Red Hat JBoss Web Server 3

HTTP Connectors and Load Balancing Guide

HTTP load balancing for Red Hat JBoss Web Server

Last Updated: 2017-10-18

Red Hat JBoss Web Server 3 HTTP Connectors and Load Balancing Guide

HTTP load balancing for Red Hat JBoss Web Server

Abstract

Read this guide to install and configure Red Hat JBoss Web Server HTTP connectors: mod_jk and mod_cluster. This guide also discusses clustering and load-balancing using these connectors.

Table of Contents

CHAPTER 1. INTRODUCTION

Two different load balancing HTTP connectors are included with Red Hat JBoss Web Server.

- The Apache Tomcat Connector (mod_jk) supports the load balancing of HTTP calls to a set of Servlet containers while maintaining sticky sessions, and communicates over AJP.

  See Chapter 2, *Apache Tomcat Connector (mod_jk)*.

- mod_cluster is a more advanced load balancer than mod_jk, and provides all of the functionality of mod_jk, plus other additional features. These include real-time load balancing calculations, application life-cycle control, automatic proxy discovery, and multiple protocol support.

  See Chapter 3, *mod_cluster Connector*.

This guide also contains information on Online Certificate Status Protocol (OSCP), and a set of working examples for basic load balancing, and Kerberos authentication using mod_auth_kerb.

IMPORTANT

Most file and directory paths shown in this guide are for a Zip installation of JBoss Web Server on Red Hat Enterprise Linux. For other platforms, use the correct paths for your respective installation as specified in the JBoss Web Server *Installation Guide*.

Report a bug

CHAPTER 2. APACHE TOMCAT CONNECTOR (MOD_JK)

2.1. OVERVIEW

The Apache Tomcat Connector, **mod_jk**, is a plug-in designed to allow request forwarding from Apache HTTP Server to a Servlet container. The module also supports load-balancing HTTP calls to a set of Servlet containers while maintaining sticky sessions.

Report a bug

2.1.1. Download and Install

Apache HTTP Server is included in the Red Hat JBoss Web Server installation.

`mod_jk` is included in the native installation binaries for JBoss Enterprise Application Platform (JBoss EAP) and JBoss Web Server.

Follow the procedures in the JBoss EAP or JBoss Web Server *Installation Guide* to download and install the correct platform and native binaries.

Report a bug

2.2. CONFIGURE LOAD BALANCING USING APACHE HTTP SERVER AND MOD_JK

You can use the mod_jk connector to configure Apache HTTP Server load balancing. Follow the tasks in this section to configure load balancing, including configuring worker nodes.

Sample configuration files are provided for mod_jk, and are located in ***`JWS_HOME`*****`/httpd/conf.d/`**. The sample configuration files are: **`mod_jk.conf.sample`**, **`workers.properties.sample`**, and **`uriworkermap.properties.sample`**. To use these samples instead of creating your own configuration files, remove the **`.sample`** extension, and modify their content as needed.

NOTE

Red Hat customers can also use the *Load Balancer Configuration Tool* on the Red Hat Customer Portal to quickly generate optimal configuration templates for mod_jk and Tomcat worker nodes.

When using this tool for JBoss Web Server 3, ensure you select **`2.4.x`** as the Apache version, and select **`Tomcat`** as the back end configuration.

Report a bug

2.2.1. Configuring Apache HTTP Server to Load mod_jk

Procedure 2.1. Configure Apache HTTP Server to Load mod_jk

1. Open ***`JWS_HOME`*****`/httpd/conf/httpd.conf`** and add the following lines at the end of the file:

```
# Include mod_jk's specific configuration file
Include conf.d/mod_jk.conf
```

2. Create a new file: ***JWS_HOME*/httpd/conf.d/mod_jk.conf**

3. Add the following configuration to the **mod_jk.conf** file.

```
# Load mod_jk module
# Specify the filename of the mod_jk lib
LoadModule jk_module modules/mod_jk.so

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

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

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

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

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

# JkRequestLogFormat
JkRequestLogFormat "%w %V %T"

# Mount your applications
JkMount /application/* loadbalancer

# Add shared memory.
# This directive is present with 1.2.10 and
# later versions of mod_jk, and is needed for
# for load balancing to work properly
JkShmFile logs/jk.shm

# Add jkstatus for managing runtime data
<Location /jkstatus/>
    JkMount status
    Require ip 127.0.0.1
</Location>
```

IMPORTANT

The **LoadModule** directive must reference the **mod_jk** library directory location for the native binary you installed.

NOTE

The **JkMount** directive specifies which URLs Apache HTTP Server will forward to the **mod_jk** module. Based on the directive's configuration, **mod_jk** forwards the received URL onto the correct Servlet containers.

To enable Apache HTTP Server to serve static content (or PHP content) directly, and only use the load balancer for Java applications, the suggested configuration above specifies that all requests with the URL **/application/*** are sent to the **mod_jk** load-balancer.

Alternatively, you can forward all URLs to **mod_jk** by specifying **/*** in the directive.

4. **Optional: JKMountFile Directive**
   In addition to the **JkMount** directive, you can use the **JkMountFile** directive to specify a mount point's configuration file. The configuration file contains multiple Tomcat forwarding URL mappings.

   a. Navigate to ***JWS_HOME*/httpd/conf.d/**.

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

   c. Using the following syntax example as a guide, specify the URL to forward and the worker name.

      The syntax required takes the form: ***/URL=WORKER_NAME***

      The example block below configures **mod_jk** to forward requests to **/jmx-console** and **/web-console** to Apache HTTP Server.

```
# Simple worker configuration file

# Mount the Servlet context to the ajp13 worker
/jmx-console=loadbalancer
/jmx-console/*=loadbalancer
/web-console=loadbalancer
/web-console/*=loadbalancer
```

   d. In ***JWS_HOME*/httpd/conf.d/mod_jk.conf**, append the following directive:

```
# Use external file for mount points.
# It will be checked for updates each 60 seconds.
# The format of the file is: /url=worker
# /examples/*=loadbalancer
JkMountFile conf.d/uriworkermap.properties
```

5. **Optional: Configure Apache HTTP Server Logging**
   You can configure the Apache HTTP Server that is doing the load balancing to log which worker node handled a request. This may be useful when troubleshooting your load balancer.

   To enable this for mod_jk, you can either:

   - include **%w** in your **JkRequestLogFormat**; or

   - log the name of the mod_jk worker used by including **%{JK_WORKER_NAME}n** in your Apache HTTP Server **LogFormat**(s).

  For more information on **JkRequestLogFormat**, see http://tomcat.apache.org/connectors-doc/webserver_howto/apache.html. For more information on Apache HTTP Server logging (including log rotation), see http://httpd.apache.org/docs/2.4/logs.html.

Report a bug

2.2.2. Configuring Worker Nodes in mod_jk

2.2.2.1. Configuring Worker Nodes in mod_jk

This procedure demonstrates two mod_jk Worker node definitions in a weighted round robin configuration with sticky sessions active between two servlet containers.

Procedure 2.2. Configure mod_jk Worker Nodes

Prerequisites

- Understand the format of the **workers.properties** directives, as specified in Section A.2, "workers.properties".
- Configure mod_jk. See Section 2.2.1, "Configuring Apache HTTP Server to Load mod_jk" .

To configure mod_jk worker nodes:

1. Navigate to ***JWS_HOME*/httpd/conf.d/**, and create a file named **workers.properties**.
2. Add the following configuration into **workers.properties**.

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

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

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

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

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

Report a bug

2.2.3. Configuring Apache Tomcat to Work with mod_jk

Tomcat is configured to use mod_jk by default. Specifically, see the ***`JWS_HOME`***`/tomcat<`***`VERSION`***`>/conf/server.xml` file, which contains the following configuration for this purpose:

```
<connector port="8009" protocol="AJP/1.3" redirectPort="8443" />
```

Additionally, configure the ***`jvmRoute`*** attribute for your engine:

```
<Engine name="Catalina" jvmRoute="node1" >
```

The ***`jvmRoute`*** attribute value must match the worker name set in `workers.properties`.

This default Tomcat configuration is ready for immediate use with mod_jk.

Report a bug

CHAPTER 3. MOD_CLUSTER CONNECTOR

3.1. OVERVIEW

The mod_cluster connector is a reduced configuration, intelligent load-balancing solution for JBoss Enterprise Application Platform (JBoss EAP) and Red Hat JBoss Web Server, based on technology originally developed by the JBoss mod_cluster community project.

mod_cluster load-balances HTTP requests to JBoss EAP and JBoss Web Server worker nodes, utilizing Apache HTTP Server as the proxy server. It serves as a load balancing solution for Tomcat in JBoss Web Server as well as for JBoss EAP.

Report a bug

3.1.1. Key Features

The mod_cluster connector has several advantages over the mod_jk connector:

- The mod_cluster Management Protocol (MCMP) is an additional connection between the Tomcat servers and the Apache HTTP Server with the mod_cluster module enabled. It is used by the Tomcat servers to transmit server-side load figures and lifecycle events back to the Apache HTTP Server via a custom set of HTTP methods.
- Dynamic configuration of Apache HTTP Server with mod_cluster allows Tomcat servers that have mod_cluster listeners to join the load balancing arrangement without manual configuration.
- Tomcat servers performs the load calculations, rather than relying on the Apache HTTP Server with mod_cluster. This makes load balancing metrics more accurate than other connectors.
- The mod_cluster connector gives fine-grained application lifecycle control. Each Tomcat server forwards web application context lifecycle events to the Apache HTTP Server, informing it to start or stop routing requests for a given context. This prevents end users from seeing HTTP errors due to unavailable resources.
- AJP, HTTP or HTTPS transports can be used.

Report a bug

3.1.2. Components

On the proxy server, mod_cluster consists of four Apache modules.

Table 3.1. Components

Component	Description
`mod_cluster_slotmem.so`	The Shared Memory Manager module shares real time worker node information with multiple Apache HTTP Server processes.

Component	Description
`mod_manager.so`	The Cluster Manager module receives and acknowledges messages from nodes, including worker node registrations, worker node load data, and worker node application life cycle events.
`mod_proxy_cluster.so`	The Proxy Balancer Module handles request routing to cluster nodes. The Proxy Balancer selects the appropriate destination node 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).
`mod_advertise.so`	The Proxy Advertisement Module 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 want to join the load-balancing cluster.

See Section A.1, “Apache Modules” for detailed information about the available modules, including user-configurable parameters.

Report a bug

3.1.3. Limitations

mod_cluster uses shared memory to keep the nodes description. The shared memory is created at the startup of Apache HTTP Server, and the structure of each item is fixed. When defining proxy server and worker node properties, ensure that you follow these character limits:

- Maximum Alias length: 100 characters (Alias corresponds to the network name of the respective virtual host; the name is defined in the Host element)
- Maximum context length: 40 characters (for example, if **`myapp.war`** is deployed in **`/myapp`**, then **`/myapp`** is the context)
- Maximum balancer name length: 40 characters (the balancer property in mbean)
- Maximum JVMRoute string length: 80 characters (JVMRoute in the <Engine> element)
- Maximum domain name length: 20 characters (the domain property in mbean)
- Maximum hostname length for a node: 64 characters (hostname address in the <Connector> element)
- Maximum port length for a node: 7 characters (8009 is 4 characters, the port property in the <Connector> element)
- Maximum scheme length for a node: 6 characters (possible values are **`http`**, **`https`**, **`ajp`**, the protocol of the connector)
- Maximum cookie name length: 30 characters (the header cookie name for session ID default value: ***`JSESSIONID`*** from *`org.apache.catalina.Globals.SESSION_COOKIE_NAME`*)

- Maximum path name length: 30 characters (the parameter name for the session ID default value: ***`JSESSIONID`*** from ***`org.apache.catalina.Globals.SESSION_PARAMETER_NAME`***)
- Maximum length of a session ID: 120 characters (session ID resembles the following: **`BE81FAA969BF64C8EC2B6600457EAAAA`**`.node01`)

Report a bug

3.2. CONFIGURE LOAD BALANCING USING APACHE HTTP SERVER AND MOD_CLUSTER

In Red Hat JBoss Web Server 2.1 and higher, mod_cluster is configured correctly for Apache HTTP Server by default. To set a custom configuration, see Section 3.2.1, “Configure a Basic Proxy Server” .

For more information on configuring a Tomcat worker node with mod_cluster, see Section 3.2.2.1, “Configuring a Tomcat Worker Node”.

NOTE

Red Hat customers can also use the *Load Balancer Configuration Tool*on the Red Hat Customer Portal to quickly generate optimal configuration templates for mod_cluster, as well as Tomcat worker nodes.

When using this tool for JBoss Web Server 3, ensure you select **`2.4.x`** as the Apache version, and select **`Tomcat`** as the back end configuration.

Report a bug

3.2.1. Configure a Basic Proxy Server

3.2.1.1. Basic Proxy Configuration Overview

Proxy server configuration consists of one mandatory and one optional step:

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 advertises itself using UDP multicast. When UDP multicast is available between the proxy server and the worker nodes, Server Advertisement adds worker nodes without 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 3.2.2.2, “Configuring a Worker Node with a Static Proxy List” . In either case, the proxy server does not need to be configured with a list of worker nodes.

Report a bug

3.2.1.2. Configure a Load-balancing Proxy Using mod_cluster

Prerequisites

- Install JBoss Web Server, and configure the mod_cluster modules for your installation. See the JBoss Web Server *Installation Guide* for details.

Procedure 3.1. Configure a Load-balancing Proxy Using mod_cluster

1. **Create a Listen Directive for the Proxy Server**
   Edit your mod_cluster configuration file (usually ***JWS_HOME*/httpd/conf.d/mod_cluster.conf**), and add the following:

   ```
   Listen IP_ADDRESS:PORT_NUMBER
   ```

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

   NOTE

   The port must be open for incoming TCP connections.

   Example 3.1. Example Listen Directive

   ```
   Listen 10.33.144.3:6666
   ```

2. **Create a Virtual Host**
   Add the following to your mod_cluster configuration file:

   ```
   <VirtualHost IP_ADDRESS:PORT_NUMBER>

      <Directory />
         Require ip IP_ADDRESS
      </Directory>

      KeepAliveTimeout 60
      MaxKeepAliveRequests 0

      ManagerBalancerName mycluster
      AdvertiseFrequency 5
      EnableMCPMReceive On

   </VirtualHost>
   ```

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

3. **Optional: Disable Server Advertisement**
   The **AdvertiseFrequency** directive makes the server to periodically send server advertisement messages via UDP multicast. By default, this occurs every 5 seconds.

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

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

```
ServerAdvertise Off
```

If server advertisements are disabled, or UDP multicast is not available on the network between the proxy server and the worker nodes, configure worker nodes with a static list of proxy servers. See Section 3.2.2.2, “Configuring a Worker Node with a Static Proxy List” .

4. **Optional: Configure Apache HTTP Server Logging**
   You can configure the Apache HTTP Server that is doing the load balancing to log which worker node handled a request. This may be useful when troubleshooting your load balancer.

   To enable this for mod_cluster, you can add the following to your Apache HTTP Server **`LogFormat`** directive(s):

   %{BALANCER_NAME}e

   The name of the balancer that served the request.

   %{BALANCER_WORKER_NAME}e

   The name of the worker node that served the request.

   For more information on Apache HTTP Server logging (including log rotation), see http://httpd.apache.org/docs/2.4/logs.html.

5. **Stop and Start the JBoss Web Server Apache Service**
   See the JBoss Web Server *Installation Guide* for detailed instructions.

Report a bug

3.2.2. Configuring Worker Nodes

3.2.2.1. Configuring a Tomcat Worker Node

Follow this procedure to install mod_cluster on a Red Hat JBoss Web Server node, and configure it for non-clustered operation.

NOTE

JBoss Web Server Tomcat worker nodes only support a subset of mod_cluster functionality.

Supported Worker Node types

- Red Hat JBoss Web Server Tomcat service

mod_cluster JBoss Web Server Node Limitations

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

Prerequisites

- Install a supported JBoss Web Server.

- Understand the Proxy Configuration parameters discussed in Appendix B, *Java Properties Reference*.

Procedure 3.2. Configure a Tomcat Worker Node

1. **Add a Listener to Tomcat**
   Add the following **Listener** element beneath the other Listener elements in ***JWS_HOME*/tomcat*<VERSION>*/conf/server.xml**.

   ```
   <Listener
   className="org.jboss.modcluster.container.catalina.standalone.ModClu
   sterListener" advertise="true" stickySession="true"
   stickySessionForce="false" stickySessionRemove="true" />
   ```

2. **Give the Worker a Unique Identity**
   Edit ***JWS_HOME*/tomcat*<VERSION>*/conf/server.xml** and add the **jvmRoute** attribute and value to the **Engine** element, as shown below:

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

3. **Configure STATUS MCMP Message Frequency**
   Tomcat worker nodes periodically send status messages containing their current load status to the Apache HTTP Server balancer. The default frequency of these messages is 10 seconds. If you have hundreds of worker nodes, the STATUS MCMP Messages can increase traffic congestion on your Apache HTTP Server network.

   You can configure the MCMP message frequency by modifying the ***org.jboss.modcluster.container.catalina.status-frequency*** Java property. By default, the property accepts values in seconds*10. For example, setting the property to **1** means 10 seconds. The example below demonstrates the frequency set to 60 seconds.

   ```
   -Dorg.jboss.modcluster.container.catalina.status-frequency=6
   ```

4. **Optional Step: Configure Firewall for Proxy Server Advertisements**
   A proxy server using mod_cluster can advertise itself via UDP multicast. Most operating system firewalls block this by default. To enable server advertising and receive these multicast messages, open port 23364 for UDP connections on the worker node's firewall.

   - **For Red Hat Enterprise Linux 6**

     ```
     /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 Automatic Proxy Discovery is not used, configure worker nodes with a static list of proxies. In this case you can safely ignore the following warning message:

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

- **For Red Hat Enterprise Linux 7**

  ```
  firewall-cmd --permanent --zone=public --add-port=23364/udp
  ```

- **For Microsoft Windows, using PowerShell**

  ```
  Start-Process "$psHome\powershell.exe" -Verb Runas -ArgumentList
  '-command "NetSh Advfirewall firewall add rule name="UDP Port
  23364" dir=in  action=allow protocol=UDP localport=23364"'
  Start-Process "$psHome\powershell.exe" -Verb Runas -ArgumentList
  '-command "NetSh Advfirewall firewall add rule name="UDP Port
  23364" dir=out action=allow protocol=UDP localport=23364"'
  ```

Report a bug

3.2.2.2. Configuring a Worker Node with a Static Proxy List

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.

Use the following procedure to configure a Red Hat JBoss Web Server worker node to operate with a static list of proxy servers.

Prerequisites

- JBoss Web Server worker node configured. See Section 3.2.2.1, “Configuring a Tomcat Worker Node”.
- Understand the Proxy Configuration parameters, detailed in Appendix B, *Java Properties Reference*.

Procedure 3.3. Configure JBoss Web Server Worker Node with a Static Proxy List

1. **Define a mod_cluster listener, and Disable Dynamic Proxy Discovery**
   Edit ***JWS_HOME*/tomcat<*VERSION*>/conf/server.xml**, and add a <Listener> element to the **server.xml** file. Set the **advertise** property of the ModClusterListener to false. For example:

   ```
   <Listener
   className="org.jboss.modcluster.container.catalina.standalone.ModClu
   sterListener" advertise="false" stickySession="true"
   stickySessionForce="false" stickySessionRemove="true"/>
   ```

2. **Create a static proxy server list**
   Add a comma separated list of proxies in the form of *IP_ADDRESS:PORT* as the **proxyList** property. For example:

   Example 3.2. Example Static Proxy List

   ```
   <Listener
   className="org.jboss.modcluster.container.catalina.standalone.ModC
   lusterListener" advertise="false" stickySession="true"
   ```

```
stickySessionForce="false" stickySessionRemove="true"
proxyList="10.33.144.3:6666,10.33.144.1:6666"/>
```

Report a bug

CHAPTER 4. WEBSOCKET ON TOMCAT

4.1. ABOUT WEBSOCKET

WebSocket is a web technology that provides bi-directional, full duplex, messages to be instantly distributed between the client and server over a single TCP socket connection. A full duplex communication allows two-way communication simultaneously.

The container provides an implementations of the WebSockets 1.0 JSR 356 API. To use the API, you must run Java 7+ and configure the APR or NIO2 HTTP/1.1 connectors of the web container.

JSR 356 is a standard for WebSocket API for Java. Developers can use the JSR 356 API for creating WebSocket applications independent of the implementation. The WebSocket API is purely event driven.

Developers can use the JSR 356 Java API for WebSocket to integrate WebSockets in applications on the server side as well as on the Java client side. Tomcat 7 and 8 implement the WebSocket protocol which adheres to JSR-356 standard.

A Java client uses JSR 356-compliant client implementation to connect to a WebSocket server. For web clients, WebSocket JavaScript API can be used to communicate with WebSocket server. The only difference between a WebSocket client and a WebSocket server is the method in which they are connected. A WebSocket client is a WebSocket point from which the connection to a peer originates. A WebSocket server is WebSocket endpoint which is already published and awaits connections from peers.

Some examples where WebSocket can be used include banking, chat, multiplayer, and social networking applications.

Report a bug

4.2. IMPLEMENTING WEBSOCKET ON TOMCAT

Configuring WebSocket on Tomcat requires individual configuration of the following:

- Configuring write timeout
- Configuring incoming binary messages
- Configuring incoming text messages
- Configuring additional programmatic deployment
- Configuring callbacks for asynchronous writes
- Configuring timeout for IO operations while establishing the connections

Configuring write timeout

You can change the write timeout in blocking mode by using the `org.apache.tomcat.websocket.BLOCKING_SEND_TIMEOUT` property. The property accepts values in milliseconds. The default value is 20000 milliseconds (20 seconds).

Configuring incoming binary messages

To configure incoming binary messages, `MessageHandler.Partial` must be defined. If `MessageHandler.Partial` is not defined then incoming binary messages must be buffered so that the entire message is delivered in a single call to `MessageHandler.Whole`.

The default buffer size for binary messages is 8192 bytes. You can change the buffer size for a web application by changing the value of the servlet context initializing parameter `org.apache.tomcat.websocket.binaryBufferSize`.

Configuring incoming text messages

To configure incoming text messages, `MessageHandler.Partial` must be defined. If `MessageHandler.Partial` is not defined then incoming text messages must be buffered so that the entire message is delivered in a single call to `MessageHandler.Whole`.

The default buffer size for text messages is 8192 bytes. You can change the buffer size for a web application by changing the value of the servlet context initializing parameter `org.apache.tomcat.websocket.textBufferSize`.

Configuring additional programmatic deployment

Java WebSocket specification 1.0 does not allow programmatic deployment after the first endpoint has started a WebSocket handshake. However, Tomcat by default allows additional programmatic deployment. Additional programmatic deployment can be done by using the servlet context initialization parameter `org.apache.tomcat.websocket.noAddAfterHandshake`.

Set the system property `org.apache.tomcat.websocket.STRICT_SPEC_COMPLIANCE` to true to change the default setting.

Configuring callbacks for asynchronous writes

Callbacks for asynchronous writes need to be performed on a different thread to the thread that initiated the write. The container thread pool is not exposed via the Servlet API. Hence the WebSocket implementation has to provide its own thread pool.

The following servlet context initialization parameters control the thread pool:

- `org.apache.tomcat.websocket.executorCoreSize`

  The core size of the executor thread pool. If not set, the default of 0 (zero) is used.

- `org.apache.tomcat.websocket.executorMaxSize`

  The maximum permitted size of the executor thread pool. If not set, the default of 10 is used.

- `org.apache.tomcat.websocket.executorKeepAliveTimeSeconds`

  The maximum time an idle thread will remain in the executor thread pool until it is terminated. If not specified, the default of 60 seconds is used.

Configuring timeout for IO operations while establishing the connections

The timeout for IO operations while establishing the connections is controlled by `userProperties` of the provided `javax.websocket.ClientEndpointConfig`. You can change timeout by changing the `org.apache.tomcat.websocket.IO_TIMEOUT_MS` property. The property accepts the values in milliseconds. The default value is 5000 (5 seconds).

To connect WebSocket client to secure server endpoints, the client SSL configuration is controlled by `userProperties` of the provided `javax.websocket.ClientEndpointConfig`.

The following user properties are supported:

- `org.apache.tomcat.websocket.SSL_CONTEXT`
- `org.apache.tomcat.websocket.SSL_PROTOCOLS`
- `org.apache.tomcat.websocket.SSL_TRUSTSTORE`
- `org.apache.tomcat.websocket.SSL_TRUSTSTORE_PWD`

The default truststore password is changeit. The `org.apache.tomcat.websocket.SSL_TRUSTSTORE` and `org.apache.tomcat.websocket.SSL_TRUSTSTORE_PWD` properties are ignored if the `org.apache.tomcat.websocket.SSL_CONTEXT` property is set.

Report a bug

CHAPTER 5. ONLINE CERTIFICATE STATUS PROTOCOL

5.1. CONFIGURING APACHE HTTP SERVER FOR SSL CONNECTIONS

Procedure 5.1. Configure Apache HTTP Server for SSL Connections

1. Install mod_ssl using the following command:

   ```
   # yum install mod_ssl
   ```

2. Edit ***JWS_HOME*/httpd/conf.d/ssl.conf**, and add ***ServerName***, ***SSLCertificateFile***, and ***SSLCertificateKeyFile***:

   ```
   <VirtualHost _default_:443>
   ServerName www.example.com:443
   SSLCertificateFile /etc/pki/tls/certs/localhost.crt
   SSLCertificateKeyFile /etc/pki/tls/private/localhost.key
   ```

   a. ***ServerName*** must match the Common Name (CN) of the SSL certificate. If the ***ServerName*** does not match the CN, client browsers display domain name mismatch errors.

   b. The ***SSLCertificateFile*** is the private key associated with the certificate (the public key).

   c. Verify that the Listen directive in the **ssl.conf** file is correct as per your configuration. For example, if an IP address is specified, it must match the IP address the httpd service is bound to.

3. Restart httpd using the following command:

   ```
   # service httpd restart
   ```

Report a bug

5.2. ABOUT ONLINE CERTIFICATE STATUS PROTOCOL (OCSP)

Online Certificate Status Protocol (OCSP) is a technology which allows web browsers and web servers to communicate over a secured connection. The encrypted data is sent from one side and decrypted by the other side before processing. The web browser and the web server both encrypt and decrypt the data.

During communication with a web server, the server presents a set of credentials in the form of certificate. The browser then checks the certificate for its validity and sends a request for certificate status information. The server sends back a status as current, expired, or unknown. The certificate specifies syntax for communication and contains control information such as start time and end time, and address information to access an OCSP responder. The web server can use an OCSP responder it has been configured for, or the one listed in the certificate to check the status. OCSP allows a grace period for expired certificates, which allows access to a server for a limited time before renewing the certificate.

Online Certificate Status Protocol overcomes limitations of the older method, Certificate Revocation List (CRL). For more information on OCSP, see the *Red Hat Certificate System Admin Guide*and https://access.redhat.com/site/articles/417843.

Report a bug

5.3. USING ONLINE CERTIFICATE STATUS PROTOCOL WITH APACHE HTTP SERVER

Before you use Online Certificate Status Protocol OCSP) for https, ensure you have configured Apache HTTP Server for SSL connections (see Section 5.1, “Configuring Apache HTTP Server for SSL Connections”).

To use Online Certificate Status Protocol with Apache HTTP Server, ensure that a Certificate Authority (CA) and OCSP Responder is configured correctly.

For more information on how to configure a CA, see the *Managing Certificates and Certificate Authorities* section in the *Red Hat Enterprise Linux 7 Linux Domain Identity, Authentication, and Policy Guide*

For more information on how to configure an OCSP Responder, see the *Configuring OCSP Responders* section in the *Red Hat Enterprise Linux 7 Linux Domain Identity, Authentication, and Policy Guide*

NOTE

Ensure your Certificate Authority is capable of issuing OSCP Certificates. The Certificate Authority must be able to append the following attributes to the Certificate:

```
[ usr_cert ]
...
authorityInfoAccess=OCSP;URI:http://HOST:PORT
...
[ v3_OCSP ]
basicConstraints = CA:FALSE
keyUsage = nonRepudiation, digitalSignature, keyEncipherment
extendedKeyUsage = OCSP Signing
```

Note that HOST and PORT will need to be replaced with the details of the OCSP Responder that you will configure.

Report a bug

5.4. CONFIGURE APACHE HTTP SERVER TO VALIDATE OCSP CERTIFICATES

Before configuring Apache HTTP Server to validate OCSP certificates, ensure that a Certificate Authority (CA) and an OCSP Responder is configured correctly. The example below shows how to enable OCSP validation of client certificates:

Example 5.1.

Use the **`SSLOCSPEnable`** attribute to enable OCSP validation:

```
# Require valid client certificates (mutual auth)
```

```
SSLVerifyClient require
SSLVerifyDepth  3
# Enable OCSP
SSLOCSPEnable on
SSLOCSPDefaultResponder http://10.10.10.25:3456
SSLOCSPOverrideResponder on
```

Report a bug

5.5. VERIFY YOUR OCSP CONFIGURATION

You can use the OpenSSL command-line tool to verify your configuration:

```
# openssl ocsp -issuer cacert.crt -cert client.cert -url http://HOST:PORT
-CA ocsp_ca.cert -VAfile ocsp.cert
```

- ***-issuer*** is the Certificate Authority certificate.
- ***-cert*** is the Client certificate which you want to verify.
- ***-url*** is the http server validating Certificate (OCSP).
- ***-CA*** is the CA certificate for verifying the Apache HTTP Server server certificate.
- ***-VAfile*** is the OCSP Responder certificate.

Report a bug

CHAPTER 6. WORKING EXAMPLES

6.1. COMPLETE WORKING EXAMPLE

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

Load Balancer

A proxy server listening on localhost:

```
LoadModule proxy_cluster_module modules/mod_proxy_cluster.so
LoadModule cluster_slotmem_module modules/mod_cluster_slotmem.so
LoadModule manager_module modules/mod_manager.so
LoadModule advertise_module modules/mod_advertise.so

MemManagerFile cache/mod_cluster

<IfModule manager_module>
  Listen 6666
  <VirtualHost *:6666>
    <Directory />
      Require ip 127.0.0.1
    </Directory>
    ServerAdvertise on
    EnableMCPMReceive
    <Location /mod_cluster_manager>
      SetHandler mod_cluster-manager
      Require ip 127.0.0.1
   </Location>
  </VirtualHost>
</IfModule>
```

Worker Configuration for Tomcat

Edit ***JWS_HOME*/tomcat<*VERSION*>/conf/server.xml**, and add the following listener element to configure the worker for Tomcat:

```
<Listener
className="org.jboss.modcluster.container.catalina.standalone.ModClusterLi
stener" advertise="true"/>
```

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
```

Report a bug

6.2. MOD_AUTH_KERB EXAMPLE

This section contains instructions for a basic example for configuring Kerberos authentication with Red Hat JBoss Web Server's Apache HTTP Server and mod_auth_kerb on Red Hat Enterprise Linux.

Report a bug

6.2.1. mod_auth_kerb Example Prerequisites

The following is a list of prerequisites for the working example. Ensure that all prerequisites are met before attempting to use the example instructions.

- Install mod_auth_kerb on Red Hat Enterprise Linux.
- Install curl with GSS-negotiated support.
- Configure and run a Kerberos or LDAP server (for example ApacheDS) on the same host as your Red Hat JBoss Web Server.
- Create the following LDAP users:
  - Create the user **`krbtgt`**:

    ```
    dn: uid=krbtgt,ou=Users,dc=example,dc=com
    objectClass: top
    objectClass: person
    objectClass: inetOrgPerson
    objectClass: krb5principal
    objectClass: krb5kdcentry
    cn: KDC Service
    sn: Service
    uid: krbtgt
    userPassword: secret
    krb5PrincipalName: krbtgt/EXAMPLE.COM@EXAMPLE.COM
    krb5KeyVersionNumber: 0
    ```

  - Create the user **`ldap`**:

    ```
    dn: uid=ldap,ou=Users,dc=example,dc=com
    objectClass: top
    objectClass: person
    objectClass: inetOrgPerson
    objectClass: krb5principal
    objectClass: krb5kdcentry
    cn: LDAP
    sn: Service
    uid: ldap
    userPassword: randall
    krb5PrincipalName: ldap/localhost@EXAMPLE.COM
    krb5KeyVersionNumber: 0
    ```

  - Create the user **HTTP**:

```
dn: uid=HTTP,ou=Users,dc=example,dc=com
objectClass: top
objectClass: person
objectClass: inetOrgPerson
objectClass: krb5principal
objectClass: krb5kdcentry
cn: HTTP
sn: Service
uid: HTTP
userPassword: secretpwd
krb5PrincipalName: HTTP/localhost@EXAMPLE.COM
krb5KeyVersionNumber: 0
```

- Create user **hnelson** (test user):

```
dn: uid=hnelson,ou=Users,dc=example,dc=com
objectClass: top
objectClass: person
objectClass: inetOrgPerson
objectClass: krb5principal
objectClass: krb5kdcentry
cn: Horatio Nelson
sn: Nelson
uid: hnelson
userPassword: secret
krb5PrincipalName: hnelson@EXAMPLE.COM
krb5KeyVersionNumber: 0
```

Report a bug

6.2.2. Configure the Kerberos Client

Use the following procedure to configure a Kerberos client for testing purposes:

Procedure 6.1. Configure the Kerberos Client

1. **Create the Kerberos Configuration File**
   Create the **krb5.conf** configuration file in the **/etc** directory and add the following to the file:

```
[logging]
  default = FILE:/var/log/krb5libs.log
  kdc = FILE:/var/log/krb5kdc.log
  admin_server = FILE:/var/log/kadmind.log

[libdefaults]
  default_realm = EXAMPLE.COM
  default_tgs_enctypes = des-cbc-md5,des3-cbc-sha1-kd
  default_tkt_enctypes = des-cbc-md5,des3-cbc-sha1-kd
  dns_lookup_realm = false
  dns_lookup_kdc = false
  allow_weak_crypto = yes
  ticket_lifetime = 24h
  renew_lifetime = 7d
  forwardable = yes

[realms]
  EXAMPLE.COM = {
    kdc = localhost:60088
    admin_server = localhost:60088
  }

[domain_realm]
  .example.com = EXAMPLE.COM
  example.com = EXAMPLE.COM
```

2. **Create a Key Tab**
   Create a key tab in the **`/etc/httpd`** folder with the following contents:

```
# ktutil
ktutil: addent -password -p HTTP/localhost@EXAMPLE.COM -k 0 -e des-
cbc-md5
Password for HTTP/localhost@EXAMPLE.COM: secretpwd
ktutil: list
slot KVNO Principal
---- ---- -------------------------------------------------------
--------------
   1    0               HTTP/localhost@EXAMPLE.COM
ktutil: wkt krb5.keytab
ktutil: quit
```

   As the root user, run the following commands to apply the correct group and permissions to the key tab:

```
# chgrp apache /etc/httpd/krb5.keytab
# chmod 640 /etc/httpd/krb5.keytab
```

3. **Check the Hosts File**
   Ensure that the following host configuration is included in the **`/etc/hosts`** file:

```
127.0.0.1 localhost
```

Report a bug

6.2.3. Configure mod_auth_kerb

Use the following procedure to configure mod_auth_kerb. As a prerequisite, ensure that the Kerberos Client is configured (see Section 6.2.2, "Configure the Kerberos Client").

Procedure 6.2. Configure mod_auth_kerb

- Create the **`auth_kerb.conf`** configuration file in the ***`JWS_HOME`*`/httpd/conf.d/`** folder and add the following information to the file:

```
#
# The mod_auth_kerb module implements Kerberos authentication over
# HTTP, following the "Negotiate" protocol.
```

```
#

LoadModule auth_kerb_module modules/mod_auth_kerb.so

<Location /kerberostest>
  AuthType Kerberos
  AuthName "Kerberos Login"
  KrbMethodNegotiate On
  KrbMethodK5Passwd Off
  KrbAuthRealms EXAMPLE.COM
  KrbServiceName HTTP
  Krb5KeyTab /etc/httpd/krb5.keytab
  require valid-user
</Location>
```

Report a bug

6.2.4. Test the Kerberos Authentication

Use the following instructions to test the Kerberos authentication. As a prerequisite for this procedure, ensure that the Kerberos Client is configured (see Section 6.2.2, “Configure the Kerberos Client”).

Procedure 6.3. Test the Kerberos Authentication

1. **Create a Test Page**
   Create a test page named **auth_kerb_page.html** in ***JWS_HOME*/httpd/www/html/kerberostest/**.

2. **Add the Contents of the Test Page**
   Add the following contents to the test page (**auth_kerb_page.html**):

   ```
   <html>
   <body>
       <h1>mod_auth_kerb successfully authenticated!</h1>
   </body>
   </html>
   ```

3. **Optional: Set Log Level**
   Optionally, set the log level for debugging in ***JWS_HOME*/httpd/conf/httpd.conf**.

4. **Start httpd**
   The the *Installation Guide* for details.

5. **Test Authentication**
   Test the authentication as follows:

   a. Initiate Kerberos authentication for the test user **hnelson**:

      ```
      $ kinit hnelson
      ```

   b. View the details for the test user **hnelson**:

      ```
      $ klist
      ```

A result similar to the following appears:

```
Ticket cache: FILE:/tmp/krb5cc_18602
Default principal: hnelson@EXAMPLE.COM

Valid starting     Expires            Service principal
06/03/13 14:21:13  06/04/13 14:21:13
krbtgt/EXAMPLE.COM@EXAMPLE.COM
renew until 06/10/13 14:21:13
```

c. **Testing Apache HTTP Server Kerberos Authentication**
Test Apache HTTP Server Kerberos authentication as follows:

```
$ curl --negotiate -u :
http://localhost/kerberostest/auth_kerb_page.html
```

If working correctly, the following result appears:

```
<html>
<body>
    <h1>mod_auth_kerb successfully authenticated!</h1>
</body>
</html>
```

See http://modauthkerb.sourceforge.net/ for more information about mod_auth_kerb.

Report a bug

APPENDIX A. REFERENCE

A.1. APACHE MODULES

This section contains expanded definitions of the Apache proxy server modules discussed in Section 3.1.2, “Components” .

Report a bug

A.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
```

Configurable directives in the **`<VirtualHost>`** element are as follows:

EnableMCPMReceive

Allows the ***`VirtualHost`*** to receive the mod_cluster Protocol Message (MCPM) from nodes. Add one ***`EnableMCPMRecieve`*** directive to the httpd configuration to allow ***`mod_cluster`*** to operate correctly. ***`EnableMCPMRecieve`*** must be added in the ***`VirtualHost`*** configuration, at the location where ***`advertise`*** is configured.

MaxMCMPMaxMessSize

Defines the maximum size of mod_cluster Management Protocol (MCMP) messages. The default value for this is calculated from other **`Max`** directives. The minimum value for this is **`1024`**.

AllowDisplay

Toggles the additional display on the ***`mod_cluster-manager`*** main page. The default value is **`off`**, which causes only version information to display on the ***`mod_cluster-manager`*** main page.

AllowCmd

Toggles permissions for commands using ***`mod_cluster-manager`*** URL. The default value is **`on`**, which allows commands.

ReduceDisplay

Toggles the reduction of information displayed on the ***`mod_cluster-manager`*** page. Reducing the information allows more nodes to display on the page. The default value is **`off`** which allows all the available information to display.

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 mod_cluster will use. The default value is **`100`**.

Maxnode

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

Maxhost

The maximum number of hosts (aliases) 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. This is used for demonstration and debugging purposes only. 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 mod_cluster-manager

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
  Require ip 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`**.)

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`**.

Report a bug

A.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, the 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 configure the following 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 **`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)$ !
```

Report a bug

A.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 is configurable using 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* must correspond to **`AdvertiseGroupAddress`**, and *port* must correspond to **`AdvertisePort`** in your worker nodes.

If your worker node is JBoss EAP-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 a port is not specified, the port defaults to **`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 mod_cluster in JBoss Web Server. 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`.

Report a bug

A.1.4. mod_proxy.so

mod_proxy.so is a standard Apache HTTP Server module. This module lets the server act as proxy for data transferred over AJP (Apache JServe Protocol), FTP, CONNECT (for SSL), and HTTP. This module does not require additional configuration. Its identifier is `proxy_module`.

`Mod_proxy` directives such as *`ProxyIOBufferSize`* are used to configure *`mod_cluster`*.

Report a bug

A.1.5. mod_proxy_ajp.so

mod_proxy_ajp.so is a standard Apache HTTP Server module that provides support for AJP (Apache JServe Protocol) proxying. `Mod_proxy.so` is required to use this module.

Report a bug

A.1.6. mod_cluster_slotmem

`mod_cluster_slotmem` does not require any configuration directives.

Report a bug

A.2. WORKERS.PROPERTIES

Apache HTTP Server worker nodes are Servlet containers that are mapped to the **mod_jk** load balancer. The worker nodes are defined in *`JWS_HOME`*`/httpd/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 lowercase letters, uppercase letters, numbers, and specific special characters (_, /).

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, see Apache Tomcat Connector - Reference Guide

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]`***.

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, see The Apache Tomcat Connector - AJP Protocol Reference

workers.properties Connection Directives

host

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

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

port

The port number of the remote server instance listening for defined protocol requests. The default value is `8009`, which is the default listen port for AJP13 workers. If you are using AJP14 workers, the default port is `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.

ping_timeout

Specifies the time to wait for CPong answers to a CPing connection probe (see ***`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.

Report a bug

APPENDIX B. JAVA PROPERTIES REFERENCE

B.1. PROXY CONFIGURATION

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.

Table B.1. Proxy Configuration Values for Tomcat

Value	Default	Description
stickySession	true	Specifies whether subsequent requests for a given session should be routed to the same node, if possible.
stickySessionRemove	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	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	-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 unusable, **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	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.

Value	Default	Description
flushPackets	false	Specifies whether packet flushing is enabled or disabled.
flushWait	-1	Specifies the time to wait before flushing packets. A value of `-1` means wait forever.
ping	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	60	Specifies the time (in seconds) idle connections persist, above the *smax* threshold.
nodeTimeout	-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	mycluster	Specifies the name of the load-balancer.
loadBalancingGroup	-	Specifies the load balancing among *jvmRoutes* within the same load balancing group. A *loadBalancingGroup* is conceptually equivalent to a *mod_jk* domain directive.

Report a bug

B.2. MOD_CLUSTER PROXY AND PROXY DISCOVERY CONFIGURATION ATTRIBUTES

The following tables contain attributes and information about mod_cluster proxy, and proxy discovery configuration attributes.

Table B.2. mod_cluster Proxy Discovery Configuration Attributes

Attribute	Property	Default Value
proxy-list	proxyList	-
proxy-url	proxyURL	-
advertise	advertise	true
advertise-security-key	advertiseSecurityKey	-
excluded-contexts	excludedContexts	-
auto-enable-contexts	autoEnableContexts	true
stop-context-timeout	stopContextTimeout	10 seconds (in seconds)
socket-timeout	nodeTimeout	20 seconds (in milliseconds)

NOTE

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

Table B.3. mod_cluster Proxy Configuration Attributes

Attribute	Property	Default Value
sticky-session	stickySession	true
sticky-session-remove	stickySessionRemove	false
sticky-session-force	stickySessionForce	true
node-timeout	workerTimeout	-1
max-attempts	maxAttempts	1
flush-packets	flushPackets	false
flush-wait	flushWait	-1
ping	ping	10
smax	smax	-1 (uses the default value)

Attribute	Property	Default Value
ttl	ttl	-1 (uses the default value)
domain	loadBalancingGroup	-
load-balancing-group	loadBalancingGroup	-

Report a bug

B.3. LOAD CONFIGURATION

The following table contains additional configuration properties that are used when **`mod_cluster`** is configured with Tomcat:

Table B.4. Load Configuration for Tomcat

Attribute	Default Value	Description
loadMetricClass	org.jboss.modcluster.load.metric.impl.BusyConnectorsLoadMetric	This is the class name of an object that is implementing org.jboss.load.metric.LoadMetric.
loadMetricCapacity	1	This is the capacity of the load metric defined via the ***`loadMetricClass`*** property.
loadHistory	9	This is the number of historic load values that must be considered in the load balance factor computation.
loadDecayFactor	2	This is the factor by which the historic load values decrease in significance.

Report a bug

APPENDIX C. REVISION HISTORY

Revision 3.0.1-15	**Friday 4 September 2015**	**Lucas Costi**
Red Hat JBoss Web Server 3.0.1 GA.		
Revision 3.0.0-12	**Tue 5 May 2015**	**Lucas Costi**
Red Hat JBoss Web Server 3.0 GA.		