Red Hat JBoss Core Services 2.4.37

Apache HTTP Server Connectors and Load Balancing Guide

For Use with Red Hat JBoss Core Services 2.4.37

Last Updated: 2022-08-24

Red Hat JBoss Core Services 2.4.37 Apache HTTP Server Connectors and Load Balancing Guide

For Use with Red Hat JBoss Core Services 2.4.37

Legal Notice

Abstract

Install and configure load-balancing solutions that use the `mod_cluster` and `mod_jk` connector modules along with other modules that Red Hat JBoss Core Services provides.

Table of Contents

PROVIDING FEEDBACK ON RED HAT DOCUMENTATION

We appreciate your feedback on our technical content and encourage you to tell us what you think. If you'd like to add comments, provide insights, correct a typo, or even ask a question, you can do so directly in the documentation.

NOTE

You must have a Red Hat account and be logged in to the customer portal.

To submit documentation feedback from the customer portal, do the following:

1. Select the **Multi-page HTML** format.
2. Click the **Feedback** button at the top-right of the document.
3. Highlight the section of text where you want to provide feedback.
4. Click the **Add Feedback** dialog next to your highlighted text.
5. Enter your feedback in the text box on the right of the page and then click **Submit**.

We automatically create a tracking issue each time you submit feedback. Open the link that is displayed after you click **Submit** and start watching the issue or add more comments.

Thank you for the valuable feedback.

MAKING OPEN SOURCE MORE INCLUSIVE

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

CHAPTER 1. HTTP CONNECTOR MODULES

Red Hat JBoss Core Services includes two different HTTP connector modules that you can use to load-balance HTTP requests to a set of back-end servlet containers:

- The Apache Tomcat connector (**mod_jk**) supports the load balancing of HTTP requests to a set of servlet containers, while maintaining sticky sessions and communicating over the Apache JServ Protocol (AJP).
- The JBoss HTTP connector (**mod_cluster**) is a more advanced load balancer than **mod_jk**. The **mod_cluster** module provides all the functionality of **mod_jk** and additional features such as real-time load-balancing calculations, application life-cycle control, automatic proxy discovery, and multiple protocol support.

The Apache HTTP Server Connectors and Load Balancing Guide describes how to install and configure the **mod_proxy** and **mod_cluster** connectors. This guide also provides information about Online Certificate Status Protocol (OCSP) and includes 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 an archive installation of JBoss Core Services on Red Hat Enterprise Linux. For other platforms, use the correct paths for your respective installation, as specified in the JBoss Core Services *Installation Guide*.

IMPORTANT

Red Hat Enterprise Linux 6 is no longer supported and subsequently was removed from the documentation.

Additional resources

- Load balancing with the Apache Tomcat connector (**mod_jk**)
- Load balancing with the JBoss HTTP connector (**mod_cluster**)

CHAPTER 2. LOAD BALANCING WITH THE APACHE TOMCAT CONNECTOR (MOD_JK)

The Apache Tomcat Connector, **mod_jk**, is a plug-in that allows the Apache HTTP Server to forward web requests to a back-end servlet container. The **mod_jk** module also allows the Apache HTTP Server to load-balance requests to a set of servlet containers, while maintaining sticky sessions.

2.1. MOD_JK INSTALLATION

The **mod_jk** module is included in the Apache HTTP Server part of a JBoss Core Services installation.

You can follow the procedures in the Red Hat JBoss Core Services Apache HTTP Server Installation Guide to download and install the Apache HTTP Server for your operating system.

Additional resources

- Red Hat JBoss Core Services Apache HTTP Server Installation Guide

2.2. APACHE HTTP SERVER LOAD-BALANCING CONFIGURATION WHEN USING MOD_JK

You can configure the Apache HTTP Server to use **mod_jk** to load-balance requests to a set of servlet containers. This setup includes the configuration of back-end worker nodes.

Red Hat JBoss Core Services provides example configuration files for **mod_jk** in the ***JBCS_HOME*/httpd/conf.d/** directory. These example configuration files are named **mod_jk.conf.sample**, **workers.properties.sample**, and **uriworkermap.properties.sample**. To use these examples instead of creating your own configuration files, you can remove the **.sample** extension, and modify the file content as needed.

NOTE

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

When you use the Load Balancer Configuration tool for Apache HTTP Server 2.4.37, ensure that you select **2.4.x** as the Apache version, and select **Tomcat** as the back-end configuration.

When the Apache HTTP Server (**httpd**) is installed on Red Hat Enterprise Linux 8, the base operating system modules are located in the **/usr/lib64/httpd/modules** directory. The Red Hat JBoss Core Services modules are currently located in the **/opt/rh/jbcs/root/usr/lib64/httpd/modules** directory.

The Red Hat JBoss Core Services modules include **mod_jk**, **mod_cluster**, **mod_rt**, and **mod_bmx**. These modules follow all Red Hat JBoss Core Services rules for naming, directories, and prefixes. If you want to use these modules, create or modify a configuration file to add the **LoadModule** command. For example:

```
LoadModule jk_module /opt/rh/jbcs/root/usr/lib64/httpd/modules/mod_jk.so
```

Alternatively, you can include the directory of the installed Red Hat JBoss Core Services modules in the ***JBCS_HOME*/httpd/conf.d** directory.

IMPORTANT

Consider the following differences between the **httpd** implementations that are provided by JBoss Core Services and Red Hat Enterprise Linux:

- You can install JBoss Core Services **httpd** from an archive file or RPM package.
- You can also install JBoss Core Services **httpd** in a Windows Server environment.
- JBoss Core Services **httpd** does not provide or support the **mod_php** module. Red Hat Enterprise Linux **httpd** supports the **mod_php** module.
- JBoss Core Services **httpd** provides the **mod_jk** and **mod_cluster** load balancer modules. Red Hat Enterprise Linux **httpd** does not provide the **mod_jk** and **mod_cluster** modules.

The use case for JBoss Core Services **httpd** is to connect to the back end with a proxy. You can use **mod_jk**, **mod_proxy_cluster**, or **mod_proxy** as a proxy. There is no difference between these modules in the **httpd** implementations that are provided by Red Hat JBoss Core Services and Red Hat Enterprise Linux.

NOTE

Since the 2.4.37 Service Pack 10 release, Red Hat JBoss Core Services does not support the tunneling of non-upgraded connections to a back-end WebSockets server. This means that when you are configuring the **ProxyPass** directive for the **mod_proxy_wstunnel** module, you must ensure that the upgrade parameter is not set to **NONE**. For more information about **mod_proxy_wstunnel**, see the Apache documentation.

2.3. CONFIGURING THE APACHE HTTP SERVER TO LOAD MOD_JK

You can configure the Apache HTTP Server to load **mod_jk**, by specifying configuration settings in the **mod_jk.conf** file.

You can also perform the following optional configuration steps:

- In addition to the **JkMount** directive, you can use the **JkMountFile** directive to specify the configuration file for a mount point. The configuration file contains multiple URL mappings for Tomcat forwarding.
- You can configure the Apache HTTP Server that is functioning as the load balancer to log details of each worker node that handles a request. This can be useful if you need to troubleshoot your load balancer.

Procedure

1. Go to the ***JBCS_HOME*/httpd/conf.d** directory.
2. Create a new file named **mod_jk.conf** and enter the following configuration details:

```
# 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 SSL 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

Ensure that the **LoadModule** directive references the **mod_jk** native binary that you have installed.

NOTE

The **JkMount** directive specifies the URLs that the Apache HTTP Server can forward to the **mod_jk** module. Based on the configuration for the **JkMount** directive, **mod_jk** forwards the received URL to the correct servlet containers.

To enable the Apache HTTP Server to serve static content (or PHP content) directly, and only use the load balancer for Java applications, the preceding configuration example specifies that the Apache HTTP Server sends only requests with the URL **/application/*** to the **mod_jk** load balancer.

Alternatively, you can configure the Apache HTTP Server to forward all URLs to **mod_jk** by specifying **/*** in the **JkMount** directive.

3. Optional: To use the **JkMountFile** directive to specify the configuration file for a mount point, perform the following steps:

   a. Go to the ***JBCS_HOME*/httpd/conf.d/** directory.

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

   c. Specify the URL that you want to forward and the worker name.
      For example:

```
# Simple worker configuration file

# Mount the Servlet context to the ajp13 worker
/application=loadbalancer
/application/*=loadbalancer
```

> **NOTE**
>
> The required syntax is in the format: /***URL=WORKER_NAME***
>
> The preceding example configures **mod_jk** to forward requests for **/application** to the JBoss Web Server Tomcat back end.

   d. In the ***JBCS_HOME*/httpd/conf.d/mod_jk.conf** file, enter 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
```

4. Optional: To enable Apache HTTP Server logging, perform either of the following steps:

   - Include **%w** in your **JkRequestLogFormat** directive, as shown in the preceding step about **mod_jk.conf** settings.

   - Log the name of the **mod_jk** worker that you want to use, by including **%{JK_WORKER_NAME}n** in your Apache HTTP Server **LogFormat**(s).

Additional resources

- The Apache Tomcat Connectors - Web Server HowTo : mod_jk Directives
- Apache HTTP Server Documentation: Log Files

2.4. CONFIGURING WORKER NODES IN MOD_JK

You can configure multiple worker nodes to handle the requests that the Apache HTTP Server forwards to the servlet containers.

The example in this procedure shows how to define two **mod_jk** worker nodes in a weighted round-robin configuration that uses sticky sessions between two servlet containers.

Prerequisites

- You are familiar with the format of the **workers.properties** directives.
- You have configured the Apache HTTP Server to load **mod_jk**.

Procedure

1. Go to the ***JBCS_HOME*/httpd/conf.d/** directory.
2. Create a file named **workers.properties**.
3. Enter the following configuration details:

```
# 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
worker.node1.secret=<YourSecret>

# 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
worker.node1.secret=<YourSecret>

# 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
```

NOTE

In the preceding example, ensure that you replace **host**, **port**, and **secret** settings with values that are relevant for your environment.

IMPORTANT

The **secret** property is required when using the Tomcat AJP Connector. You can specify the **secret** property for a worker node or a load balancer in the **workers.properties** file. For example:

worker.<*WORKER_NAME*>.secret=<*YOUR_AJP_SECRET*>

In the preceding example, replace ***<WORKER_NAME>*** and ***<YOUR_AJP_SECRET>*** with values that are relevant for your environment.

2.5. CONFIGURING TOMCAT TO WORK WITH MOD_JK

Tomcat is configured to receive Apache JServ Protocol (AJP) traffic from **mod_jk** by default. However, before you can use a worker node with **mod_jk**, you must perform the following additional configuration steps:

- Configure the AJP connector. The AJP connector is not configured by default.
- Configure a unique value for the **jvmRoute** attribute in the Engine of each worker node.
- Specify the **secret** property for a worker node or a load balancer. The **secret** property is required when you use the Tomcat AJP connector.

Procedure

1. To configure the AJP connector, perform the following steps:
   a. Open the ***JBCS_HOME*/tomcat*<VERSION>*/conf/server.xml** file.
   b. In the **server.xml** file, enter the following line:

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

2. To configure a unique value for the **jvmRoute** attribute in the Engine of each worker node, enter the following details:

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

   IMPORTANT

   Ensure that the **jvmRoute** attribute value matches the worker name that is specified in the **workers.properties** file.

3. To specify the **secret** property for a worker node or a load balancer, perform the following steps:
   a. Open the ***JBCS_HOME*/httpd/conf.d/workers.properties** file.
   b. In the **workers.properties** file, ensure that the **secret** property is specified in the following format:

   ```
   worker.<WORKER_NAME>.secret=<YOUR_AJP_SECRET>`
   ```

NOTE

Ensure that you replace ***<WORKER_NAME>*** and ***<YOUR_AJP_SECRET>*** with values that are relevant for your environment.

NOTE

If you set a **secret** on a load balancer by using the **ProxyPass** directive, all of its members inherit this **secret**. For example:

```
<Proxy balancer://mycluster>`
    BalancerMember ajp://node1:8009 route=node1
secret=YOUR_AJP_SECRET
    BalancerMember ajp://node2:8009 route=node2
secret=YOUR_AJP_SECRET
</Proxy>
ProxyPass /example/ balancer://mycluster/example/
stickysession=JSESSIONID|jsessionid
```

CHAPTER 3. LOAD BALANCING WITH THE JBOSS HTTP CONNECTOR (MOD_CLUSTER)

The **mod_cluster** connector is a reduced configuration, intelligent load-balancing solution for JBoss EAP and Apache HTTP Server Tomcat. The **mod_cluster** module is based on technology originally developed by the JBoss **mod_cluster** community project.

3.1. JBOSS HTTP CONNECTOR (MOD_CLUSTER)

The **mod_cluster** module load-balances HTTP requests to JBoss EAP and Apache HTTP Server Tomcat worker nodes. The **mod_cluster** module uses the Apache HTTP Server as the proxy server.

Key features of mod_cluster

The **mod_cluster** connector has several advantages over the **mod_jk** connector:

- When the **mod_cluster** module is enabled, the **mod_cluster** Management Protocol (MCMP) is an additional connection between the Tomcat servers and the Apache HTTP Server. The Tomcat servers use MCMP to transmit server-side load figures and lifecycle events back to the Apache HTTP Server, by using 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 the need for manual configuration.
- Tomcat servers perform the load calculations rather than rely on the Apache HTTP Server. This makes load-balancing metrics more accurate than other connectors.
- The **mod_cluster** connector provides fine-grained application lifecycle control. Each Tomcat server forwards web application context lifecycle events to the Apache HTTP Server. These lifecycle events include informing the Apache HTTP Server to start or stop routing requests for a specific context. This prevents end users from seeing HTTP errors because of unavailable resources.
- You can use Apache JServ Protocol (AJP), Hypertext Transfer Protocol (HTTP), or Hypertext Transfer Protocol Secure (HTTPS) transports with **mod_cluster**.

Mod_cluster components

On the proxy server, **mod_cluster** consists of four Apache modules:

Component	Description
mod_cluster_slotmem.so	The Shared Memory Manager module shares real-time worker node information with multiple Apache HTTP Server processes.
mod_manager.so	The Cluster Manager module receives and acknowledges messages from worker nodes, including node registrations, node load data, and node application life cycle events.

Component	Description
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, the 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 server is listening for responses from worker nodes that want to join the load-balancing cluster.

Additional resources

- Apache HTTP Server modules

3.2. MOD_CLUSTER CHARACTER LIMITS

The **mod_cluster** module 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 you define proxy server and worker node properties, ensure that you adhere to the following character limits:

Property	Maximum character limit	Description
Alias length	100 characters	Alias corresponds to the network name of the respective virtual host; the name is defined in the **Host** element.
Context length	40 characters	For example, if **myapp.war** is deployed in /**myapp** , /**myapp** is included in the context.
Balancer name length	40 characters	This is the balancer property in **mbean**.
JVMRoute string length	80 characters	**JVMRoute** in the **<Engine>** element.
Domain name length	20 characters	This is the domain property in **mbean**.
Hostname length for a node	64 characters	This is hostname address in the **<Connector>** element.

Property	Maximum character limit	Description
Port length for a node	7 characters	This is the port property in the **<Connector>** element. For example, **8009** is 4 characters.
Scheme length for a node	6 characters	This is the protocol of the connector. Possible values are **http**, **https**, and **ajp**.
Cookie name length	30 characters	This is the header cookie name for the session ID. The default value is **JSESSIONID** based on the **org.apache.catalina.Globals.SESSION_COOKIE_NAME** property.
Path name length	30 characters	This is the parameter name for the session ID. The default value is **JSESSIONID** based on the **org.apache.catalina.Globals.SESSION_PARAMETER_NAME** property.
Session ID length	120 characters	A session ID is in the following type of format: **BE81FAA969BF64C8EC2B6600457EAAAA.node01**

3.3. MOD_CLUSTER INSTALLATION

The **mod_cluster** module is included in the Apache HTTP Server part of a JBoss Core Services installation.

You can follow the procedures in the Red Hat JBoss Core Services Apache HTTP Server Installation Guide to download and install the Apache HTTP Server for your operating system.

Additional resources

- Red Hat JBoss Core Services Apache HTTP Server Installation Guide

3.4. APACHE HTTP SERVER LOAD-BALANCING CONFIGURATION WHEN USING MOD_CLUSTER

In Apache HTTP Server 2.1 and higher, **mod_cluster** is configured correctly for Apache HTTP Server by default. For more information about setting a custom configuration, see Configuring a basic proxy server.

NOTE

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

When you use the Load Balancer Configuration tool for Apache HTTP Server 2.4.37, ensure that you select **2.4.x** as the Apache version, and select **Tomcat** as the back-end configuration.

When the Apache HTTP Server (**httpd**) is installed on Red Hat Enterprise Linux 8, the base operating system modules are located in the **/usr/lib64/httpd/modules** directory. The Red Hat JBoss Core Services modules are currently located in the **/opt/rh/jbcs/root/usr/lib64/httpd/modules** directory.

The Red Hat JBoss Core Services modules include **mod_jk**, **mod_cluster**, **mod_rt**, and **mod_bmx**. These modules follow all Red Hat JBoss Core Services rules for naming, directories, and prefixes. If you want to use these modules, create or modify a configuration file to add the **LoadModule** command. For example:

```
LoadModule jk_module /opt/rh/jbcs/root/usr/lib64/httpd/modules/mod_jk.so
```

Alternatively, you can include the directory of the installed Red Hat JBoss Core Services modules in the ***JBCS_HOME*/httpd/conf.d** directory.

NOTE

- When you want to use the **mod_proxy_cluster** module, you must enable the **mod_proxy** module and disable the **mod_proxy_balancer** module.
- If you want **mod_proxy_cluster** to use the Apache JServ Protocol (AJP), you must enable the **proxy_ajp_module**.
- Use AJPSecret **your_secret** to provide the secret for the AJP backend. If **your_secret** does not correspond to the value configured in the back end, the back end sends a **503** error response for any request that is sent through the proxy.

NOTE

Since the 2.4.37 Service Pack 10 release, Red Hat JBoss Core Services does not support the tunneling of non-upgraded connections to a backend websockets server. This means that when you are configuring the ProxyPass directive for the mod_proxy_wstunnel module, you must ensure that the upgrade parameter is not set to NONE. For more information about mod_proxy_wstunnel, see the Apache documentation.

Additional resources

- Configuring a basic proxy server
- Configuring a Tomcat worker node in **mod_cluster**

3.5. CONFIGURING A BASIC PROXY SERVER

You can configure the Apache HTTP Server to function as a proxy server that forwards requests and responses between web clients and back-end web servers. You must configure a proxy server listener to receive connection requests and responses from the back-end worker nodes. When you want to configure a load-balancing proxy server that uses **mod_cluster**, you must also configure a virtual host for the management channel.

Prerequisites

- You have installed the Apache HTTP Server and configured the **mod_cluster** modules for your installation. For more information, see the Red Hat JBoss Core Services Apache HTTP Server Installation Guide.
- The port that you specify for the proxy server listener must be open for incoming TCP connections.

Procedure

1. Open the **mod_cluster** configuration file.

   NOTE

   The **mod_cluster** configuration file is typically located in the ***JBCS_HOME*/httpd/conf.d/mod_cluster.conf** directory.

2. To create a **Listen** directive for the proxy server, enter the following line in the **mod_cluster.conf** file:

   ```
   Listen IP_ADDRESS:PORT_NUMBER
   ```

   NOTE

   In the preceding example, replace ***IP_ADDRESS*** with the address of the server network interface that the proxy server uses to communicate with the worker nodes, and replace ***PORT_NUMBER*** with the port that the proxy server listens on.

   Ensure that the port is open for incoming TCP connections.

3. To create a virtual host, enter the following details in the **mod_cluster.conf** file:

   ```
   <VirtualHost IP_ADDRESS:PORT_NUMBER>

     <Directory />
       Require ip IP_ADDRESS
     </Directory>

     KeepAliveTimeout 60
     MaxKeepAliveRequests 0

     ManagerBalancerName mycluster
     AdvertiseFrequency 5
     EnableMCPMReceive On

   </VirtualHost>
   ```

NOTE

In the preceding example, replace ***IP_ADDRESS*** and ***PORT_NUMBER*** with the address of the server network interface and port number that you have specified for the **Listen** directive.

This address and port combination is only used for **mod_cluster** management messages. This address and port combination is not used for general traffic.

For more information about configuring **mod_jk** and starting the Apache HTTP Server service, see the Red Hat JBoss Core Services Apache HTTP Server Installation Guide .

Additional resources

- Red Hat JBoss Core Services Apache HTTP Server Installation Guide

3.5.1. Disabling server advertisement

The proxy server uses UDP multicast to advertise itself. The **AdvertiseFrequency** directive instructs the server to send server advertisement messages every 10 seconds by default. Server advertisement messages contain the ***IP_ADDRESS*** and ***PORT_NUMBER*** that you specify in the **VirtualHost** definition. Worker nodes that are configured to respond to server advertisements use this information to register themselves with the proxy server. If you want to prevent worker nodes from registering with the proxy server, you can optionally disable server advertisement.

NOTE

When UDP multicast is available between the proxy server and the worker nodes, server advertisement adds worker nodes without requiring further configuration on the proxy server. Server advertisement requires only minimal configuration on the worker nodes.

Prerequisites

- You have configured a basic proxy server.

Procedure

1. Open the **mod_cluster** configuration file.

   NOTE

   The **mod_cluster** configuration file is typically located in the ***JBCS_HOME*/httpd/conf.d/mod_cluster.conf** directory.

2. Add the following directive to the **VirtualHost** definition:

```
ServerAdvertise Off
```

NOTE

If server advertisements are disabled, or UDP multicast is not available on the network between the proxy server and the worker nodes, you can configure worker nodes with a static list of proxy servers. In either case, you do not need to configure the proxy server with a list of worker nodes.

Additional resources

- Configuring worker nodes with a static list of proxy servers

3.5.2. Logging worker node details

When you configure a load-balancing proxy server that uses **mod_cluster**, you can optionally configure the Apache HTTP Server to log details of each worker node that handles a request. Logging worker node details can be useful if you need to troubleshoot your load balancer.

Prerequisites

- You have configured a basic proxy server.

Procedure

1. Open the **mod_cluster** configuration file.

   NOTE

   The **mod_cluster** configuration file is typically located in the ***JBCS_HOME*/httpd/conf.d/mod_cluster.conf** directory.

2. Add the following details 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.

Additional resources

- Apache HTTP Server documentation about log files

3.6. CONFIGURING A TOMCAT WORKER NODE IN MOD_CLUSTER

When you use **mod_cluster**, you can configure a Tomcat worker node as an Apache HTTP Server Tomcat service that operates in non-clustered mode only. In this situation, only one load metric can be used at a time when calculating the load-balance factor.

NOTE

Apache HTTP Server Tomcat worker nodes support only a subset of **mod_cluster** functionality. Full **mod_cluster** functionality is available with JBoss EAP.

Prerequisites

- You have installed a supported instance of Apache HTTP Server .
- You are familiar with the **mod_cluster** proxy and proxy discovery configuration parameters.

Procedure

1. To add a listener to Tomcat, add the following **Listener** element beneath the other **Listener** elements in the ***JWS_HOME*/tomcat*<VERSION>*/conf/server.xml** file:

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

2. To give the worker node a unique identity, edit the ***JWS_HOME*/tomcat*<VERSION>*/conf/server.xml** file, to add the **jvmRoute** attribute and value to the **Engine** element:

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

3. To configure **STATUS MCMP** message frequency, modify the **org.jboss.modcluster.container.catalina.status-frequency** Java system property.
   For example:

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

   NOTE

   Tomcat worker nodes periodically send status messages that contain 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 system property. By default, the property accepts values that are specified in seconds multiplied by 10. For example, setting the property to **1** means 10 seconds. In the preceding example, the property is set to **6**, which means 60 seconds.

4. Optional: To configure the firewall for proxy server advertisements, complete either of the following steps to open port **23364** for UDP connections on the worker node's firewall:

   - 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"'
```

NOTE

When a proxy server uses **mod_cluster**, the proxy server can use UDP multicast to advertise itself. Most operating system firewalls block the server advertisement feature by default. To enable server advertisement and receive these multicast messages, you can open port **23364** for UDP connections on the worker node's firewall, as shown in the preceding examples.

IMPORTANT

Red Hat Enterprise Linux 6 is no longer supported and subsequently was removed from the documentation.

3.7. CONFIGURING A WORKER NODE TO OPERATE WITH A STATIC LIST OF PROXY SERVERS

Server advertisement allows worker nodes to discover and register themselves with proxy servers dynamically. If UDP multicast is not available or server advertisement is disabled, you must configure Apache HTTP Server worker nodes with a static list of proxy server addresses and ports.

Prerequisites

- You have configured an Apache HTTP Server worker node .
- You are familiar with the proxy configuration parameters for Tomcat.

Procedure

1. Open the ***JWS_HOME*/tomcat*<VERSION>*/conf/server.xml** file.
2. To define a **mod_cluster** listener and disable dynamic proxy discovery, add or change the **Listener** element for **ModClusterListener**.
   For example:

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

   NOTE

   Ensure that you set the **advertise** property to **false**.

3. To create a static proxy server list, update the **proxyList** property by adding a comma-separated list of proxies in the following format: ***IP_ADDRESS*:*PORT*,*IP_ADDRESS*:*PORT***
   For example:

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

CHAPTER 4. SECURING CONNECTIONS BY USING OCSP

Online Certificate Status Protocol (OCSP) is a technology that 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.

4.1. ONLINE CERTIFICATE STATUS PROTOCOL

When a web browser and a web server communicate over a secured connection, the server presents a set of credentials in the form of a certificate. The browser then validates the certificate and sends a request for certificate status information. The server responds with a certificate status of current, expired, or unknown.

The certificate contains the following types of information:

- Syntax for communication
- Control information such as start time, end time, and address information to access an Online Certificate Status Protocol (OCSP) responder.

The web server uses an OCSP responder to check the certificate status. You can configure the web server to use the OCSP responder that is listed in the certificate or another OCSP responder. OCSP allows a grace period for expired certificates, which allows access to a server for a limited time before renewing the certificate.

OCSP overcomes limitations of the older Certificate Revocation List (CRL) method.

Additional resources

- Red Hat Certificate System *Planning, Installation, and Deployment Guide*.

4.2. CONFIGURING THE APACHE HTTP SERVER FOR SSL CONNECTIONS

You can configure the Apache HTTP Server to support SSL connections, by installing the **mod_ssl** package and specifying configuration settings in the **ssl.conf** file.

Prerequisites

- You have generated an SSL certificate and private key.
- You know the location of the SSL certificate and private key file.
- You have obtained the Common Name (CN) that is associated with the SSL certificate.

Procedure

1. To install **mod_ssl**, enter the following command:

```
# yum install jbcs-httpd24-mod_ssl
```

2. To specify SSL configuration settings:

a. Open the ***JBCS_HOME*/httpd/conf.d/ssl.conf** file.

b. Enter details for the **ServerName**, **SSLCertificateFile**, and **SSLCertificateKeyFile**.
For example:

```
<VirtualHost _default_:443>
ServerName www.example.com:443
SSLCertificateFile /opt/rh/jbcs-httpd24/root/etc/pki/tls/certs/localhost.crt
SSLCertificateKeyFile /opt/rh/jbcs-httpd24/root/etc/pki/tls/private/localhost.key
```

NOTE

- The **ServerName** must match the Common Name (CN) that is associated with the SSL certificate. If the **ServerName** does not match the CN, client browsers display domain name mismatch errors.
- The **SSLCertificateFile** specifies the path to the SSL certificate file.
- The **SSLCertificateKeyFile** specifies the path to the private key file that is associated with the SSL certificate.

3. Verify that the **Listen** directive matches the hostname or IP address for the **httpd** service for your deployment.

4. To restart the Apache HTTP Server, enter the following command:

```
# service jbcs-httpd24-httpd restart
```

4.3. USING OCSP WITH THE APACHE HTTP SERVER

You can use the Online Certificate Status Protocol (OCSP) for secure connections with the Apache HTTP Server.

Prerequisites

- You have configured Apache HTTP Server for SSL connections.

Procedure

1. Configure a certificate authority.

NOTE

Ensure that your CA can issue OCSP certificates. The CA 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
```

In the preceding example, replace ***HOST*** and ***PORT*** with the details of the OCSP responder that you will configure.

2. Configure an OCSP responder.

Additional resources

- Managing certificates and certificate authorities.
- Configuring OCSP responders.

4.4. CONFIGURING THE APACHE HTTP SERVER TO VALIDATE OCSP CERTIFICATES

You can configure the Apache HTTP Server to validate OCSP certificates, by defining OCSP settings in the **ssl_conf** file.

Prerequisites

- You have configured a Certificate Authority (CA) .
- You have configured an OCSP Responder.

Procedure

1. Open the ***JBCS_HOME*/httpd/conf.d/ssl.conf** file.
2. Specify the appropriate OCSP configuration details for your deployment.
   For example:

```
# Require valid client certificates (mutual auth)
SSLVerifyClient require
SSLVerifyDepth  3
# Enable OCSP
SSLOCSPEnable on
SSLOCSPDefaultResponder http://<HOST>:<PORT>
SSLOCSPOverrideResponder on
```

NOTE

The preceding example shows how to enable OCSP validation of client certificates. In the preceding example, replace ***<HOST>*** and ***<PORT>*** with the IP address and port of the default OCSP Responder.

4.5. VERIFYING THE OCSP CONFIGURATION FOR THE APACHE HTTP SERVER

You can use the OpenSSL command-line tool to verify the OCSP configuration for the Apache HTTP Server.

Procedure

- On the command line, enter the **openssl** command in the following format:

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

  In the preceding command, ensure that you specify the following details:

  - Use the **-issuer** option to specify the CA certificate.
  - Use the **-cert** option to specify the client certificate that you want to verify.
  - Use the **-url** option to specify the HTTP server validating Certificate (OCSP).
  - Use the **-CA** option to specify the is the CA certificate for verifying the Apache HTTP Server server certificate.
  - Use the **-VAfile** option to specify the OCSP responder certificate.

CHAPTER 5. CONFIGURATION EXAMPLE FOR LOAD-BALANCING WITH MOD_CLUSTER

You can configure JBoss Core Services to use the **mod_cluster** connector for load-balancing in a Red Hat Enterprise Linux system.

When you want to configure a load-balancing solution that uses **mod_cluster**, you must perform the following tasks:

1. Set up JBoss Core Services as a proxy server.
2. Configure a Tomcat worker node.
3. Define iptables firewall rules.

5.1. SETTING UP JBOSS CORE SERVICES AS A PROXY SERVER

When you configure JBoss Core Services to use **mod_cluster**, you must set up JBoss Core Services as a proxy server by specifying configuration details in the **mod_cluster.conf** file.

Procedure

1. Go to the ***JBCS_HOME*****/httpd/conf.d/** directory.
2. Create a file named **mod_cluster.conf**.
3. Enter the following configuration details:

```
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>
```

NOTE

The preceding example shows how to set up JBoss Core Services as a proxy server that is listening on **localhost**.

5.2. CONFIGURING A TOMCAT WORKER NODE

When you configure JBoss Core Services to use **mod_cluster**, you must configure a Tomcat worker node by adding a **Listener** element to the **server.xml** file.

Prerequisites

- You have set up JBoss Core Services as a proxy server.

Procedure

1. Open the ***JWS_HOME*/tomcat*<VERSION>*/conf/server.xml** file.
2. Add the following **Listener** element:

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

5.3. DEFINING IPTABLES FIREWALL RULES EXAMPLE

When you configure JBoss Core Services to use **mod_cluster**, you must define firewall rules by using **iptables**.

Prerequisites

- You have configured a Tomcat worker node.

Procedure

- Use **iptables** to define a set of firewall rules.
  For example:

```
/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
```

NOTE

The preceding example shows to define firewall rules for a cluster node on the **192.168.1.0/24** subnet.

CHAPTER 6. CONFIGURATION EXAMPLE FOR KERBEROS AUTHENTICATION WITH MOD_AUTH_KERB

You can configure Kerberos authentication with the JBoss Core Services Apache HTTP Server and the **mod_auth_kerb** module on Red Hat Enterprise Linux.

When you want to configure Kerberos authentication, you must perform the following tasks:

1. Configure the Kerberos client.
2. Configure **mod_auth_kerb**.
3. Test the Kerberos authentication.

6.1. PREREQUISITES

- You have installed the **curl** command-line utility with Generic Security Services (GSS)-negotiated support.
- You have configured and run a Kerberos or LDAP server, such as ApacheDS, on the same host as JBoss Core Services.
- If you are using an LDAP server, you have created LDAP users called **krbtgt**, **ldap**, **HTTP**, and a test LDAP user called **hnelson**.
  - To create the **krbtgt** user, enter the following details:

```
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
```

  - To create the **ldap** user, enter the following details:

```
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
```

- To create the **HTTP** user, enter the following details:

```
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
```

- To create the test user, **hnelson**, enter the following details:

```
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
```

6.2. CONFIGURING THE KERBEROS CLIENT

When you configure Kerberos authentication, you must configure the Kerberos client by performing the following steps:

1. Specify configuration settings in the **krb5.conf** file.
2. Create a key tab in the ***JBCS_HOME*/httpd/conf** file.
3. Assign permissions to the key tab.
4. Ensure that **localhost** is included in the **/etc/hosts** file.

Prerequisites

- You are compliant with all prerequisites for configuring Kerberos authentication.

Procedure

1. To specify configuration settings in the **krb5.conf** file:
   a. Go to the **/etc** directory.
   b. Create a file named **krb5.conf**.

c. Enter the following configuration details:

```
[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. To create a key tab:

a. Open the ***JBCS_HOME*/httpd/conf** file.

b. Enter the following details:

```
# 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 JBCS_HOME/httpd/conf/krb5.keytab
ktutil: quit
```

IMPORTANT

Environment variables are not expanded within the **ktutil** prompt. You must substitute the full path for the **JBCS_HOME** variable.

3. To apply the correct group and permissions to the key tab, enter the following commands as the root user:

```
# chgrp apache JBCS_HOME/httpd/conf/krb5.keytab
# chmod 640 JBCS_HOME/httpd/conf/krb5.keytab
```

4. Ensure that the following host configuration is included in the **/etc/hosts** file:

```
127.0.0.1 localhost
```

6.3. CONFIGURING MOD_AUTH_KERB

When you configure Kerberos authentication, you must configure **mod_auth_kerb** by specifying settings in the **auth_kerb.conf** file.

Prerequisites

- You have configured the Kerberos client.

Procedure

1. Go to the ***JBCS_HOME*/httpd/conf.d/** directory.
2. Create a file named **auth_kerb.conf**.
3. Enter the following configuration details:

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

# The LoadModule statement is done in conf.d/10-auth_kerb.conf
# 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 $JBCS_HOME/httpd/krb5.keytab
  require valid-user
</Location>
```

IMPORTANT

Environment variables are not expanded within the configuration files. In the preceding example, ensure that you substitute the full path for the **JBCS_HOME** variable.

6.4. TESTING THE KERBEROS AUTHENTICATION

When you configure Kerberos authentication, you can use a test page to test the Kerberos authentication.

Prerequisites

- You have configured **mod_auth_kerb**.

Procedure

1. To create a test page, perform the following steps:

   a. Go to the ***JBCS_HOME*/httpd/www/html/kerberostest** directory.

   b. Create a test page named **auth_kerb_page.html**.

   c. Enter the following details:

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

2. Optional: In the ***JBCS_HOME*/httpd/conf/httpd.conf** file, set the log level to debugging.

3. Start the Apache HTTP Server. For more information, see the Red Hat JBoss Core Services Apache HTTP Server Installation Guide.

4. To initiate Kerberos authentication for the test user, **hnelson**, enter the following command:

   ```
   $ kinit hnelson
   ```

5. To view the details for the test user, **hnelson**, enter the following command:

   ```
   $ klist
   ```

   The **klist** command produces the following type of output:

   ```
   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
   ```

Verification

- To test Kerberos authentication, enter the following command:

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

  If Kerberos authentication is working correctly, the **curl** command produces the following output:

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

6.5. ADDITIONAL RESOURCES (OR NEXT STEPS)

- Kerberos Module for Apache website.

APPENDIX A. APACHE HTTP SERVER PROXY MODULES

This section contains expanded definitions of the Apache HTTP Server proxy modules that **mod_cluster** includes.

A.1. MOD_MANAGER MODULE AND DIRECTIVES

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 for mod_manager

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 **EnableMCPMReceive** directive to the Apache HTTP Server configuration to allow **mod_cluster** to operate correctly. **EnableMCPMReceive** 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 is calculated from other **Max** directives. The minimum value 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 an 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 **20**.

Maxsessionid

The maximum number of active session identifiers stored. A session is considered inactive when no information is received from that session for 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**.

NOTE

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**.

A.2. MOD_PROXY_CLUSTER MODULE AND DIRECTIVES

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 the 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
```

Configurable directives for mod_proxy_cluster

You can also configure the following directives in the **<VirtualHost>** element to change the load balancing behavior.

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 such as **http://local.com/**, the following **ProxyPass** directive converts 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)$ !
```

A.3. MOD_ADVERTISE MODULE AND DIRECTIVES

The Proxy Advertisement Module, **mod_advertise**, 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.

The **mod_advertise** module must be defined along with the **mod_manager** module in the **VirtualHost** element. In the following example, the identifier for the **mod_advertise** module is **advertise_module**:

```
LoadModule advertise_module modules/mod_advertise.so
```

Configurable directives for mod_advertise

The **mod_advertise** module is configurable by using the following directives:

ServerAdvertise

Defines how the advertising mechanism is used.
The default value is **Off**. When set to **Off**, the proxy does not advertise its location.

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 host name 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.

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 that is used to identify **mod_cluster** in Apache HTTP 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**.

A.4. MOD_PROXY MODULE AND DIRECTIVES

The **mod_proxy** module is a standard Apache HTTP Server module that enables the server to act as a proxy for data transferred over the AJP (Apache JServe Protocol), FTP, CONNECT (for SSL), and HTTP protocols. The **mod_proxy** module does not require additional configuration. The identifier for the **mod_proxy** module is **proxy_module**.

The configurable directives for **mod_proxy**, such as **ProxyIOBufferSize**, are used to configure the **mod_cluster** module.

A.5. MOD_PROXY_AJP MODULE AND DIRECTIVES

The **mod_proxy_ajp** module is a standard Apache HTTP Server module that provides support for AJP (Apache JServe Protocol) proxying. The **mod_proxy** module is also required if you want to use **mod_proxy_ajp**. Additionally, the **secret** property is required when using the Tomcat AJP Connector. The **secret** property can be added to the **ProxyPass** settings using the following command:

```
ProxyPass /example/ ajp://localhost:8009/example/ secret=YOUR_AJP_SECRET
```

NOTE

If you set a **secret** on a load balancer, all of its members inherit this **secret**.

A.6. MOD_CLUSTER_SLOTMEM MODULE AND DIRECTIVES

The **mod_cluster_slotmem** module is a shared memory provider for creating and accessing a shared memory segment in which the data sets are organized in "slots".

The **mod_cluster_slotmem** module does not require any configuration directives.

APPENDIX B. WORKERS.PROPERTIES FILE

Apache HTTP Server worker nodes are servlet containers that are mapped to the **mod_jk** load balancer. The worker nodes are defined in ***JBCS_HOME*/httpd/conf/workers.properties**. This file specifies where the different servlet containers are located, and how calls should be load-balanced across them.

B.1. WORKERS.PROPERTIES OVERVIEW

The **workers.properties** file contains a global properties section and a worker properties section.

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.

B.2. WORKERS.PROPERTIES DIRECTIVES

The **workers.properties** file directives are divided into global, mandatory, connection, and load-balancing classifications.

Global directives for workers.properties

worker.list

Specifies the list of worker names that **mod_jk** uses. 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***.

Mandatory directives for workers.properties

type

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

Other values include **lb** and **status**.

For detailed information about AJPv13, see the Apache Tomcat Connector - AJP Protocol Reference.

Connection directives for **workers.properties**

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 host name 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 the defined protocol requests. The default value is **8009**, which is the default listen port for AJPv13 workers.

ping_mode

Specifies the conditions under which connections are probed for their current network health. The probe uses an empty AJPv13 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 that 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 that 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 that all directive flags are applied. For information about the ***_timeout** advanced directives, see the 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).

Load balancing directives for **workers.properties**

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 can go to different servers based on server load.

If you specify **worker.loadbalancer.sticky_session=1**, each session is persisted (locked) to one server until the session is terminated, providing that server is available.

Additional resources

- Apache Tomcat Connectors - Reference Guide .

APPENDIX C. MULTI-PROCESSING MODULES (MPMS)

Red Hat JBoss Core Services includes a variety of multi-processing modules (MPMs). You can use these MPMs to customize how the Apache HTTP Server responds to incoming requests.

C.1. MPMS OVERVIEW

Multi-processing modules (MPMs) are available for both Red Hat Enterprise Linux and Microsoft Windows.

MPMs for Red Hat Enterprise Linux

prefork

The **prefork** MPM implements a non-threaded, pre-forking web server. A single control process is responsible for launching child processes, which listen for incoming connections and service them when they arrive. Each request is handled by a single process, ensuring that each request is isolated, and will not affect any other requests.

worker

The **worker** MPM implements a hybrid multi-process, multi-threaded server. Each child process creates a fixed number of server threads, allowing the server to handle a large number of requests with fewer system resources.

event

The **event** MPM is based off the **worker** MPM, but allows additional requests to be served simultaneously by passing off some processing work to the listener threads, therefore freeing up the worker threads to serve new requests.

MPMs for Microsoft Windows

winnt

The **winnt** MPM is the only one available for Windows systems. It uses a single control process, which launches a single process used to create threads for incoming requests.

C.2. SWITCHING THE MPM

The server selects the MPM based on the **LoadModule** directives in the ***JBCS_HOME*/httpd/conf.modules.d/00-mpm.conf** file. You can select a specific MPM by removing the comment character (**#**) from the **LoadModule** directive for that MPM in the **00-mpm.conf** file.

By default, the **prefork** MPM is selected. For example:

```
# prefork MPM: Implements a non-threaded, pre-forking web server
# See: http://httpd.apache.org/docs/2.4/mod/prefork.html
LoadModule mpm_prefork_module modules/mod_mpm_prefork.so
```

Procedure

1. Edit the ***JBCS_HOME*/httpd/conf.modules.d/00-mpm.conf** to add a comment (**#**) character to the **LoadModule** directive for the **prefork** MPM. For example:

   ```
   # prefork MPM: Implements a non-threaded, pre-forking web server
   # See: http://httpd.apache.org/docs/2.4/mod/prefork.html
   #LoadModule mpm_prefork_module modules/mod_mpm_prefork.so
   ```

2. In the same **00-mpm.conf** file, remove the comment (**#**) character from the **LoadModule** directive for the MPM that you want to switch to. These lines are located immediately below the **prefork** MPM.
   For example, to load the **worker** MPM, remove the comment (**#**) character from the **LoadModule** directive for the **worker** MPM:

   ```
   # worker MPM: Multi-Processing Module implementing a hybrid
   # multi-threaded multi-process web server
   # See: http://httpd.apache.org/docs/2.4/mod/worker.html
   LoadModule mpm_worker_module modules/mod_mpm_worker.so
   ```

3. To verify the MPM is configured correctly, enter the following command:

   ```
   $ sbin/apachectl -V
   ```

   This command displays the current MPM. For example:

   ```
   Server MPM:     worker
   ```

APPENDIX D. WORKER NODE 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 D.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 Apache HTTP Server 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 Apache HTTP Server 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 Apache HTTP Server 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 Apache HTTP Server proxy will attempt to send a given request to a worker before aborting. The minimum value is **1**: try once before aborting.
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.

Value	Default	Description
smax		Specifies the soft maximum idle connection count. The maximum value is determined by the Apache HTTP Server 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.

APPENDIX E. 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 E.1. Proxy discovery configuration attributes for**mod_cluster**

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 (120 seconds by default in the JBCS httpd.conf). **nodeTimeout**, **ProxyTimeout**, and **Timeout** are set at the socket level.

Table E.2. Proxy configuration attributes for**mod_cluster**

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

Attribute	Property	Default Value
flush-wait	**flushWait**	-1
ping	**ping**	10 (seconds)
smax	**smax**	-1 (uses the default value)
ttl	**ttl**	-1 (uses the default value)
domain	**loadBalancingGroup**	
load-balancing-group	**loadBalancingGroup**	

APPENDIX F. LOAD CONFIGURATION FOR TOMCAT

You can configure the following additional properties for load metrics when you want to use **mod_cluster** with Apache Tomcat.

Table F.1. Load Configuration for Tomcat

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