes.

- **processExpiresInterval** is the minimum number of seconds between efforts by the valve to find and invalidate SSO's that have exceeded their 'maxEmptyLife'. Does not imply effort will be spent on such cleanup every 'processExpiresInterval', just that it won't 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* 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`**.

16.2. 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 login 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.HttpSession.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.

16.3. Limitations

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

- Only useful within a cluster of JBoss servers; 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 (e.g. username and password).

16.4. 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:

```
<Valve className="org.jboss.web.tomcat.service.sso.ClusteredSingleSignOn"
                 cookieDomain="xyz.com" />
```

Chapter 17.

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"/-->
```

JBoss Web Client Service Dependencies

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.

```
<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
```

Appendix A. workers.properties Reference

Apache httpd 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.

> **Note**
>
> For the full list of **`worker.properties`** configuration directives, refer directly to the *Apache Tomcat Connector - Reference Guide*[1]

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]`***.

[1] http://tomcat.apache.org/connectors-doc/reference/workers.html

workers.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 httpd Server.

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

For detailed information about ajp13, refer to *The Apache Tomcat Connector - AJP Protocol Reference*[2]

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=CI`**

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*[3].

[2] http://tomcat.apache.org/connectors-doc/ajp/ajpv13a.html

[3] http://tomcat.apache.org/connectors-doc/reference/workers.html

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 than 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. Java Properties Reference

Read this appendix to learn about the JBoss HTTP Connector mod_cluster configuration properties that apply to a 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 Values

stickySession (defaults to **`true`**)

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

stickySessionRemove (defaults to **`false`**)

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`**.

stickySessionForce (defaults to **`true`**)

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`**.

workerTimeout (defaults to **`-1`**)

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 an 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.

maxAttempts (defaults to **`1`**)

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.

flushPackets (defaults to **`false`**)

Specifies whether packet flushing is enabled or disabled.

flushWait (defaults to **`-1`**)

Specifies the time to wait before flushing packets. A value of **`-1`** means wait forever.

ping (defaults to **`10`**)

Time to wait (in seconds) for a pong answer to a ping.

smax

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

ttl (defaults to **`60`**)

Specifies the time (in seconds) idle connections persist, above the `smax` threshold.

nodeTimeout (defaults to **-1**)

Specifies the time (in seconds) mod_cluster waits for the back-end server response before returning an error.

mod_cluster always uses a cping/cpong before forwarding a request. The connectiontimeout value used by mod_cluster is the ping value.

balancer (defaults to **mycluster**)

Specifies the name of the load-balancer.

domain (no default)

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).

B.2. Load Configuration

Appendix C. Revision History

Revision 1.0.2-9	**Tue Jun 21 2011**	**Rebecca Newton** rnewton@redhat.com

Final build for EWS 1.0.2.GA.