



Red Hat JBoss Enterprise Application Platform 6.4

Administration and Configuration Guide

For Use with Red Hat JBoss Enterprise Application Platform 6

Red Hat JBoss Enterprise Application Platform 6.4 Administration and Configuration Guide

For Use with Red Hat JBoss Enterprise Application Platform 6

Legal Notice

Copyright © 2017 Red Hat, Inc..

This document is licensed by Red Hat under the [Creative Commons Attribution-ShareAlike 3.0 Unported License](https://creativecommons.org/licenses/by-sa/3.0/). If you distribute this document, or a modified version of it, you must provide attribution to Red Hat, Inc. and provide a link to the original. If the document is modified, all Red Hat trademarks must be removed.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux[®] is the registered trademark of Linus Torvalds in the United States and other countries.

Java[®] is a registered trademark of Oracle and/or its affiliates.

XFS[®] is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL[®] is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js[®] is an official trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack[®] Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

Abstract

This book is a guide to the administration and configuration of Red Hat JBoss Enterprise Application Platform 6 and its patch releases.

Table of Contents

CHAPTER 1. INTRODUCTION	6
1.1. ABOUT RED HAT JBOSS ENTERPRISE APPLICATION PLATFORM 6	6
1.2. FEATURES OF JBOSS EAP 6	6
1.3. ABOUT JBOSS EAP 6 OPERATING MODES	7
1.4. ABOUT STANDALONE SERVERS	7
1.5. ABOUT MANAGED DOMAINS	7
1.6. ABOUT THE DOMAIN CONTROLLER	9
1.7. ABOUT DOMAIN CONTROLLER DISCOVERY AND FAILOVER	9
1.8. ABOUT HOST CONTROLLER	10
1.9. ABOUT SERVER GROUPS	11
1.10. ABOUT JBOSS EAP 6 PROFILES	12
1.11. MANAGE SERVERS OF DIFFERENT VERSIONS	12
CHAPTER 2. APPLICATION SERVER MANAGEMENT	14
2.1. JBOSS EAP DOCUMENTATION CONVENTIONS	14
2.2. START AND STOP JBOSS EAP 6	14
2.3. START AND STOP SERVERS	26
2.4. CONFIGURATION FILES	29
2.5. FILESYSTEM PATHS	39
CHAPTER 3. MANAGEMENT INTERFACES	45
3.1. MANAGE THE APPLICATION SERVER	45
3.2. MANAGEMENT APPLICATION PROGRAMMING INTERFACES (APIS)	45
3.3. THE MANAGEMENT CONSOLE	47
3.4. THE MANAGEMENT CLI	62
3.5. MANAGEMENT CLI OPERATIONS	77
3.6. THE MANAGEMENT CLI COMMAND HISTORY	96
3.7. MANAGEMENT INTERFACE AUDIT LOGGING	98
CHAPTER 4. USER MANAGEMENT	102
4.1. ABOUT JBOSS EAP USER MANAGEMENT	102
4.2. USER CREATION	102
4.3. ADD-USER SCRIPT COMMAND LINE EXAMPLES	106
CHAPTER 5. NETWORK AND PORT CONFIGURATION	109
5.1. INTERFACES	109
5.2. SOCKET BINDING GROUPS	114
5.3. IPV6	123
5.4. REMOTING	125
CHAPTER 6. DATASOURCE MANAGEMENT	134
6.1. INTRODUCTION	134
6.2. JDBC DRIVERS	136
6.3. NON-XA DATASOURCES	141
6.4. XA DATASOURCES	145
6.5. DATASOURCE SECURITY	154
6.6. DATABASE CONNECTION VALIDATION	155
6.7. DATASOURCE CONFIGURATION	157
6.8. EXAMPLE DATASOURCES	168
CHAPTER 7. CONFIGURING MODULES	182
7.1. INTRODUCTION	182
7.2. DISABLE SUBDEPLOYMENT MODULE ISOLATION FOR ALL DEPLOYMENTS	184

7.3. ADD A MODULE TO ALL DEPLOYMENTS	185
7.4. CREATE A CUSTOM MODULE	186
7.5. DEFINE AN EXTERNAL JBOSS MODULE DIRECTORY	188
7.6. REFERENCE	189
CHAPTER 8. JSVC	190
8.1. INTRODUCTION	190
CHAPTER 9. GLOBAL VALVES	195
9.1. ABOUT VALVES	195
9.2. ABOUT GLOBAL VALVES	195
9.3. ABOUT AUTHENTICATOR VALVES	195
9.4. INSTALL A GLOBAL VALVE	195
9.5. CONFIGURE A GLOBAL VALVE	196
CHAPTER 10. APPLICATION DEPLOYMENT	198
10.1. ABOUT APPLICATION DEPLOYMENT	198
10.2. DEPLOY WITH THE MANAGEMENT CONSOLE	199
10.3. DEPLOY WITH THE MANAGEMENT CLI	205
10.4. DEPLOY WITH THE HTTP API	208
10.5. DEPLOY WITH THE DEPLOYMENT SCANNER	209
10.6. DEPLOY WITH MAVEN	218
10.7. CONTROL THE ORDER OF DEPLOYED APPLICATIONS ON JBOSS EAP 6	221
10.8. DEFINE A CUSTOM DIRECTORY FOR DEPLOYED CONTENT	222
10.9. DEPLOYMENT DESCRIPTOR OVERRIDES	223
10.10. ROLLOUT PLAN	224
CHAPTER 11. SUBSYSTEM CONFIGURATION	228
11.1. SUBSYSTEM CONFIGURATION OVERVIEW	228
CHAPTER 12. THE LOGGING SUBSYSTEM	229
12.1. INTRODUCTION	229
12.2. CONFIGURE LOGGING IN THE MANAGEMENT CONSOLE	240
12.3. LOGGING CONFIGURATION IN THE CLI	241
12.4. PER-DEPLOYMENT LOGGING	275
12.5. LOGGING PROFILES	276
12.6. LOGGING CONFIGURATION PROPERTIES	280
12.7. SAMPLE XML CONFIGURATION FOR LOGGING	287
CHAPTER 13. INFINISPAN	290
13.1. ABOUT INFINISPAN	290
13.2. CLUSTERING MODES	290
13.3. CACHE CONTAINERS	291
13.4. ABOUT INFINISPAN STATISTICS	293
13.5. ENABLE INFINISPAN STATISTICS COLLECTION	293
13.6. SWITCHING TO DISTRIBUTED CACHE MODE FOR WEB SESSION REPLICATION	295
13.7. JGROUPS	295
13.8. JGROUPS TROUBLESHOOTING	296
CHAPTER 14. JVM	298
14.1. ABOUT JVM	298
CHAPTER 15. WEB SUBSYSTEM	309
15.1. CONFIGURE THE WEB SUBSYSTEM	309
15.2. CONFIGURE THE HTTP SESSION TIMEOUT	309

15.3. SERVLET/HTTP CONFIGURATION	310
15.4. REPLACE THE DEFAULT WELCOME WEB APPLICATION	319
15.5. SYSTEM PROPERTIES IN JBOSSWEB	319
15.6. ABOUT HTTP-ONLY SESSION MANAGEMENT COOKIES	325
CHAPTER 16. WEB SERVICES SUBSYSTEM	327
16.1. CONFIGURE WEB SERVICES OPTIONS	327
16.2. OVERVIEW OF HANDLERS AND HANDLER CHAINS	328
CHAPTER 17. HTTP CLUSTERING AND LOAD BALANCING	330
17.1. HTTP SERVER NAME CONVENTIONS	330
17.2. INTRODUCTION	331
17.3. CONNECTOR CONFIGURATION	335
17.4. WEB SERVER CONFIGURATION	338
17.5. CLUSTERING	350
17.6. WEB, HTTP CONNECTORS, AND HTTP CLUSTERING	356
17.7. APACHE MOD_JK	380
17.8. APACHE MOD_PROXY	391
17.9. MICROSOFT ISAPI CONNECTOR	394
17.10. ORACLE NSAPI CONNECTOR	401
CHAPTER 18. MESSAGING	407
18.1. INTRODUCTION	407
18.2. CONFIGURATION OF TRANSPORTS	409
18.3. DEAD CONNECTION DETECTION	417
18.4. WORK WITH LARGE MESSAGES	419
18.5. PAGING	421
18.6. DIVERTS	424
18.7. THE CLIENT CLASSPATH	425
18.8. CONFIGURATION	426
18.9. PRE_ACKNOWLEDGE MODE	453
18.10. THREAD MANAGEMENT	455
18.11. MESSAGE GROUPING	457
18.12. DUPLICATE MESSAGE DETECTION	460
18.13. JMS BRIDGES	462
18.14. PERSISTENCE	465
18.15. HORNETQ CLUSTERING	466
18.16. HIGH AVAILABILITY	476
18.17. PERFORMANCE TUNING	484
CHAPTER 19. TRANSACTION SUBSYSTEM	489
19.1. TRANSACTION SUBSYSTEM CONFIGURATION	489
19.2. TRANSACTION ADMINISTRATION	496
19.3. TRANSACTION REFERENCES	501
19.4. ORB CONFIGURATION	502
19.5. JDBC OBJECT STORE SUPPORT	505
CHAPTER 20. MAIL SUBSYSTEM	507
20.1. USE CUSTOM TRANSPORTS IN MAIL SUBSYSTEM	507
CHAPTER 21. ENTERPRISE JAVABEANS	509
21.1. INTRODUCTION	509
21.2. CONFIGURING BEAN POOLS	510
21.3. CONFIGURING EJB THREAD POOLS	515
21.4. CONFIGURING SESSION BEANS	519

21.5. CONFIGURING MESSAGE-DRIVEN BEANS	524
21.6. CONFIGURING THE EJB3 TIMER SERVICE	525
21.7. CONFIGURING THE EJB ASYNCHRONOUS INVOCATION SERVICE	528
21.8. CONFIGURING THE EJB3 REMOTE INVOCATION SERVICE	529
21.9. CONFIGURING EJB 2.X ENTITY BEANS	529
CHAPTER 22. JAVA CONNECTOR ARCHITECTURE (JCA)	533
22.1. INTRODUCTION	533
22.2. CONFIGURE THE JAVA CONNECTOR ARCHITECTURE (JCA) SUBSYSTEM	534
22.3. DEPLOY A RESOURCE ADAPTER	540
22.4. CONFIGURE A DEPLOYED RESOURCE ADAPTER	541
22.5. RESOURCE ADAPTER DESCRIPTOR REFERENCE	546
22.6. VIEW DEFINED CONNECTION STATISTICS	551
22.7. RESOURCE ADAPTER STATISTICS	551
22.8. DEPLOY THE WEBSHERE MQ RESOURCE ADAPTER	552
22.9. INSTALL JBOSS ACTIVE MQ RESOURCE ADAPTER	557
22.10. CONFIGURE A GENERIC JMS RESOURCE ADAPTER FOR USE WITH A THIRD-PARTY JMS PROVIDER	558
22.11. CONFIGURING THE HORNETQ JCA ADAPTER FOR REMOTE CONNECTIONS	562
CHAPTER 23. HIBERNATE SEARCH	566
23.1. GETTING STARTED WITH HIBERNATE SEARCH	566
23.2. CONFIGURATION	570
23.3. MONITORING	594
CHAPTER 24. DEPLOY JBOSS EAP 6 ON AMAZON EC2	595
24.1. INTRODUCTION	595
24.2. DEPLOYING JBOSS EAP 6 ON AMAZON EC2	597
24.3. NON-CLUSTERED JBOSS EAP 6	597
24.4. NON-CLUSTERED INSTANCES	597
24.5. NON-CLUSTERED MANAGED DOMAINS	601
24.6. CLUSTERED JBOSS EAP 6	607
24.7. CLUSTERED INSTANCES	613
24.8. CLUSTERED MANAGED DOMAINS	617
24.9. ESTABLISHING MONITORING WITH JBOSS OPERATIONS NETWORK (JON)	622
24.10. USER SCRIPT CONFIGURATION	625
24.11. TROUBLESHOOTING	629
CHAPTER 25. EXTERNALIZE SESSIONS	631
25.1. EXTERNALIZE HTTP SESSION FROM JBOSS EAP TO JBOSS DATA GRID	631
APPENDIX A. SUPPLEMENTAL REFERENCES	633
A.1. DOWNLOAD FILES FROM THE RED HAT CUSTOMER PORTAL	633
A.2. CONFIGURE THE DEFAULT JAVA DEVELOPMENT KIT ON RED HAT ENTERPRISE LINUX	633
A.3. MANAGEMENT INTERFACE AUDIT LOGGING REFERENCE	634
APPENDIX B. REVISION HISTORY	638

CHAPTER 1. INTRODUCTION

1.1. ABOUT RED HAT JBOSS ENTERPRISE APPLICATION PLATFORM 6

Red Hat JBoss Enterprise Application Platform 6 (JBoss EAP 6) is a middleware platform built on open standards and compliant with the Java Enterprise Edition 6 specification. It integrates JBoss Application Server 7 with high-availability clustering, messaging, distributed caching, and other technologies.

JBoss EAP 6 includes a new, modular structure that allows service enabling only when required, improving startup speed.

The Management Console and Management Command Line Interface make editing XML configuration files unnecessary and add the ability to script and automate tasks.

In addition, JBoss EAP 6 includes APIs and development frameworks for quickly developing secure and scalable Java EE applications.

[Report a bug](#)

1.2. FEATURES OF JBOSS EAP 6

Table 1.1. JBoss EAP 6 Features

Feature	Description
Java Certification	Java Enterprise Edition 6 Full Profile and Web Profile certified.
Managed Domain	<ul style="list-style-type: none"> ● Centralized management of multiple server instances and physical hosts, while a standalone server allows for a single server instance. ● Per-server group management of configuration, deployment, socket bindings, modules, extensions and system properties. ● Centralized and simplified management of application security (including security domains).
Management Console and Management CLI	New domain or standalone server management interfaces. XML configuration file editing is no longer required. The Management CLI also includes a batch mode that can script and automate management tasks.
Simplified directory layout	The modules directory now contains all application server modules. The common and server-specific lib directories are deprecated. The domain and standalone directories contain the artifacts and configuration files for domain and standalone deployments respectively.

Feature	Description
Modular class loading mechanism	Modules are loaded and unloaded on demand. This improves performance, has security benefits and reduces start-up and restart times.
Streamlined Data source management	Database drivers are deployed like other services. In addition, datasources are created and managed directly in the Management Console or Management CLI.
Reduced and more efficient resource use	JBoss EAP 6 uses fewer system resources and uses them more efficiently than previous versions. Among other benefits, JBoss EAP 6 starts and stops faster than JBoss EAP 5.

[Report a bug](#)

1.3. ABOUT JBOSS EAP 6 OPERATING MODES

JBoss EAP 6 provides two operating modes for JBoss EAP 6 instances: standalone server or managed domain.

The two modes differ in how servers are managed, not in their capacity to service end-user requests. It is important to note that the high-availability (HA) cluster functionality is available via either operating mode. A group of standalone servers can be configured to form an HA cluster.

[Report a bug](#)

1.4. ABOUT STANDALONE SERVERS

Standalone server mode is an independent process and is analogous to the only running mode available in previous JBoss EAP versions.

A JBoss EAP 6 instance running as a standalone server is a single instance only but can optionally run in a clustered configuration.

[Report a bug](#)

1.5. ABOUT MANAGED DOMAINS

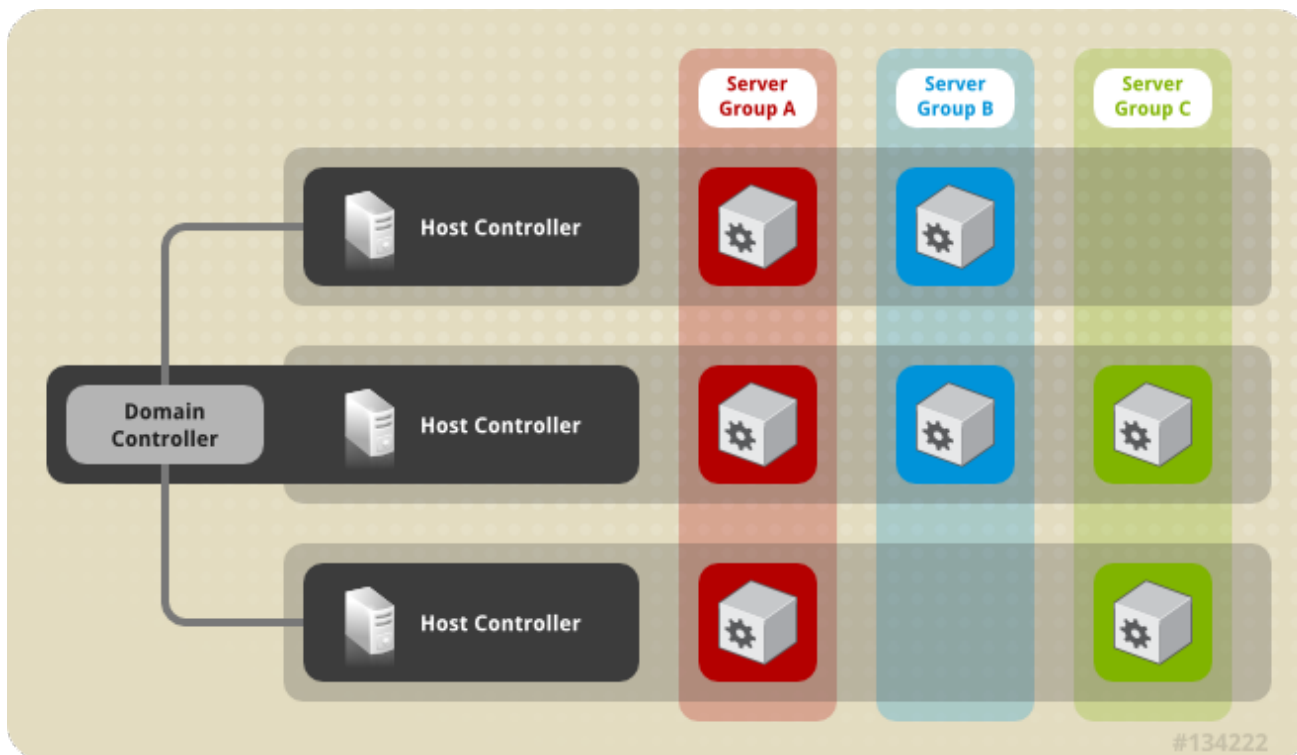


Figure 1.1. Graphical Representation of a Managed Domain

The managed domain operating mode allows for management of multiple JBoss EAP 6 instances from a single control point.

Centrally managed JBoss EAP 6 server collections are known as members of a domain. All JBoss EAP 6 instances in a domain share a common management policy.

A domain consists of one domain controller, one or more host controller(s), and zero or more server groups per host.

A domain controller is the central point from which the domain is controlled. It ensures that each server is configured according to the management policy of the domain. The domain controller is also a host controller.

A host controller is a physical or virtual host on which the **domain.sh** or **domain.bat** script is run. Host controllers are configured to delegate domain management tasks to the domain controller.

The host controller on each host interacts with the domain controller to control the lifecycle of the application server instances running on its host and to assist the domain controller to manage them. Each host can contain multiple server groups.

A server group is a set of server instances which have JBoss EAP 6 installed on them and are managed and configured as one. The domain controller manages the configuration of and applications deployed onto server groups. Consequently, each server in a server group shares the same configuration and deployments.

It is possible for a domain controller, a single host controller, and multiple servers to run within the same JBoss EAP 6 instance, on the same physical system.

Host controllers are tied to specific physical (or virtual) hosts. You can run multiple host controllers on the same hardware if you use different configurations, ensuring their ports and other resources do not conflict.

[Report a bug](#)

1.6. ABOUT THE DOMAIN CONTROLLER

A domain controller is the JBoss EAP 6 server instance that acts as a central management point for a domain. One host controller instance is configured to act as a domain controller.

The primary responsibilities of the domain controller are:

- Maintain the domain's central management policy.
- Ensure all host controllers are aware of its current contents.
- Assist the host controllers in ensuring that all running JBoss EAP 6 instances are configured in accordance with this policy.

By default, the central management policy is stored in the **domain/configuration/domain.xml** file. This file is in the unzipped JBoss EAP 6 installation file, on the domain controller's host's filesystem.

A **domain.xml** file must be located in the **domain/configuration/** directory of the host controller set to run as the domain controller. This file is not mandatory for installations on host controllers that are not meant to run as a domain controller. The presence of a **domain.xml** file on such a server does no harm, however.

The **domain.xml** file contains the profile configurations that can be run on the server instances in a domain. A profile configuration includes the detailed settings of the various subsystems that comprise a profile. The domain configuration also includes the definition of socket groups and the server group definitions.

[Report a bug](#)

1.7. ABOUT DOMAIN CONTROLLER DISCOVERY AND FAILOVER

When setting up a managed domain, each host controller must be configured with information needed to contact the domain controller. In JBoss EAP 6, each host controller can be configured with multiple options for finding the domain controller. Host controllers iterate through the list of options until one succeeds.

This allows host controllers to be pre-configured with contact information for a backup domain controller. A backup host controller can be promoted to master if there is a problem with the primary domain controller, allowing host controllers to automatically fail over to the new master once it's been promoted.

The following is an example of how to configure a host controller with multiple options for finding the domain controller.

Example 1.1. Host controller configured with multiple domain controller options

```
<domain-controller>
  <remote security-realm="ManagementRealm">
    <discovery-options>
      <static-discovery name="primary" host="172.16.81.100" port="9999"/>
      <static-discovery name="backup" host="172.16.81.101" port="9999"/>
    </discovery-options>
  </remote>
</domain-controller>
```

A static discovery option includes the following mandatory attributes:

name

The name for this domain controller discovery option

host

The remote domain controller's host name.

port

The remote domain controller's port.

In the example above, the first discovery option is the one expected to succeed. The second can be used in failover situations.

If a problem arises with the primary domain controller, a host controller that was started with the **--backup** option can be promoted to act as the domain controller.

**NOTE**

Starting a host controller with the **--backup** option will cause that controller to maintain a local copy of the domain configuration. This configuration will be used if the host controller is reconfigured to act as the domain controller.

Procedure 1.1. Promoting a host controller to be the domain controller

1. Ensure the original domain controller has, or is, stopped.
2. Use the Management CLI to connect to the host controller that is to become the new domain controller.
3. Execute the following command to configure the host controller to act as the new domain controller.

```
/host=HOST_NAME:write-local-domain-controller
```

4. Execute the following command to reload the host controller.

```
reload --host=HOST_NAME
```

The host controller chosen in step 2 will now act as the domain controller.

[Report a bug](#)

1.8. ABOUT HOST CONTROLLER

A host controller is launched when the **domain.sh** or **domain.bat** script is run on a host.

The primary responsibility of a host controller is server management. It delegates domain management tasks and is responsible for starting and stopping the individual application server processes that run on its host.

It interacts with the domain controller to help manage the communication between the servers and the domain controller. Multiple host controllers of a domain can interact with only a single domain controller. Hence, all the host controllers and server instances running on a single domain mode have a single domain controller and must belong to the same domain.

By default each host controller reads its configuration from the **domain/configuration/host.xml** file located in the unzipped JBoss EAP 6 installation file on its host's filesystem. The **host.xml** file contains the following configuration information that is specific to the particular host:

- The names of the JBoss EAP 6 instances meant to run from this installation.
- Any of the following configurations:
 - How the host controller contacts the domain controller to register itself and access the domain configuration.
 - How to find and contact a remote domain controller.
 - That the host controller is to act as the domain controller
- Configurations specific to the local physical installation. For example, named interface definitions declared in **domain.xml** can be mapped to an actual machine-specific IP address in **host.xml**. And abstract path names in **domain.xml** can be mapped to actual filesystem paths in **host.xml**.

[Report a bug](#)

1.9. ABOUT SERVER GROUPS

A server group is a collection of server instances that are managed and configured as one. In a managed domain, every application server instance belongs to a server group, even if it is the only member. The server instances in a group share the same profile configuration and deployed content.

A domain controller and a host controller enforce the standard configuration on all server instances of every server group in its domain.

A domain can consist of multiple server groups. Different server groups can be configured with different profiles and deployments. A domain can be configured with different server tiers providing different services, for example.

Different server groups can also have the same profile and deployments. This can, for example, allow for rolling application upgrades where the application is upgraded on one server group and then updated on a second server group, avoiding a complete service outage.

The following is an example of a server group definition:

Example 1.2. Server group definition

```
<server-group name="main-server-group" profile="default">
  <socket-binding-group ref="standard-sockets"/>
  <deployments>
    <deployment name="foo.war_v1" runtime-name="foo.war"/>
    <deployment name="bar.ear" runtime-name="bar.ear"/>
  </deployments>
</server-group>
```

-

A server group includes the following mandatory attributes:

- name: the server group name.
- profile: the server group profile name.
- socket-binding-group: the default socket binding group used for servers in the group. This name can be overridden on a per-server basis in **host.xml**. However, this is a mandatory element for every server group and the domain can not start if it is missing.

A server group includes the following optional attributes:

- deployments: the deployment content to be deployed on the servers in the group.
- system-properties: the system properties to be set on servers in the group
- jvm: the default JVM settings for all servers in the group. The host controller merges these settings with any other configuration provided in **host.xml** to derive the settings used to launch the server's JVM.
- socket-binding-port-offset: the default offset to be added to the port values given by the socket binding group.
- management-subsystem-endpoint: set to **true** to have servers belonging to the server group connect back to the host controller using the endpoint from their Remoting subsystem (the Remoting subsystem must be present for this to work).

[Report a bug](#)

1.10. ABOUT JBOSS EAP 6 PROFILES

The concept of profiles that was used in previous versions of JBoss EAP is no longer used. JBoss EAP 6 now uses a small number of configuration files to hold all information about its configuration.

Modules and drivers are now loaded on an as-needed basis. Consequently the concept of a default profile - used in previous versions of JBoss EAP 6 to make the server start more efficiently - does not apply.

At deployment time, module dependencies are determined, ordered, resolved by the server or domain controller, and loaded in the correct order. Modules are unloaded when no deployment needs them any longer.

It is possible to disable modules or unload drivers and other services manually by removing the subsystems from the configuration. However, for most cases this is unnecessary. If none of your applications use a module, it will not be loaded.

[Report a bug](#)

1.11. MANAGE SERVERS OF DIFFERENT VERSIONS



NOTE

You must have the latest release of JBoss EAP is functioning as the domain controller in order to manage different versions of JBoss EAP servers.

- JBoss EAP schema uses different versions. Hence, JBoss EAP domain controller of a higher version must not have issues controlling a JBoss EAP host of a lower version, but the **domain.xml** must be the **oldest** of all the versions in use.
- If there is a cluster, all member servers of the cluster must belong to the same version of JBoss EAP.
- On every host in the domain, there are several Java processes like Process Controller, Host Controller and managed servers. These Java processes must be launched from the same installation of JBoss EAP, hence have the same version.



WARNING

However, there is a minor incompatibility when the domain controller from JBoss EAP 6.3 manages slaves from JBoss EAP 6.2 or below that must be corrected: the **[named-formatter]** attribute is not understood in the target model version and must be replaced with older attributes. For more details, refer to <https://access.redhat.com/solutions/1238073>

[Report a bug](#)

CHAPTER 2. APPLICATION SERVER MANAGEMENT

2.1. JBOSS EAP DOCUMENTATION CONVENTIONS

All instances of *EAP_HOME* in this guide refer to the JBoss EAP root installation directory, which depends on the installation method you used.

Zip Installation Method

EAP_HOME refers to the directory in which the JBoss EAP ZIP file was extracted.

Installer Method

EAP_HOME refers to the directory in which you chose to install JBoss EAP.

RPM Installation Method

EAP_HOME refers to the directory `/usr/share/jbossas`.



NOTE

The notation *EWS_HOME* is used to refer to JBoss EWS installation locations following the same conventions outlined above for JBoss EAP.

[Report a bug](#)

2.2. START AND STOP JBOSS EAP 6

2.2.1. Start JBoss EAP 6

JBoss EAP runs in one of two modes, Standalone Server or Managed Domain, and is supported on two platforms, Red Hat Enterprise Linux and Microsoft Windows Server. The specific command to start JBoss EAP depends on the underlying platform and the desired mode.

Table 2.1. Commands to start JBoss EAP

Operating System	Standalone Server	Managed Domain
Red Hat Enterprise Linux	<i>EAP_HOME</i>/bin/standalone.sh	<i>EAP_HOME</i>/bin/domain.sh
Microsoft Windows Server	<i>EAP_HOME</i>bin\standalone.bat	<i>EAP_HOME</i>bin\domain.bat

For more information on how to start JBoss EAP 6, refer [Section 2.2.2, “Start JBoss EAP 6 as a Standalone Server”](#) and [Section 2.2.4, “Start JBoss EAP 6 as a Managed Domain”](#).

[Report a bug](#)

2.2.2. Start JBoss EAP 6 as a Standalone Server

Summary

This topic covers the steps to start JBoss EAP 6 as a Standalone Server.

Procedure 2.1. Start the Platform Service as a Standalone Server

1. **For Red Hat Enterprise Linux.**
Run the command: ***EAP_HOME/bin/standalone.sh***
2. **For Microsoft Windows Server.**
Run the command: ***EAP_HOME\bin\standalone.bat***
3. **Optional: Specify additional parameters.**
To list all available parameters for the start-up scripts, use the ***-h*** parameter.

Result

The JBoss EAP 6 Standalone Server instance starts.

[Report a bug](#)

2.2.3. Running Multiple JBoss EAP Standalone Servers on a Single Machine

Summary

This topic describes the steps for running multiple JBoss EAP Standalone servers on a single machine.

Procedure 2.2. Run multiple instances of JBoss EAP standalone servers on a single machine

1. Create a copy of the ***EAP_HOME/standalone/*** directory directly under *EAP_HOME/* for each standalone server. For example, to create a directory for standalone servers **node1** and **node2**, type the following commands.

```
$ cd EAP_HOME
$ cp -a ./standalone ./node1
$ cp -a ./standalone ./node2
```

2. Start each JBoss EAP standalone instance by specifying the node name, IP address, server directory, optional server configuration file, and optional port offset. The command uses the following syntax:

```
$. /bin/standalone.sh -Djboss.node.name=UNIQUE_NODENAME -
Djboss.server.base.dir=EAP_HOME/NODE_DIRECTORY -b IP_ADDRESS -bmanagement
MGMT_IP_ADDRESS --server-config=SERVER_CONFIGURATION_FILE -
Djboss.socket.binding.port-offset=PORT_OFFSET
```

- a. This example starts **node1**

```
$ cd EAP_HOME
$. /bin/standalone.sh -Djboss.node.name=node1 -
Djboss.server.base.dir=EAP_HOME/node1 -b 10.10.10.10 -bmanagement 127.0.0.1
```

- b. This example to start **node2** depends on whether the machine supports multiple IP addresses.

- If the machine supports multiple IP addresses, the following command is to be used.

```
$ cd EAP_HOME
$ ./bin/standalone.sh -Djboss.node.name=node2 -
Djboss.server.base.dir=EAP_HOME/node2 -b 10.10.10.40 -bmanagement
127.0.0.40
```

- If the machine does not support multiple IP addresses, you must specify a **jboss.socket.binding.port-offset** property to avoid a port conflict.

```
$ cd EAP_HOME
$ ./bin/standalone.sh -Djboss.node.name=node2 -
Djboss.server.base.dir=EAP_HOME/node2 -b 10.10.10.10 -bmanagement 127.0.0.1
-Djboss.socket.binding.port-offset=100
```



NOTE

If you would like to manage two nodes at once or two nodes that have the same configuration, you are recommended to run them in a managed domain instead of running a standalone server.

[Report a bug](#)

2.2.4. Start JBoss EAP 6 as a Managed Domain

Order of Operations

The domain controller must be started before any slave servers in any server groups in the domain. Use this procedure first on the domain controller, and then on each associated host controller and each other host associated with the domain.

Procedure 2.3. Start the Platform Service as a Managed Domain

1. **For Red Hat Enterprise Linux.**
Run the command: ***EAP_HOME/bin/domain.sh***
2. **For Microsoft Windows Server.**
Run the command: ***EAP_HOME\bin\domain.bat***
3. **Optional: Pass additional parameters to the start-up script.**
To list all available parameters for the start-up scripts, use the **-h** parameter.

Result

The JBoss EAP 6 Managed Domain instance starts.

[Report a bug](#)

2.2.5. Configure the Name of a Host in a Managed Domain

Summary

Every host running in a managed domain must have a unique host name. To ease administration and allow for the use of the same host configuration files on multiple hosts, the server uses the following precedence for determining the host name.

1. If set, the **host** element **name** attribute in the **host.xml** configuration file.
2. The value of the **jboss.host.name** system property.
3. The value that follows the final period (".") character in the **jboss.qualified.host.name** system property, or the entire value if there is no final period (".") character.
4. The value that follows the period (".") character in the **HOSTNAME** environment variable for POSIX-based operating systems, the **COMPUTERNAME** environment variable for Microsoft Windows, or the entire value if there is no final period (".") character.

For information about how to set environment variables, see the documentation for your operating system. For information about how to set system properties, see [Section 3.5.11, "Configure System Properties Using the Management CLI"](#).

This topic describes how set the name of the host in the configuration file, using either a system property or a hard-coded name.

Procedure 2.4. Configure the Host Name Using a System Property

1. Open the host configuration file for editing, for example, **host.xml**.
2. Find the **host** element in the file, for example:

```
<host name="master" xmlns="urn:jboss:domain:1.6">
```

3. If it is present, remove the **name="HOST_NAME"** attribute declaration. The **host** element should now look like the following example.

```
<host xmlns="urn:jboss:domain:1.6">
```

4. Start the server passing the **-Djboss.host.name** argument, for example:

```
-Djboss.host.name=HOST_NAME
```

Procedure 2.5. Configure the Host Name Using a Specific Name

1. Start the JBoss EAP slave host using the following syntax:

```
bin/domain.sh --host-config=HOST_FILE_NAME
```

For example:

```
bin/domain.sh --host-config=host-slave01.xml
```

2. Launch the Management CLI.
3. Use the following syntax to replace the host name:

```
/host=EXISTING_HOST_NAME:write-attribute(name="name",value=UNIQUE_HOST_NAME)
```

For example:

```
/host=master:write-attribute(name="name",value="host-slave01")
```

You should see the following result.

```
"outcome" => "success"
```

This modifies the host **name** attribute in the **host-slave01.xml** file as follows:

```
<host name="host-slave01" xmlns="urn:jboss:domain:1.6">
```

4. You must reload the server configuration using the old host name to complete the process

```
reload --host=EXISTING_HOST_NAME
```

For example:

```
reload --host=master
```

[Report a bug](#)

2.2.6. Create Managed Domain on Two Machines



NOTE

You may need to configure your firewall to run this example.

You can create managed domain on two machines, wherein one machine is a domain controller and the other machine is a host. For more information, see [Section 1.6, "About the Domain Controller"](#).

- IP1 = IP address of the domain controller (Machine 1)
- IP2 = IP address of the host (Machine 2)

Procedure 2.6. Create managed domain on two machines

1. On Machine 1

- a. Use the `add-user.sh` script to add management user. For example, **slave01**, so the host can authenticate the domain controller. Note the **SECRET_VALUE** from the **add-user** output.
- b. Start domain with **host-master.xml** config file, which is preconfigured for dedicated domain controller.
- c. Use **-bmanagement=\$IP1** to make domain controller visible to other machines.

```
EAP_HOME/bin/domain.sh --host-config=host-master.xml -bmanagement=$IP1
```

2. On Machine 2

- a. Update ***EAP_HOME/domain/configuration/host-slave.xml*** file with user credentials.

```
<?xml version='1.0' encoding='UTF-8'?>
  <host xmlns="urn:jboss:domain:1.6" name="slave01">
    <!-- add user name here -->
    <management>
      <security-realms>
        <security-realm name="ManagementRealm">
          <server-identities>
            <secret value="$SECRET_VALUE" />
            <!-- use secret value from add-user.sh output-->
          </server-identities>
          ...
```

- b. Start host.

```
EAP_HOME/bin/domain.sh --host-config=host-slave.xml -
Djboss.domain.master.address=$IP1 -b=$IP2
```

3. Now we can manage the domain.

via CLI:

```
EAP_HOME/bin/jboss-cli.sh -c --controller=$IP1
```

via Web Console:

```
http://$IP1:9990
```

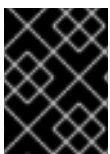
Access the server index page:

```
http://$IP2:8080/
http://$IP2:8230/
```

[Report a bug](#)

2.2.7. Create Managed Domain on a Single Machine

Multiple host controllers can be run on a single machine by using the **`jboss.domain.base.dir`** property.



IMPORTANT

It is not supported to configure more than one JBoss EAP host controller as a system service on a single machine.

Procedure 2.7. Run Multiple Host Controllers on a Single Machine

1. Copy the ***EAP_HOME/domain*** directory for the domain controller.

```
cp -r EAP_HOME/domain /path/to/domain1
```

- Copy the **EAP_HOME/domain** directory for a host controller.

```
cp -r EAP_HOME/domain /path/to/host1
```

- Start the domain controller using **/path/to/domain1**.

```
EAP_HOME/bin/domain.sh --host-config=host-master.xml -  
Djboss.domain.base.dir=/path/to/domain1
```

- Start the host controller using **/path/to/host1**.

```
EAP_HOME/bin/domain.sh --host-config=host-slave.xml -  
Djboss.domain.base.dir=/path/to/host1 -Djboss.domain.master.address=IP_ADDRESS -  
Djboss.management.native.port=PORT
```

Result

Each instance started in this manner will share the rest of the resources in the base installation directory (i.e. **EAP_HOME/modules/**), but use the domain configuration from the directory specified by **jboss.domain.base.dir**.

[Report a bug](#)

2.2.8. Start JBoss EAP 6 with an Alternative Configuration

If you do not specify a configuration file, the server starts with the default file.

You can also specify a configuration manually. This process varies depending on whether you are using a Managed Domain or Standalone Server, and operating system.

Prerequisites

- Before using an alternative configuration file, prepare it using the default configuration as a template.
- For Managed Domains, alternative configuration files are stored in the **EAP_HOME/domain/configuration/** directory.
- For Standalone Servers, alternative configuration files are stored in the **EAP_HOME/standalone/configuration/** directory.



NOTE

Example configurations are included in the **EAP_HOME/docs/examples/configs/** directory. Use these examples to enable features such as clustering or the Transactions XTS API.

Start the Instance with an Alternative Configuration

Standalone server

For a Standalone Server, provide the configuration filename using the **--server-config** switch. The configuration file must be in the **EAP_HOME/standalone/configuration/** directory, and you must specify the file path relative to this directory.

■

Example 2.1. Using an Alternate Configuration file for a Standalone Server in Red Hat Enterprise Linux

```
[user@host bin]$ ./standalone.sh --server-config=standalone-alternate.xml
```

This example uses the ***EAP_HOME/standalone/configuration/standalone-alternate.xml*** configuration file.

Example 2.2. Using an Alternate Configuration file for a Standalone Server in Microsoft Windows Server

```
C:\EAP_HOME\bin> standalone.bat --server-config=standalone-alternate.xml
```

This example uses the ***EAP_HOME\standalone\configuration\standalone-alternate.xml*** configuration file.

Managed Domain

For a Managed Domain, provide the configuration filename using the ***--domain-config*** switch. The configuration file must be in the ***EAP_HOME/domain/configuration/*** directory, and you need to specify the path relative to that directory.

Example 2.3. Using an Alternate Configuration file for a Managed Domain in Red Hat Enterprise Linux

```
[user@host bin]$ ./domain.sh --domain-config=domain-alternate.xml
```

This example uses the ***EAP_HOME/domain/configuration/domain-alternate.xml*** configuration file.

Example 2.4. Using an Alternate Configuration file for a Managed Domain in Microsoft Windows Server

```
C:\EAP_HOME\bin> domain.bat --domain-config=domain-alternate.xml
```

This example uses the ***EAP_HOME\domain\configuration\domain-alternate.xml*** configuration file.

[Report a bug](#)

2.2.9. Stop JBoss EAP 6

The way that you stop JBoss EAP 6 depends on how it was started. This task covers stopping an instance that was started interactively, stopping an instance that was started by a service, and stopping an instance that was forked into the background by a script.

**NOTE**

For information on how to stop a server or server group in a Managed Domain see [Section 2.3.3, “Stop a Server Using the Management Console”](#) . For information on how to stop a server using the Management CLI, see [Section 2.3.1, “Start and Stop Servers Using the Management CLI”](#).

Procedure 2.8. Stop an instance of JBoss EAP 6

- **Stop an instance which was started interactively from a command prompt.**
Press **Ctrl-C** in the terminal where JBoss EAP 6 is running.

Procedure 2.9. Stop an instance which was started as an operating system service.

Depending on the operating system, use one of the following procedures.

- ○ **Red Hat Enterprise Linux**
For Red Hat Enterprise Linux, if you have written a service script, use its **stop** facility. This needs to be written into the script. Then you can use **service *scriptname* stop**, where *scriptname* is the name of the script.
- **Microsoft Windows Server**
In Microsoft Windows, use the **net service** command, or stop the service from the **Services** applet in the Control Panel.

Procedure 2.10. Stop an instance which is running in the background (Red Hat Enterprise Linux)

1. Obtain the process ID (PID) of the process:

- **If only a single instance is running (standalone mode)**

Either of the following commands will return the PID of a single instance of JBoss EAP 6:

- **pidof java**
- **jps**

(The **jps** command will return an ID for two processes; one for **jboss-modules.jar** and one for **jps** itself. Use the ID for **jboss-modules.jar** to stop the EAP instance)

- **If multiple EAP instances are running (domain mode)**

Identifying the correct process to end if more than one instance of EAP is running requires more comprehensive commands be used.

- The **jps** command can be used in verbose mode to provide more information about the java processes it finds.

Below is an abridged output from a verbose **jps** command identifying the different EAP processes running by PID and role:

```
$ jps -v
12155 jboss-modules.jar -D[Server:server-one] -XX:PermSize=256m -
XX:MaxPermSize=256m -Xms1303m
...
12196 jboss-modules.jar -D[Server:server-two] -XX:PermSize=256m -
```

```

XX:MaxPermSize=256m -Xms1303m
...

12096 jboss-modules.jar -D[Host Controller] -Xms64m -Xmx512m -
XX:MaxPermSize=256m
...

11872 Main -Xms128m -Xmx750m -XX:MaxPermSize=350m -
XX:ReservedCodeCacheSize=96m -XX:+UseCodeCacheFlushing
...

11248 jboss-modules.jar -D[Standalone] -XX:+UseCompressedOops -verbose:gc
...

12892 Jps
...

12080 jboss-modules.jar -D[Process Controller] -Xms64m -Xmx512m -
XX:MaxPermSize=256m
...

```

- The **ps aux** command can also be used to return information about multiple EAP instances.

Below is an abridged output from a verbose **ps aux** command identifying the different EAP processes running by PID and role:

```

$ ps aux | grep java
username 12080 0.1 0.9 3606588 36772 pts/0 Sl+ 10:09 0:01 /path/to/java -
D[Process Controller] -server -Xms128m -Xmx128m -XX:MaxPermSize=256m
...

username 12096 1.0 4.1 3741304 158452 pts/0 Sl+ 10:09 0:13 /path/to/java -
D[Host Controller] -Xms128m -Xmx128m -XX:MaxPermSize=256m
...

username 12155 1.7 8.9 4741800 344224 pts/0 Sl+ 10:09 0:22 /path/to/java -
D[Server:server-one] -XX:PermSize=256m -XX:MaxPermSize=256m -Xms1000m -
Xmx1000m -server -
...

username 12196 1.8 9.4 4739612 364436 pts/0 Sl+ 10:09 0:22 /path/to/java -
D[Server:server-two] -XX:PermSize=256m -XX:MaxPermSize=256m -Xms1000m -
Xmx1000m -server
...

```

In the above examples, the **Process Controller** processes are the processes to stop in order to stop the entire domain.

The **grep** utility can be used with either of these commands to identify the **Process Controller**:

```
jps -v | grep "Process Controller"
```

```
ps aux | grep "Process Controller"
```

- 2. Send the process the **TERM** signal, by running **kill *PID***, where *PID* is the process ID identified by one of the commands above.

Result

Each of these alternatives shuts JBoss EAP 6 down cleanly so that data is not lost.

[Report a bug](#)

2.2.10. Reference of Switches and Arguments to pass at Server Runtime

The application server startup script accepts arguments and switches at runtime. This allows the server to start under alternative configurations to those defined in the **standalone.xml**, **domain.xml**, and **host.xml** configuration files.

Alternative configurations might include starting the server with an alternative socket bindings set or a secondary configuration.

The available parameters list can be accessed by passing the help switch **-h** or **--help** at startup.

Table 2.2. Runtime Switches and Arguments

Argument or Switch	Mode	Description
--admin-only	Standalone	Set the server's running type to ADMIN_ONLY . This will cause it to open administrative interfaces and accept management requests, but not start other runtime services or accept end user requests.
--admin-only	Domain	Set the host controller's running type to ADMIN_ONLY causing it to open administrative interfaces and accept management requests but not start servers or, if this host controller is the master for the domain, accept incoming connections from slave host controllers.
-b=<value>, -b <value>	Standalone, Domain	Set system property jboss.bind.address , which is used in configuring the bind address for the public interface. This defaults to 127.0.0.1 if no value is specified. See the -b<interface>=<value> entry for setting the bind address for other interfaces.
-b<interface>=<value>	Standalone, Domain	Set system property jboss.bind.address.<interface> to the given value. For example, -bmanagement=IP_ADDRESS
--backup	Domain	Keep a copy of the persistent domain configuration even if this host is not the Domain Controller.
-c=<config>, -c <config>	Standalone	Name of the server configuration file to use. The default is standalone.xml .

Argument or Switch	Mode	Description
-c=<config>, -c <config>	Domain	Name of the server configuration file to use. The default is domain.xml .
--cached-dc	Domain	If the host is not the Domain Controller and cannot contact the Domain Controller at boot, boot using a locally cached copy of the domain configuration.
--debug [<port>]	Standalone	Activate debug mode with an optional argument to specify the port. Only works if the launch script supports it.
-D<name>[=<value>]	Standalone, Domain	Set a system property.
--domain-config=<config>	Domain	Name of the server configuration file to use. The default is domain.xml .
-h, --help	Standalone, Domain	Display the help message and exit.
--host-config=<config>	Domain	Name of the host configuration file to use. The default is host.xml .
--interprocess-hc-address= <address>	Domain	Address on which the host controller should listen for communication from the process controller.
--interprocess-hc-port= <port>	Domain	Port on which the host controller should listen for communication from the process controller.
--master-address= <address>	Domain	Set system property jboss.domain.master.address to the given value. In a default slave Host Controller config, this is used to configure the address of the master Host Controller.
--master-port=<port>	Domain	Set system property jboss.domain.master.port to the given value. In a default slave Host Controller config, this is used to configure the port used for native management communication by the master Host Controller.
--read-only-server-config= <config>	Standalone	Name of the server configuration file to use. This differs from --server-config and -c in that the original file is never overwritten.
--read-only-domain-config= <config>	Domain	Name of the domain configuration file to use. This differs from --domain-config and -c in that the initial file is never overwritten.

Argument or Switch	Mode	Description
--read-only-host-config=<config>	Domain	Name of the host configuration file to use. This differs from --host-config in that the initial file is never overwritten.
-P=<url>, -P <url>, --properties=<url>	Standalone, Domain	Load system properties from the given URL.
--pc-address=<address>	Domain	Address on which the process controller listens for communication from processes it controls.
--pc-port=<port>	Domain	Port on which the process controller listens for communication from processes it controls.
-S<name>[=<value>]	Standalone	Set a security property.
--server-config=<config>	Standalone	Name of the server configuration file to use. The default is standalone.xml .
-u=<value>, -u <value>	Standalone, Domain	Set system property jboss.default.multicast.address , which is used in configuring the multicast address in the socket-binding elements in the configuration files. This defaults to 230.0.0.4 if no value is specified.
-v, -V, --version	Standalone, Domain	Display the application server version and exit.



WARNING

The configuration files that ship with JBoss EAP 6 are set up to handle the behavior of the switches (i.e. **-b**, **-u**). If you change your configuration files to no longer use the system property controlled by the switch, then adding it to the launch command will have no effect.

[Report a bug](#)

2.3. START AND STOP SERVERS

2.3.1. Start and Stop Servers Using the Management CLI

Prerequisites

- [Section 3.4.2, “Launch the Management CLI”](#)

Servers can be started and stopped using the Management CLI or Management Console. Both tools can control a single Standalone Server instance or manage multiple servers across a Managed Domain deployment.

To use the Management Console, refer to [Section 2.3.2, “Start a Server Using the Management Console”](#). When using the Management CLI, the process varies for Standalone Server and Managed Domain instances.

Stop a Standalone Server with the Management CLI

Standalone Servers, started either by a script or manually at a shell prompt, can be shut down from the Management CLI using the **shutdown** command.

Example 2.5. Stop a Standalone Server instance via the Management CLI

```
[standalone@localhost:9999 /] shutdown
```

To restart the JBoss EAP 6 Standalone Server instance, run the instance’s startup script or start it manually as described in [Section 2.2.2, “Start JBoss EAP 6 as a Standalone Server”](#).

Start and Stop a Managed Domain with the Management CLI

The Management Console can selectively start or stop specific servers in a domain. This includes server groups across the whole of a domain as well as specific server instances on a host.

Example 2.6. Stop a Server Host in a Managed Domain via the Management CLI

Similar to Standalone Server instance, the **shutdown** command is used to shut down a declared Managed Domain host. This example stops a server host named *master* by declaring the instance name before calling the shutdown operation.

```
[domain@localhost:9999 /] shutdown --host=master
```

Example 2.7. Start and Stop a Server Group in a Managed Domain via the Management CLI

This example starts a default server group named *main-server-group* by declaring the group before calling the **start** and **stop** operations.

```
[domain@localhost:9999 /] /server-group=main-server-group:start-servers
```

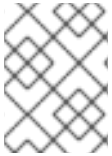
```
[domain@localhost:9999 /] /server-group=main-server-group:stop-servers
```

Example 2.8. Start and Stop a Server Instance in a Managed Domain via the Management CLI

This example starts and then stops a server instance named *server-one* on the *master* host by declaring the host and server configuration before calling the **start** and **stop** operations.

```
[domain@localhost:9999 /] /host=master/server-config=server-one:start
```

```
[domain@localhost:9999 /] /host=master/server-config=server-one:stop
```

**NOTE**

Use the **tab** key to assist with string completion and to expose visible variables such as available host and server configuration values.

[Report a bug](#)

2.3.2. Start a Server Using the Management Console

Prerequisites

- [Section 2.2.4, "Start JBoss EAP 6 as a Managed Domain"](#)
- [Section 3.3.2, "Log in to the Management Console"](#)

Procedure 2.11. Start the Server for a Managed Domain

1. Select the **Domain** tab at the top of the console and then, select the **TOPOLOGY** tab. In the left navigation bar, under **Domain**, select **Overview**.
2. From the list of **Server Instances**, select the server you want to start. Servers that are running are indicated by a check mark.

Hover the cursor over an instance in this list to show options in blue text below the server's details.

3. To start the instance, click on the **Start Server** text when it appears. A confirmation dialogue box will open. Click **Confirm** to start the server.

Result

The selected server is started and running.

[Report a bug](#)

2.3.3. Stop a Server Using the Management Console

Prerequisites

- [Section 2.2.4, "Start JBoss EAP 6 as a Managed Domain"](#)
- [Section 3.3.2, "Log in to the Management Console"](#)

Procedure 2.12. Stop a Server in a Managed Domain Using the Management Console

1. Select the **Domain** tab at the top of the console and then, select the **TOPOLOGY** tab. In the left navigation bar, under **Domain**, select **Overview**.
2. A list of available **Server Instances** is displayed on the **Hosts, groups and server instances** table. Servers that are running are indicated by a check mark.

3. Hover the cursor over the chosen server. Click on the **Stop Server** text that appears. A confirmation dialogue window will appear.
4. Click **Confirm** to stop the server.

Result

The selected server is stopped.

[Report a bug](#)

2.4. CONFIGURATION FILES

2.4.1. About JBoss EAP 6 Configuration Files

The configuration for JBoss EAP 6 has changed considerably from previous versions. One of the most obvious differences is the use of a simplified configuration file structure, which includes one or more of the files listed below.

Table 2.3. Configuration File Locations

Server mode	Location	Purpose
domain.xml	<i>EAP_HOME/domain/configuration/domain.xml</i>	This is the main configuration file for a managed domain. Only the domain master reads this file. On other domain members, it can be removed.
host.xml	<i>EAP_HOME/domain/configuration/host.xml</i>	This file includes configuration details specific to a physical host in a managed domain, such as network interfaces, socket bindings, the name of the host, and other host-specific details. The host.xml file includes all of the features of both host-master.xml and host-slave.xml , which are described below. This file is not present for standalone servers.
host-master.xml	<i>EAP_HOME/domain/configuration/host-master.xml</i>	This file includes only the configuration details necessary to run a server as a managed domain master server. This file is not present for standalone servers.
host-slave.xml	<i>EAP_HOME/domain/configuration/host-slave.xml</i>	This file includes only the configuration details necessary to run a server as a managed domain slave server. This file is not present for standalone servers.

Server mode	Location	Purpose
standalone.xml	<i>EAP_HOME/standalone/configuration/standalone.xml</i>	This is the default configuration file for a standalone server. It contains all information about the standalone server, including subsystems, networking, deployments, socket bindings, and other configurable details. This configuration is used automatically when you start your standalone server.
standalone-full.xml	<i>EAP_HOME/standalone/configuration/standalone-full.xml</i>	This is an example configuration for a standalone server. It includes support for every possible subsystem except for those required for high availability. To use it, stop your server and restart using the following command: <i>EAP_HOME/bin/standalone.sh -c standalone-full.xml</i>
standalone-ha.xml	<i>EAP_HOME/standalone/configuration/standalone-ha.xml</i>	This example configuration file enables all of the default subsystems and adds the mod_cluster and JGroups subsystems for a standalone server, so that it can participate in a high-availability or load-balancing cluster. This file is not applicable for a managed domain. To use this configuration, stop your server and restart using the following command: <i>EAP_HOME/bin/standalone.sh -c standalone-ha.xml</i>
standalone-full-ha.xml	<i>EAP_HOME/standalone/configuration/standalone-full-ha.xml</i>	This is an example configuration for a standalone server. It includes support for every possible subsystem, including those required for high availability. To use it, stop your server and restart using the following command: <i>EAP_HOME/bin/standalone.sh -c standalone-full-ha.xml</i>

These are only the default locations. You can specify a different configuration file at runtime.

**NOTE**

For information about how to backup JBoss EAP 6 configuration data, refer [Section 2.4.2, “Back up JBoss EAP Configuration Data”](#) .

[Report a bug](#)

2.4.2. Back up JBoss EAP Configuration Data

Summary

This topic describes the files that must be backed up in order to later restore the JBoss EAP server configuration.

Procedure 2.13. Back Up the Configuration Data

1. To keep user and profile data, domain, host, slave, and logging configuration, back up the entire contents of the following directories.
 - *EAP_HOME*/standalone/configuration/
 - *EAP_HOME*/domain/configuration
2. Back up any custom modules created in the *EAP_HOME/modules/system/layers/base/* directory.
3. Back up any welcome content in the *EAP_HOME/welcome-content/* directory.
4. Back up any custom scripts created in the *EAP_HOME/bin/* directory.

[Report a bug](#)

2.4.3. Descriptor-based Property Replacement

Application configuration - for example, datasource connection parameters - typically varies between development, testing, and production deployments. This variance is sometimes accommodated by build system scripts, as the Java EE specification does not contain a method to externalize these configurations. With JBoss EAP 6 you can use *descriptor-based property replacement* to manage configuration externally.

Descriptor-based property replacement substitutes properties based on descriptors, allowing you to remove assumptions about the environment from the application and the build chain. Environment-specific configurations can be specified in deployment descriptors rather than annotations or build system scripts. You can provide configuration in files or as parameters at the command line.

Descriptor-based property replacement is enabled globally through **standalone.xml** or **domain.xml**:

Example 2.9. Descriptor-based property replacement

```
<subsystem xmlns="urn:jboss:domain:ee:1.2">
  <spec-descriptor-property-replacement>
    true
  </spec-descriptor-property-replacement>
  <jboss-descriptor-property-replacement>
```

```

    true
  </jboss-descriptor-property-replacement>
</subsystem>

```

Java EE descriptor replacement is *disabled* by default. When enabled, descriptors can be replaced in the following configuration files: **ejb-jar.xml** and **persistence.xml**.

JBoss-specific descriptor replacement is *enabled* by default. When enabled, descriptors can be replaced in the following configuration files:

- **jboss-ejb3.xml**
- **jboss-app.xml**
- **jboss-web.xml**
- ***-jms.xml**
- ***-ds.xml**

For example, given a Bean with the following annotation:

Example 2.10. Example annotation

```

@ActivationConfigProperty(propertyName = "connectionParameters", propertyValue =
"host=192.168.1.1;port=5445")

```

With descriptor-based property replacement enabled, the **connectionParameters** can be specified via the command-line as:

```

./standalone.sh -DconnectionParameters='host=10.10.64.1;port=5445'

```

To accomplish the same via system properties you use an *expression* in place of the literal value. Expressions take the format **`\${parameter:default}**. Where an expression is used in configuration, the value of that parameter takes its place. If the parameter does not exist then the specified default value is used instead.

Example 2.11. Using an Expression in a Descriptor

```

<activation-config>
  <activation-config-property>
    <activation-config-property-name>
      connectionParameters
    </activation-config-property-name>
    <activation-config-property-value>
      ${jms.connection.parameters:'host=10.10.64.1;port=5445'}
    </activation-config-property-value>
  </activation-config-property>
</activation-config>

```

The expression **`\${jms.connection.parameters:'host=10.10.64.1;port=5445'}** allows the connection parameters to be overridden by a command-line supplied parameter, while providing a default value.

[Report a bug](#)

2.4.4. Enabling or Disabling Descriptor Based Property Replacement

Summary

Finite control over descriptor property replacement was introduced in **jboss-as-ee_1_1.xsd**. This task covers the steps required to configure descriptor based property replacement.

Prerequisites

- [Section 2.2.1, "Start JBoss EAP 6"](#)
- [Section 3.4.2, "Launch the Management CLI"](#)

Descriptor based property replacement flags have boolean values:

- When set to **true**, property replacements are enabled.
- When set to **false**, property replacements are disabled.

Procedure 2.14. **jboss-descriptor-property-replacement**

jboss-descriptor-property-replacement is used to enable or disable property replacement in the following descriptors:

- **jboss-ejb3.xml**
- **jboss-app.xml**
- **jboss-web.xml**
- ***-jms.xml**
- ***-ds.xml**

The default value for **jboss-descriptor-property-replacement** is **true**.

1. In the Management CLI, run the following command to determine the value of **jboss-descriptor-property-replacement**:

```
/subsystem=ee:read-attribute(name="jboss-descriptor-property-replacement")
```

2. Run the following command to configure the behavior:

```
/subsystem=ee:write-attribute(name="jboss-descriptor-property-replacement",value=VALUE)
```

Procedure 2.15. **spec-descriptor-property-replacement**

spec-descriptor-property-replacement is used to enable or disable property replacement in the following descriptors:

- **ejb-jar.xml**

- **persistence.xml**
- **application.xml**
- **web.xml**

The default value for **spec-descriptor-property-replacement** is **false**.

1. In the Management CLI, run the following command to confirm the value of **spec-descriptor-property-replacement**:

```
/subsystem=ee:read-attribute(name="spec-descriptor-property-replacement")
```

2. Run the following command to configure the behavior:

```
/subsystem=ee:write-attribute(name="spec-descriptor-property-replacement",value=VALUE)
```

Result

The descriptor based property replacement tags have been successfully configured.

[Report a bug](#)

2.4.5. Nested Expressions

Expressions may be nested, which allows for more advanced use of expressions in place of fixed values. The format of a nested expression is like that of a normal expression, but one expression is embedded in the other, for example:

Example 2.12. Nested expression

```
#{system_value_1#{system_value_2}}
```

Nested expressions are evaluated recursively, so the *inner* expression is first evaluated, then the *outer* expression is evaluated. Nested expressions are permitted anywhere that expressions are permitted, with the exception of Management CLI commands.

As for normal expressions, the supported sources for resolving nested expressions are: system properties, environment variables and the Vault. For deployments only, the source can be properties listed in a **META-INF/jboss.properties** file in the deployment archive. In an EAR or other deployment type that supports subdeployments, the resolution is scoped to all subdeployments if the **META-INF/jboss.properties** is in the outer deployment (e.g. the EAR) and is scoped to a subdeployment if **META-INF/jboss.properties** is in the subdeployment archive (e.g. a WAR inside an EAR.)

Example 2.13. Use a Nested Expression in a Configuration File

A real-life application of a nested expression is in a datasource definition. If the password used in a datasource definition is masked, the resulting line in the datasource definition might be as follows:

```
<password>#{VAULT::ds_ExampleDS::password::1}</password>
```

Using a nested expression, the value of **ds_ExampleDS** could be replaced with a system property. If a system property **datasource_name** is assigned the value **ds_ExampleDS**, the line in the datasource definition could instead be as follows:

```
<password>${VAULT::${datasource_name}::password::1}</password>
```

JBoss EAP would first evaluate the expression **\${datasource_name}**, then input this to the larger expression and evaluate the resulting expression. The advantage of this configuration is that the name of the datasource is abstracted from the fixed configuration.

Expressions may also be recursive, where an expressions resolves to an expression which is then resolved. Nested expressions and recursive expressions are a form of indirection. Note that recursive expressions are not permitted in Management CLI commands.

Example 2.14. Recursive Expression

Continuing the previous example, you might use the expression **\${foo}** which resolves to the expression **\${VAULT::ds_ExampleDS::password::1}**, which then resolves to a value contained in the Vault: **secret**.

[Report a bug](#)

2.4.6. Configuration File History

The application server configuration files include **standalone.xml**, as well as the **domain.xml** and **host.xml** files. While these files may be modified by direct editing, the recommended method is to configure the application server model with the available management operations, including the Management CLI and the Management Console.

To assist in the maintenance and management of the server instance, the application server creates a timestamped version of the original configuration file at the time of startup. Any additional configuration changes made by management operations result in the original file being automatically backed up, and a working copy of the instance being preserved for reference and rollback. This archival functionality extends to saving, loading and deleting snapshots of the server configuration to allow for recall and rollback scenarios.

- [Section 2.4.7, "Start the Server with a Previous Configuration"](#)
- [Section 2.4.8, "Save a Configuration Snapshot Using the Management CLI"](#)
- [Section 2.4.9, "Load a Configuration Snapshot Using the Management CLI"](#)
- [Section 2.4.10, "Delete a Configuration Snapshot Using Management CLI"](#)
- [Section 2.4.11, "List All Configuration Snapshots Using Management CLI"](#)

[Report a bug](#)

2.4.7. Start the Server with a Previous Configuration

The following example shows how to start the application server with a previous configuration in a standalone server with **standalone.xml**. The same concept applies to a managed domain with **domain.xml** and **host.xml** respectively.

This example recalls a previous configuration saved automatically by the application server as management operations modify the server model.

Example 2.15. Start the server with a saved configuration

1. Identify the backed up version that you want to start. This example will recall the instance of the server model prior to the first modification after successfully booting up.

```
EAP_HOME/standalone/configuration/standalone_xml_history/current/standalone.v1.xml
```

2. Start the server with this configuration of the backed up model by passing in the relative filename under **jboss.server.config.dir**.

```
EAP_HOME/bin/standalone.sh --server-  
config=standalone_xml_history/current/standalone.v1.xml
```

Result

The application server starts with the selected configuration.

**NOTE**

The domain configuration history is located in **EAP_HOME/domain/configuration/domain_xml_history/current/domain.v1.xml**

Start the server with this configuration of the backed up model by passing the relative filename under **jboss.domain.config.dir**.

To start the domain with this configuration:

```
EAP_HOME/bin/domain.sh --domain-  
config=domain_xml_history/current/domain.v1.xml
```

[Report a bug](#)

2.4.8. Save a Configuration Snapshot Using the Management CLI**Summary**

Configuration snapshots are a point-in-time copy of the current server configuration. These copies can be saved and loaded by the administrator.

The following example uses the **standalone.xml** configuration file, but the same process applies to the **domain.xml** and **host.xml** configuration files.

Prerequisites

- [Section 3.4.2, "Launch the Management CLI"](#)

Procedure 2.16. Take a Configuration Snapshot and Save It

- **Save a snapshot**
Run the **take-snapshot** operation to capture a copy of the current server configuration.

```
[standalone@localhost:9999 /] :take-snapshot
```



```
{
  "outcome" => "success",
  "result" =>
  "/home/User/EAP_HOME/standalone/configuration/standalone_xml_history/snapshot/2011063
  0-172258657standalone.xml"
```

Result

A snapshot of the current server configuration has been saved.

[Report a bug](#)

2.4.9. Load a Configuration Snapshot Using the Management CLI

Configuration snapshots are a point-in-time copy of the current server configuration. These copies can be saved and loaded by the administrator. The process of loading snapshots is similar to the method used to [Section 2.4.7, “Start the Server with a Previous Configuration”](#), running from the command line rather than the Management CLI interface used to create, list and delete snapshots.

The following example uses the **standalone.xml** file, but the same process applies to the **domain.xml** and **host.xml** files.

Procedure 2.17. Load a Configuration Snapshot

1. Identify the snapshot to be loaded. This example will recall the following file from the snapshot directory. The default path for the snapshot files is as follows.

```
EAP_HOME/standalone/configuration/standalone_xml_history/snapshot/20110812-
191301472standalone.xml
```

The snapshots are expressed by their relative paths, by which the above example can be written as follows.

```
jboss.server.config.dir/standalone_xml_history/snapshot/20110812-
191301472standalone.xml
```

2. Start the server with the selected configuration snapshot by passing in the filename.

```
EAP_HOME/bin/standalone.sh --server-config=standalone_xml_history/snapshot/20110913-
164449522standalone.xml
```

Result

The server restarts with the configuration selected in the loaded snapshot.

[Report a bug](#)

2.4.10. Delete a Configuration Snapshot Using Management CLI

Prerequisites

- [Section 3.4.2, “Launch the Management CLI”](#)

Configuration snapshots are a point-in-time copy of the current server configuration. These copies can be saved and loaded by the administrator.

The following examples use the **standalone.xml** file, but the same process applies to the **domain.xml** and **host.xml** files.

Procedure 2.18. Delete a Specific Snapshot

1. Identify the snapshot to be deleted. This example will delete the following file from the snapshot directory.

```
EAP_HOME/standalone/configuration/standalone_xml_history/snapshot/20110630-165714239standalone.xml
```

2. Run the **:delete-snapshot** command to delete a specific snapshot, specifying the name of the snapshot as in the example below.

```
[standalone@localhost:9999 /] :delete-snapshot(name="20110630-165714239standalone.xml")
{"outcome" => "success"}
```

Result

The snapshot has been deleted.

Procedure 2.19. Delete All Snapshots

- Run the **:delete-snapshot(name="all")** command to delete all snapshots as in the example below.

```
[standalone@localhost:9999 /] :delete-snapshot(name="all")
{"outcome" => "success"}
```

Result

All snapshots have been deleted.

[Report a bug](#)

2.4.11. List All Configuration Snapshots Using Management CLI

Prerequisites

- [Section 3.4.2, "Launch the Management CLI"](#)

Configuration snapshots are a point-in-time copy of the current server configuration. These copies can be saved and loaded by the administrator.

The following example uses the **standalone.xml** file, but the same process applies to the **domain.xml** and **host.xml** files.

Procedure 2.20. List All Configuration Snapshots

- **List all snapshots**

List all of the saved snapshots by running the **:list-snapshots** command.

```
[standalone@localhost:9999 /] :list-snapshots
{
  "outcome" => "success",
  "result" => {
    "directory" =>
"/home/hostname/EAP_HOME/standalone/configuration/standalone_xml_history/snapshot",
    "names" => [
      "20110818-133719699standalone.xml",
      "20110809-141225039standalone.xml",
      "20110802-152010683standalone.xml",
      "20110808-161118457standalone.xml",
      "20110912-151949212standalone.xml",
      "20110804-162951670standalone.xml"
    ]
  }
}
```

Result

The snapshots are listed.

[Report a bug](#)

2.5. FILESYSTEM PATHS

JBoss EAP 6 uses logical names for filesystem paths. The **domain.xml**, **host.xml**, and **standalone.xml** configuration files each include a section for declaring paths.

Other sections of each file can then reference the paths using their logical name, avoiding the need to use absolute paths for each instance and allowing specific host configurations to resolve to universal logical names.

The default logging subsystem configuration, for example, declares **jboss.server.log.dir** as the logical name for the server's **log** directory.

Example 2.16. Relative path example for the logging directory

```
<file relative-to="jboss.server.log.dir" path="server.log"/>
```

JBoss EAP 6 automatically provides a number of standard paths without any need for the user to configure them in a configuration file.

Table 2.4. Standard Paths

Value	Description
java.ext.dirs	The Java development kit extension directory paths.
jboss.home.dir	The root directory of the JBoss EAP 6 distribution.

Value	Description
user.home	The user home directory.
user.dir	The user's current working directory.
java.home	The Java installation directory
jboss.server.base.dir	The root directory for an individual server instance.
jboss.server.data.dir	The directory the server will use for persistent data file storage.
jboss.server.config.dir	The directory that contains the server configuration.
jboss.server.log.dir	The directory the server will use for log file storage.
jboss.server.temp.dir	The directory the server will use for temporary file storage.
jboss.server.deploy.dir	The directory that the server will use for storing deployed content.
jboss.controller.temp.dir	The directory the host controller will use for temporary file storage.
jboss.domain.base.dir	The base directory for domain content.
jboss.domain.config.dir	The directory that contains the domain configuration.
jboss.domain.data.dir	The directory that the domain will use for persistent data file storage.
jboss.domain.log.dir	The directory that the domain will use for persistent log file storage.
jboss.domain.temp.dir	The directory that the domain will use for temporary file storage.
jboss.domain.deployment.dir	The directory that the domain will use for storing deployed content.
jboss.domain.servers.dir	The directory that the domain will use for storing outputs of the managed domain instances.

Override a Path

If you are running a standalone server, you can override all the **jboss.server.*** paths in one of the two ways.

- You can pass command line arguments when you start the server. For example:

```
bin/standalone.sh -Djboss.server.log.dir=/var/log
```

- You can modify the **JAVA_OPTS** variable in the server configuration file. Open the **EAP_HOME/bin/standalone.conf** file and add the following line at the end of the file:

```
JAVA_OPTS="$JAVA_OPTS -Djboss.server.log.dir=/var/log"
```

Path overrides is supported for servers running in a managed domain. For example, the **jboss.domain.servers.dir** can be used to change the base directories of servers in a managed domain.

Add a Custom Path

You can also create your own custom path. For example, you may want to define a relative path to use for logging. You can then change the log handler to use **my.relative.path**,

Example 2.17. A custom logging path

```
my.relative.path=/var/log
```

[Report a bug](#)

2.5.1. Directory Grouping

In domain mode, each server's files are stored in the **EAP_HOME/domain/** directory. Subdirectories are named according to the **directory-grouping** attribute, either by server or file type.

Directory Grouping by Server

The default directory grouping is **by server**. If your administration is *server-centric*, this configuration is recommended. For example, it allows backups and log file handling to be configured per server instance.

Example 2.18. Directory Grouping by Server

If JBoss EAP is installed using the Zip method and all default options apply, the directory structure in domain mode will be as follows.

```
EAP_HOME/domain
├── servers
│   ├── server-one
│   │   ├── data
│   │   ├── tmp
│   │   └── log
│   └── server-two
│       ├── data
│       ├── tmp
│       └── log
```

If the **directory-grouping** attribute has been changed from the default, and you want to reset it, enter the following management CLI command.

```
/host=master:write-attribute(name="directory-grouping",value="by-server")
```

This will update the controller's **host.xml** configuration file:

```
<servers directory-grouping="by-server">
  <server name="server-one" group="main-server-group" >
  </server>
  <server name="server-two" group="main-server-group" auto-start="true">
  </server>
</servers>
```

Directory Grouping by Type

Instead of grouping each servers' directories by server, you can instead group them by file type. If your administration is *file type-centric*, this configuration is recommended. For example, backup configuration is simpler if you want to include only data files.

To group domain data directories by *type*, enter the following management CLI command:

```
/host=master:write-attribute(name="directory-grouping",value="by-type")
```

This will update the controller's **host.xml** configuration file:

```
<servers directory-grouping="by-type">
  <server name="server-one" group="main-server-group" >
  </server>
  <server name="server-two" group="main-server-group" auto-start="true">
  </server>
</servers>
```

Example 2.19. Directory Grouping by Type

If JBoss EAP is installed using the Zip method and the domain's files are grouped by type, the directory structure in domain mode will be as follows.

```
EAP_HOME/domain
├── data
│   └── servers
│       ├── server-one
│       └── server-two
├── log
│   └── servers
│       ├── server-one
│       └── server-two
└── tmp
    └── servers
        ├── server-one
        └── server-two
```

[Report a bug](#)

2.5.2. Use Case: Overriding Directories

In this example, the objective is to store domain files in the `/opt/jboss_eap/data/domain_data` directory, and give each top-level directory a custom name. The directory grouping used is the default: **by-server**.

- Log files stored in the subdirectory **all_logs**
- Data files stored in the subdirectory **all_data**
- Temporary files stored in the subdirectory **all_temp**
- Servers' files stored in the subdirectory **all_servers**

To achieve this configuration, you would override several system properties when starting JBoss EAP.

```
./domain.sh \
-Djboss.domain.temp.dir=/opt/jboss_eap/data/domain_data/all_temp \
-Djboss.domain.log.dir=/opt/jboss_eap/data/domain_data/all_logs \
-Djboss.domain.data.dir=/opt/jboss_eap/data/domain_data/all_data\
-Djboss.domain.servers.dir=/opt/jboss_eap/data/domain_data/all_servers
```

The resulting path structure will be as follows:

```
/opt/jboss_eap/data/domain_data/
├── all_data
│   ├── content
│   └── logging.properties
├── all_logs
│   ├── host-controller.log
│   └── process-controller.log
├── all_servers
│   ├── server-one
│   │   ├── data
│   │   │   ├── content
│   │   │   └── logging.properties
│   │   ├── log
│   │   │   └── server.log
│   │   └── tmp
│   │       ├── vfs
│   │       │   ├── temp
│   │       │   └── work
│   │       └── jboss.web
│   │           └── default-host
│   └── server-two
│       ├── data
│       │   ├── content
│       │   └── logging.properties
│       ├── log
│       │   └── server.log
│       └── tmp
│           ├── vfs
│           │   ├── temp
│           │   └── work
│           └── jboss.web
│               └── default-host
```

┌── all_temp
└── auth
 ...

[Report a bug](#)

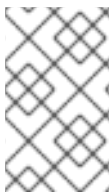
CHAPTER 3. MANAGEMENT INTERFACES

3.1. MANAGE THE APPLICATION SERVER

JBoss EAP 6 uses XML files for server configuration and offers three approaches to configuring and managing JBoss EAP 6 servers; a web interface, a command line client and direct editing of the XML configuration files.

The recommended methods for editing the configuration files are the Management CLI and the new, web-based Management Console. Edits made to the XML configuration files is still possible but configuration and administration through the Management Console and Management CLI provides extra validation and advanced features for server instance management.

Edits made to a server configuration by any of the three approaches are synchronized across the different views.



NOTE

However, edits made to the XML configuration files while a server instance is running will be overwritten by the server model. All comments added to an XML configuration file while a server instance is running will be removed as well.

To manage servers through a graphical user-interface in a web browser, use the Management Console. To manage servers through a command line interface, use the Management CLI.

As well as being the recommended management tools, the Management Console and Management CLI also serve as examples of the underlying Management API that enables expert users to develop their own tools if they desire.

[Report a bug](#)

3.2. MANAGEMENT APPLICATION PROGRAMMING INTERFACES (APIS)

HTTP API

The Management Console is a web interface built with the Google Web Toolkit (GWT). It communicates with the server using the HTTP management interface.

The HTTP API endpoint is the entry point for management clients which rely on the HTTP protocol to integrate with the management layer.

Management clients that rely on the HTTP protocol use a JSON encoded protocol and a de-typed, RPC-style API to describe and execute management operations against a Managed Domain or Standalone Server.

The HTTP API is used by the Management Console but offers integration capabilities for other clients as well.

The HTTP API endpoint is co-located with the domain controller or the standalone server instance. It serves two different contexts: one for executing management operations and the other to access the web interface. By default, the HTTP API endpoint runs on port 9990.

Example 3.1. HTTP API Configuration File Example

```

<management-interfaces>
  [...]
  <http-interface security-realm="ManagementRealm">
    <socket-binding http="management-http"/>
  </http-interface>
</management-interfaces>

```

The Management Console is served on the same port as the HTTP management API. It is important to distinguish between the Management Console as accessed on a default localhost, the Management Console as accessed remotely by a specific host and port combination, and the exposed domain API.

Table 3.1. URLs to access the Management Console or exposed HTTP API

URL	Description
http://localhost:9990/console	The Management Console accessed on the local host, controlling the Managed Domain configuration.
http://hostname:9990/console	The Management Console accessed remotely, naming the host and controlling the Managed Domain configuration.
http://hostname:9990/management	The HTTP Management API runs on the same port as the Management Console, displaying the raw attributes and values exposed to the API.

Example 3.2. Retrieve attribute values using the HTTP API

The following URL retrieves the HTTP web connector attribute values (the default operation is **read-resource**).

```
http://hostname:9990/management/subsystem/web/connector/http
```

Example 3.3. Retrieve a single attribute value using the HTTP API

The following URL retrieves the **enabled** attribute for the ExampleDS datasource.

```
http://hostname:9990/management/subsystem/datasources/data-source/ExampleDS?
operation=attribute&name=enabled
```

See [Section 10.4.1, "Deploy an application using the HTTP API"](#) for instructions on deploying applications using the HTTP API.

Native API

The Management CLI is a Native API tool. It is available for a Managed Domain or Standalone server instance, allowing an administrator to connect to a domain controller or Standalone Server instance and execute management operations available through the de-typed management model.

The Native API endpoint is the entry point for management clients that rely on the native protocol to integrate with the management layer. It uses an open binary protocol and an RPC-style API based on a very small number of Java types to describe and execute management operations. It is used by the Management CLI management tool, but offers integration capabilities for a wide range of other clients too.

The Native API endpoint is co-located with either a host controller or a Standalone Server. It must be enabled to use the Management CLI. It runs on port 9999 by default.

Example 3.4. Native API Configuration File Example

```
<management-interfaces>
  <native-interface security-realm="ManagementRealm">
    <socket-binding native="management-native"/>
  </native-interface>
  [...]
</management-interfaces>
```

[Report a bug](#)

3.3. THE MANAGEMENT CONSOLE

3.3.1. Management Console

The Management Console is a web-based administration tool for JBoss EAP 6.

Use the Management Console to start and stop servers, deploy and undeploy applications, tune system settings, and make persistent modifications to the server configuration. The Management Console also has the ability to perform administrative tasks, with live notifications when any changes require the server instance to be restarted or reloaded.

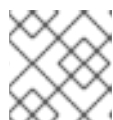
In a Managed Domain, server instances and server groups in the same domain can be centrally managed from the Management Console of the domain controller.

[Report a bug](#)

3.3.2. Log in to the Management Console

Prerequisites

- JBoss EAP 6 must be running.
 - You must have already created a user with permissions to access the Console.
1. Launch your web browser and go to this address: <http://localhost:9990/console/App.html>



NOTE

Port 9990 is predefined as the Management Console socket binding.

2. Enter your username and password to log in to the Management Console.

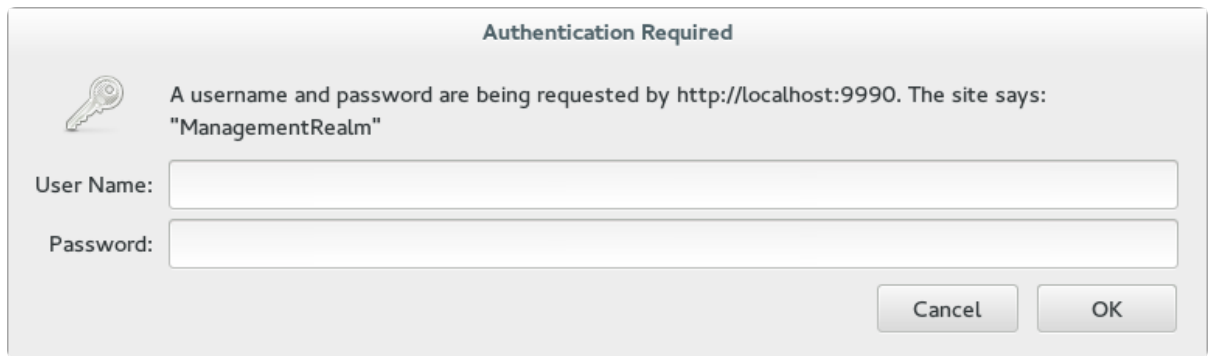


Figure 3.1. Log in screen for the Management Console

Result

Once logged in, you are redirected to the following address and the Management Console landing page appears: <http://localhost:9990/console/App.html#home>

[Report a bug](#)

3.3.3. Change the Language of the Management Console

The language settings of web-based Management Console use English by default. You can choose to use one of the following languages instead.

Supported Languages

- German (de)
- Simplified Chinese (zh-Hans)
- Brazilian Portuguese (pt-BR)
- French (fr)
- Spanish (es)
- Japanese (ja)

Procedure 3.1. Change the Language of the Web-based Management Console

1. **Log into the Management Console.**
Log into the web-based Management Console.
2. **Open the Settings dialog.**
Near the bottom right of the screen is a **Settings** label. Click it to open the settings for the Management Console.
3. **Select the desired language.**
Select the desired language from the **Locale** selection box. Select **Save**. A confirmation box informs you that you need to reload the application. Click **Confirm**. The system refreshes your web browser automatically to use the new locale.

[Report a bug](#)

3.3.4. Analytics in JBoss EAP Console

About Google Analytics

Google Analytics is a free web analytics service which provides comprehensive usage statistics on a website. It provides vital data regarding a site's visitors, including their visits, page views, pages per visit and average time spent on site. Google Analytics provides more visibility around a website's presence and its visitors.

About Google Analytics in JBoss EAP Management Console

JBoss EAP 6 provides users the option to enable or disable Google Analytics in the management console. The Google Analytics feature aims to help Red Hat EAP team understand how the customers are using the console and which parts of the console matter the most to the customers. This information will in-turn help the team adapt the console design, features and content to the immediate needs of the customers.



NOTE

By default, Google Analytics is disabled in JBoss EAP 6 console and its usage is optional.

[Report a bug](#)

3.3.5. Enable Google Analytics in JBoss EAP Console

To enable Google Analytics in JBoss EAP Management Console:

- Log in to the Management Console
- Click **Settings** on the Management Console

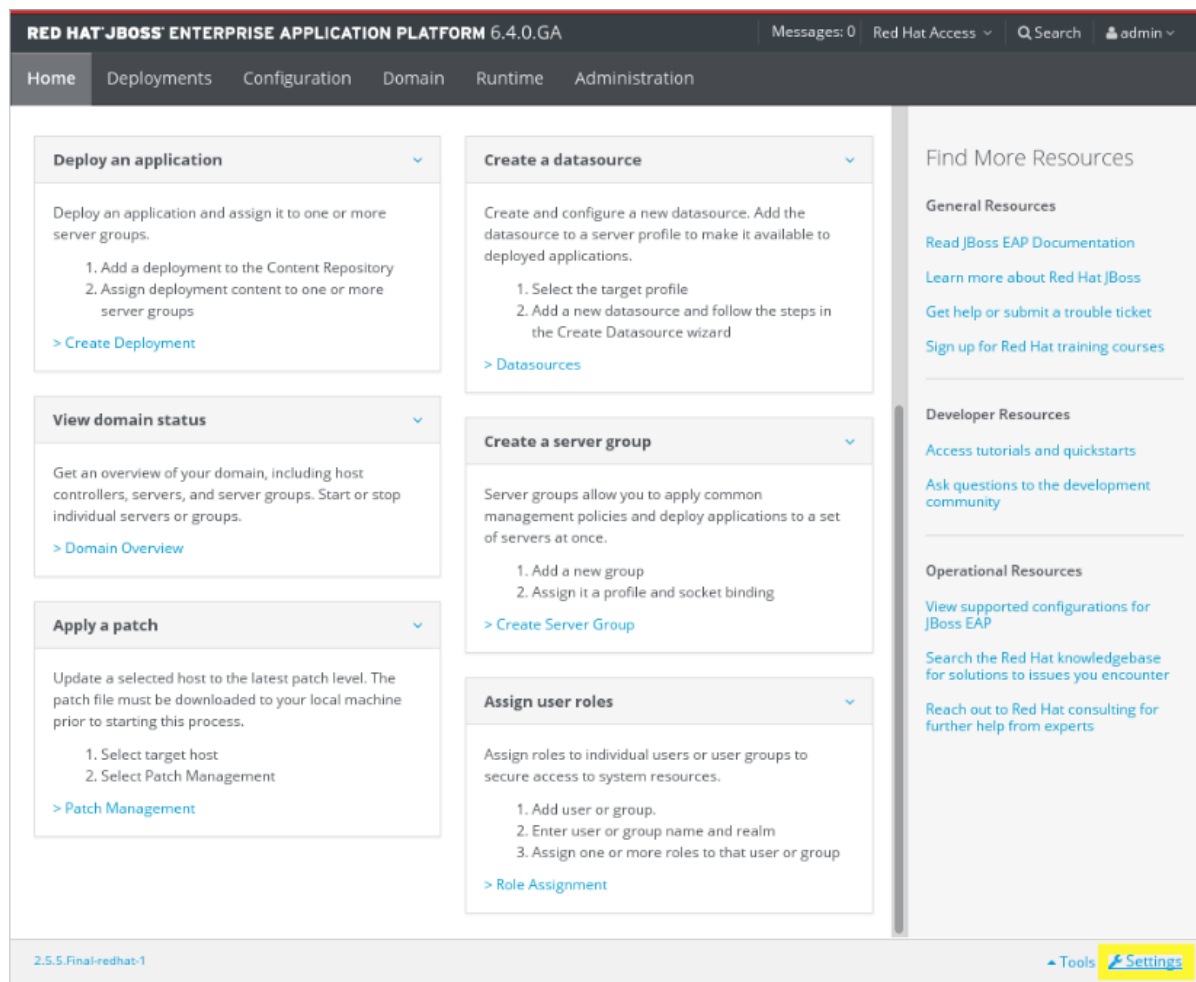


Figure 3.2. Log in screen of the Management Console

- Select **Enable Usage Data Collection** checkbox on the **Settings** dialog and click **Save** button. Confirm the application reload to activate the new settings.

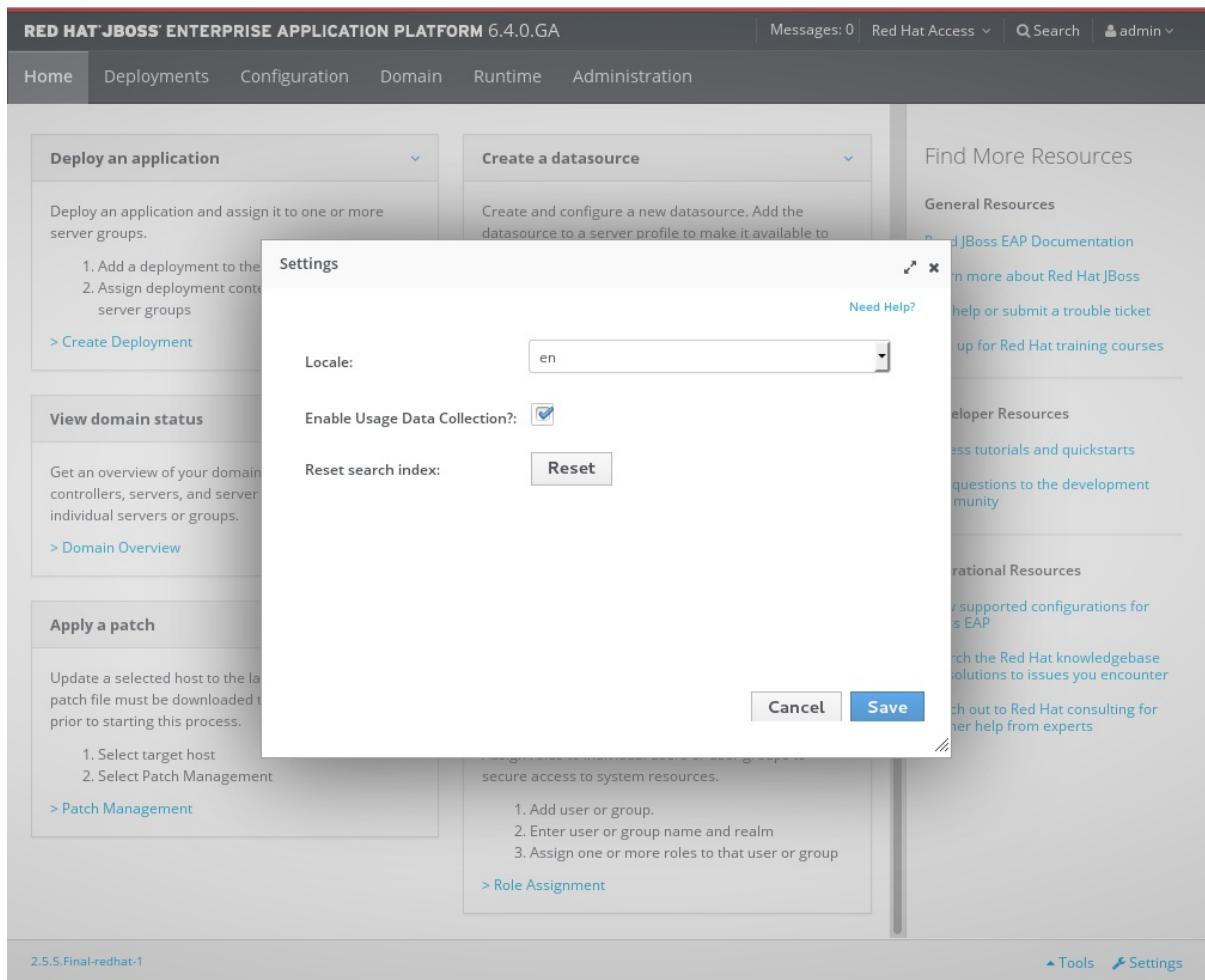


Figure 3.3. Settings dialog (Enable Usage Data Collection)

[Report a bug](#)

3.3.6. Disable Google Analytics in JBoss EAP Console

To disable Google Analytics in JBoss EAP Management Console:

- Log in to the Management Console
- Click **Settings** on the Management Console

RED HAT JBOSS ENTERPRISE APPLICATION PLATFORM 6.4.0.GA Messages: 0 Red Hat Access Search admin

Home Deployments Configuration Domain Runtime Administration

Deploy an application

Deploy an application and assign it to one or more server groups.

1. Add a deployment to the Content Repository
2. Assign deployment content to one or more server groups

> Create Deployment

View domain status

Get an overview of your domain, including host controllers, servers, and server groups. Start or stop individual servers or groups.

> Domain Overview

Apply a patch

Update a selected host to the latest patch level. The patch file must be downloaded to your local machine prior to starting this process.

1. Select target host
2. Select Patch Management

> Patch Management

Create a datasource

Create and configure a new datasource. Add the datasource to a server profile to make it available to deployed applications.

1. Select the target profile
2. Add a new datasource and follow the steps in the Create Datasource wizard

> Datasources

Create a server group

Server groups allow you to apply common management policies and deploy applications to a set of servers at once.

1. Add a new group
2. Assign it a profile and socket binding

> Create Server Group

Assign user roles

Assign roles to individual users or user groups to secure access to system resources.

1. Add user or group.
2. Enter user or group name and realm
3. Assign one or more roles to that user or group

> Role Assignment

Find More Resources

General Resources

- [Read JBoss EAP Documentation](#)
- [Learn more about Red Hat JBoss](#)
- [Get help or submit a trouble ticket](#)
- [Sign up for Red Hat training courses](#)

Developer Resources

- [Access tutorials and quickstarts](#)
- [Ask questions to the development community](#)

Operational Resources

- [View supported configurations for JBoss EAP](#)
- [Search the Red Hat knowledgebase for solutions to issues you encounter](#)
- [Reach out to Red Hat consulting for further help from experts](#)

2.5.5.Final-redhat-1 Tools Settings

Figure 3.4. Log in screen of the Management Console

- Uncheck the **Enable Usage Data Collection** option on the **Settings** dialog to remove the selection. Click **Save** button. Confirm the application reload to activate the new settings.

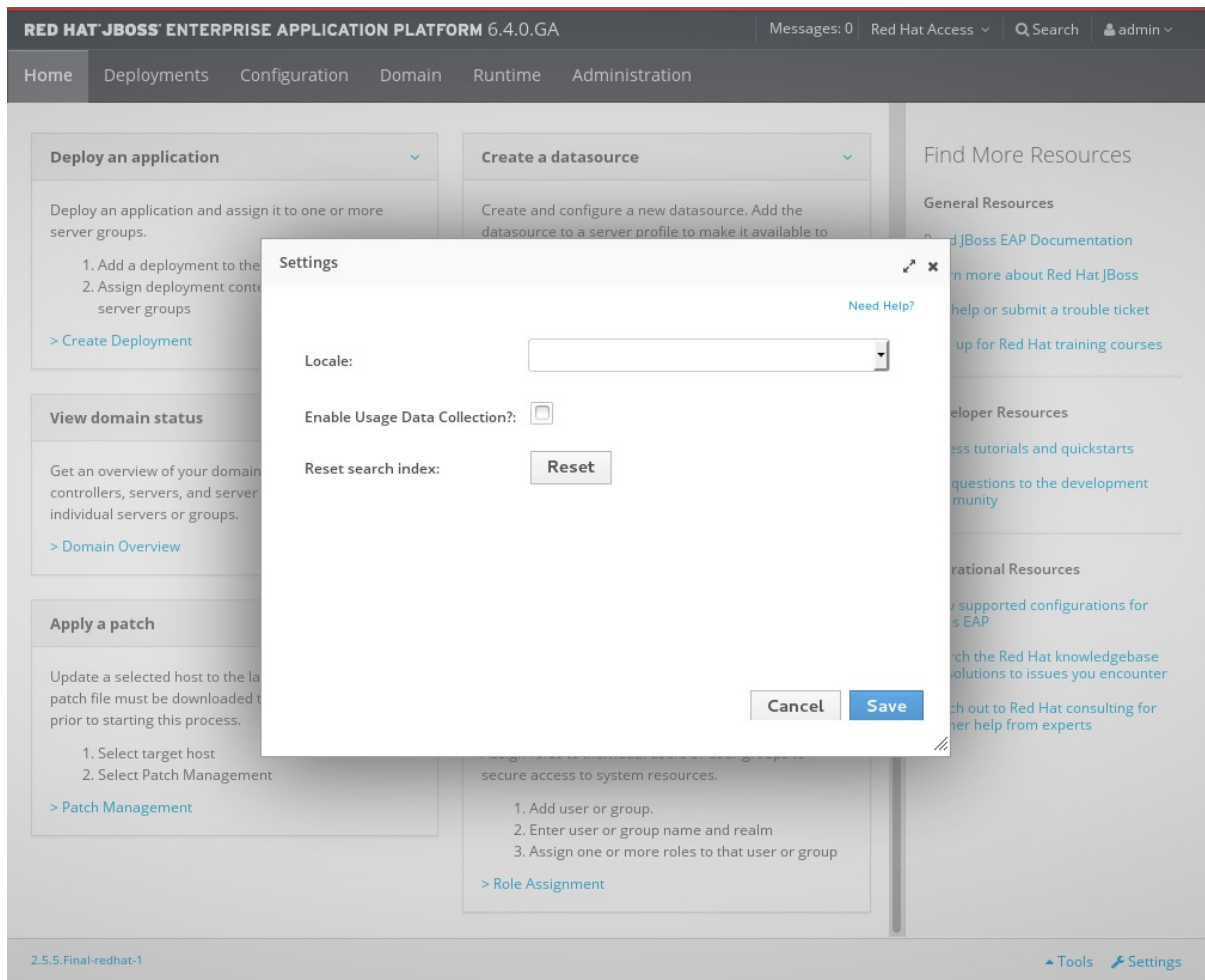


Figure 3.5. Settings dialog (Disable Usage Data Collection)

[Report a bug](#)

3.3.7. Configure a Server Using the Management Console

Prerequisites

- [Section 2.2.4, "Start JBoss EAP 6 as a Managed Domain"](#)
- [Section 3.3.2, "Log in to the Management Console"](#)

Procedure 3.2. Configure the Server

1. Select the **Domain** tab from the top of the console. Available server instances will be displayed in a table.
2. Click **Server Configurations**.

The **Server Configurations** panel for the relevant host appears.

3. Select the server instance from the **Available Server Configurations** table.
4. Click **Edit** above the details of the chosen server.
5. Make changes to the configuration attributes.

6. Click **Save** to finish.

The screenshot shows the Management Console interface for Red Hat JBoss Enterprise Application Platform 6.4.0.GA. The 'Domain' tab is selected, and the 'Server Configurations' section is active. The 'Available Server Configurations' table is as follows:

Configuration Name	Server Group	Start Mode	Running?
server-one	main-server-group	auto	✓
server-three	other-server-group	on-demand	
server-two	main-server-group	auto	✓

Below the table, the 'Attributes' tab is selected, showing the configuration details for 'server-one':

- Name: server-one
- Auto Start?: true
- Server Group: main-server-group

Figure 3.6. Server configuration

Result

The server configuration is changed, and will take effect next time the server restarts.

[Report a bug](#)

3.3.8. Add a Deployment in the Management Console

Prerequisites

- [Section 3.3.2, “Log in to the Management Console”](#)
 1. Select the **Deployments** tab at the top of the console.
 2. Select **Add** on the **Content Repository** tab. A **Create Deployment** dialog box appears.

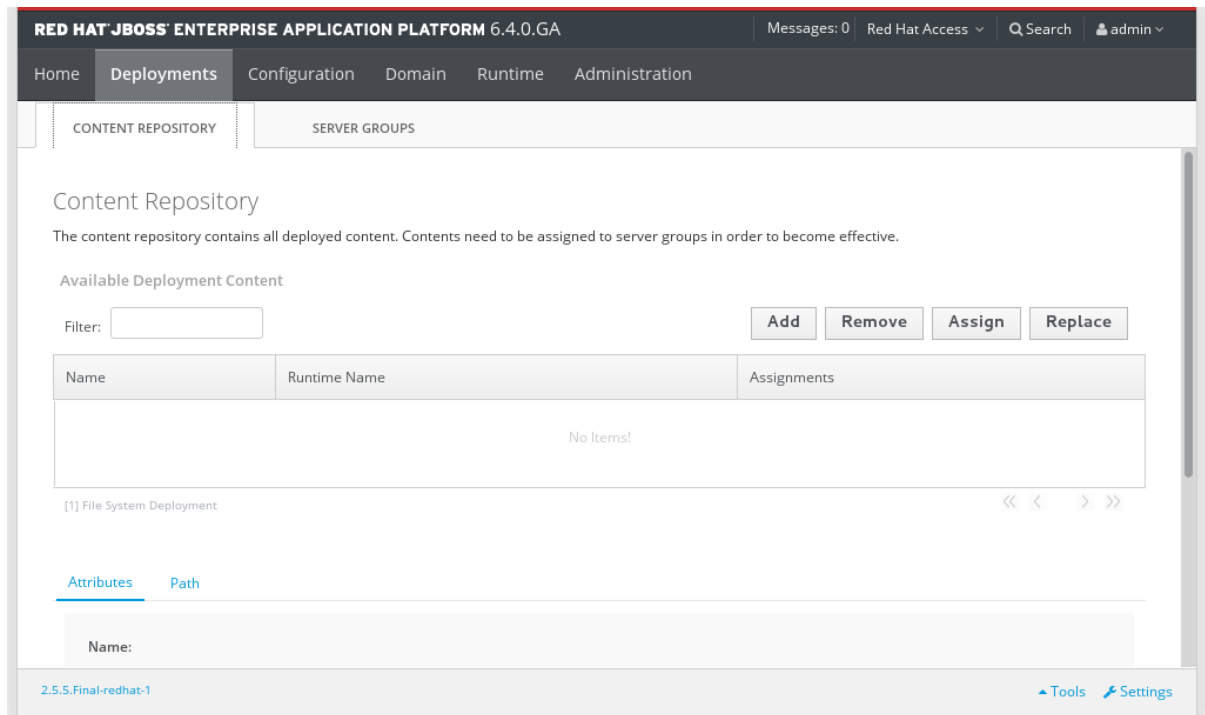


Figure 3.7. Manage standalone deployments

3. In the dialog box, click **Browse**. Browse to the file you want to deploy, select it and upload it. Click **Next** to proceed.

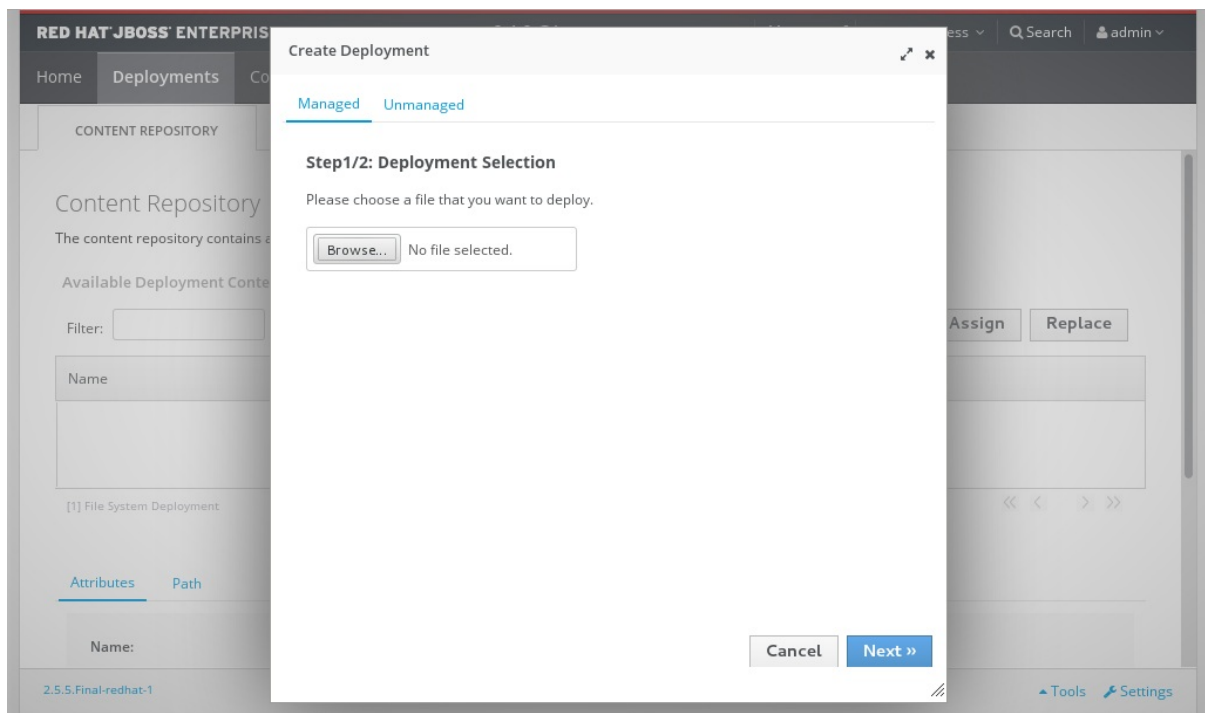


Figure 3.8. Deployment selection

4. Verify the deployment name and runtime name that appear in the **Create Deployments** dialog box. Click **Save** to upload the file once the names are verified.

Result

The selected content is uploaded to the server and is now ready for deployment.

[Report a bug](#)

3.3.9. Create a New Server in the Management Console

Prerequisites

- [Section 2.2.4, “Start JBoss EAP 6 as a Managed Domain”](#)
- [Section 3.3.2, “Log in to the Management Console”](#)

Procedure 3.3. Create a New Server Configuration

1. **Navigate to the Server Configurations page in the Management Console**
Select the **Domain** tab from the top of the console.

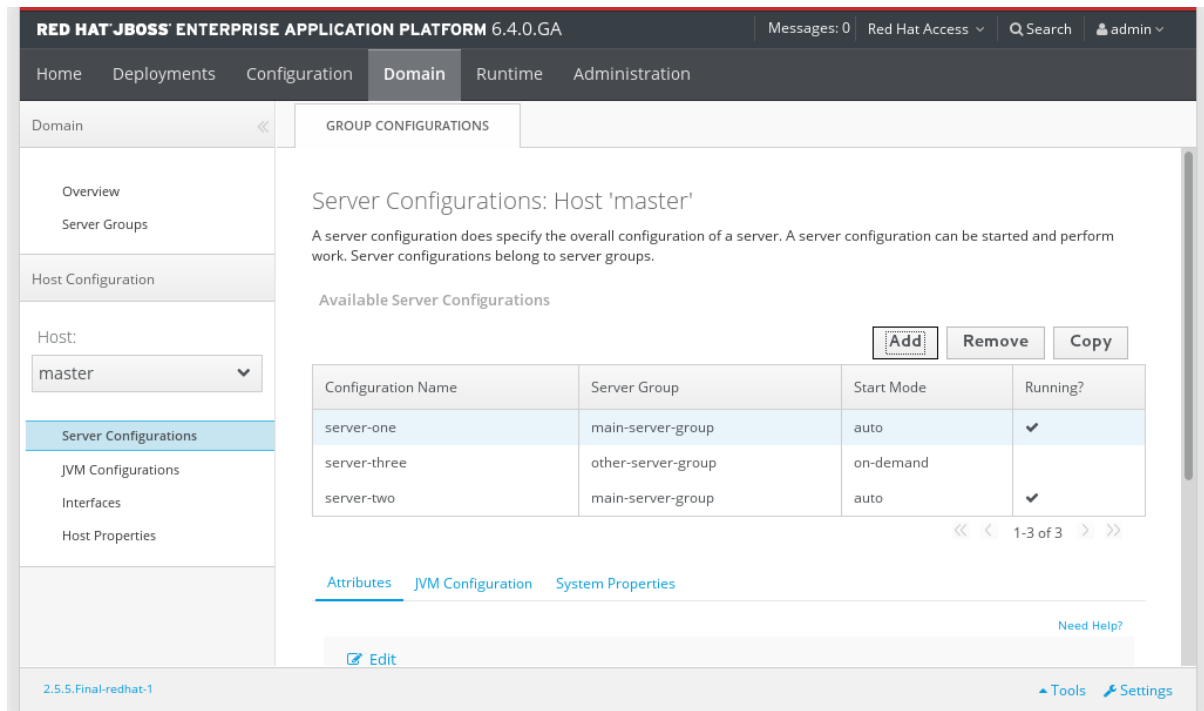


Figure 3.9. Server Configuration

2. Click **Server Configurations** in the left menu.
3. **Create a new configuration**
 - a. Select the **Add** button above the **Available Server Configurations** table.
 - b. Enter the basic server settings in the **Create Server Configuration** dialog.
 - c. Select the **Save** button to save the new Server Configuration.

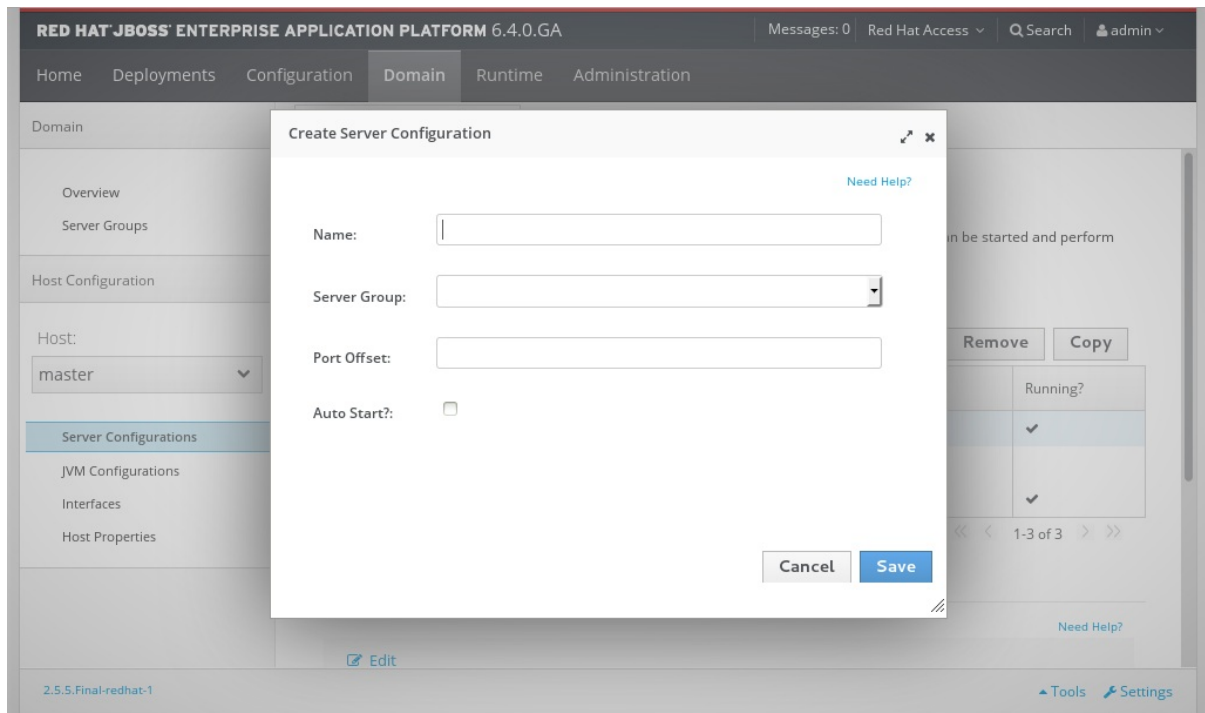


Figure 3.10. Create a new configuration

[Report a bug](#)

3.3.10. Change the Default Log Levels Using the Management Console

Procedure 3.4. Edit the Logging Levels

1. **Navigate to the Logging panel in the Management Console**
 - a. If you are working with a managed domain, select the **Configuration** tab at the top of the console, then select the relevant profile from the drop-down list on the left of the console.
 - b. For either a managed domain or a standalone server, expand the **Core** menu from the list on the left of the console and click the **Logging** entry.
 - c. Click on the **Log Categories** tab in the top of the console.

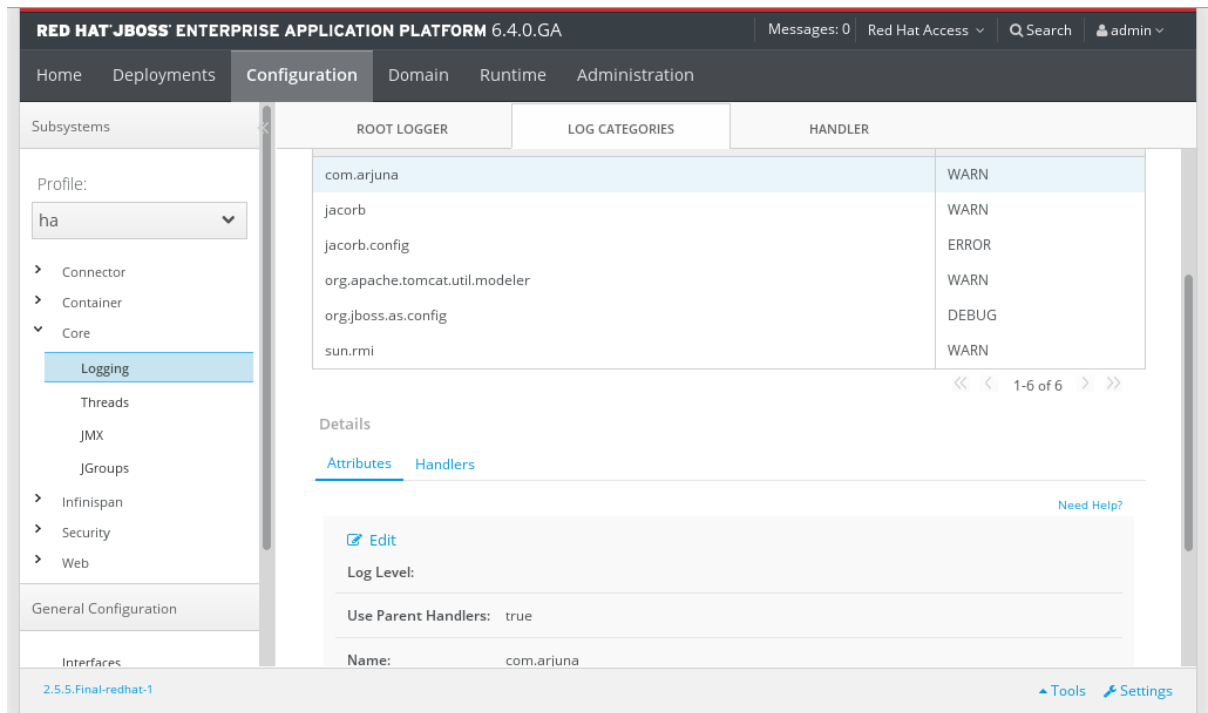


Figure 3.11. Logging panel

2. Edit logger details

Edit the details for any of the entries in the **Log Categories** table.

- Select an entry in the **Log Categories** table, then click **Edit** in the **Details** section below.
- Set the log level for the category with the **Log Level** drop-down box. Click the **Save** button when done.

Result

The log levels for the relevant categories are now updated.

[Report a bug](#)

3.3.11. Create a New Server Group in the Management Console

Prerequisites

- [Section 3.3.2, "Log in to the Management Console"](#)

Procedure 3.5. Configure and Add a new Server Group

- Navigate to the Server Groups view**
Select the **Domain** tab from the top of the console.
- Select **Server Groups** in the left hand column.

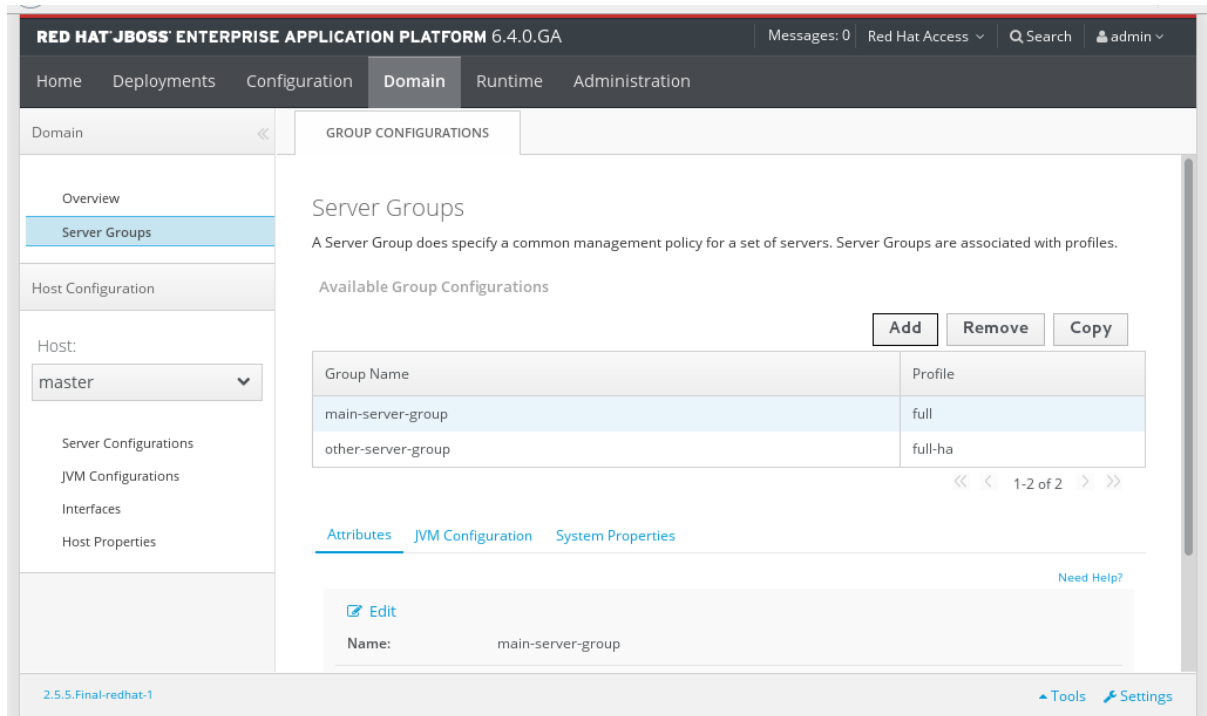


Figure 3.12. The Server Groups view

3. **Add a server group**
Click the **Add** button to add a new server group.
4. **Configure the server group**
 - a. Enter a name for the server group.
 - b. Select the profile for the server group.
 - c. Select the socket binding for the server group.
 - d. Click the **Save** button to save your new group.

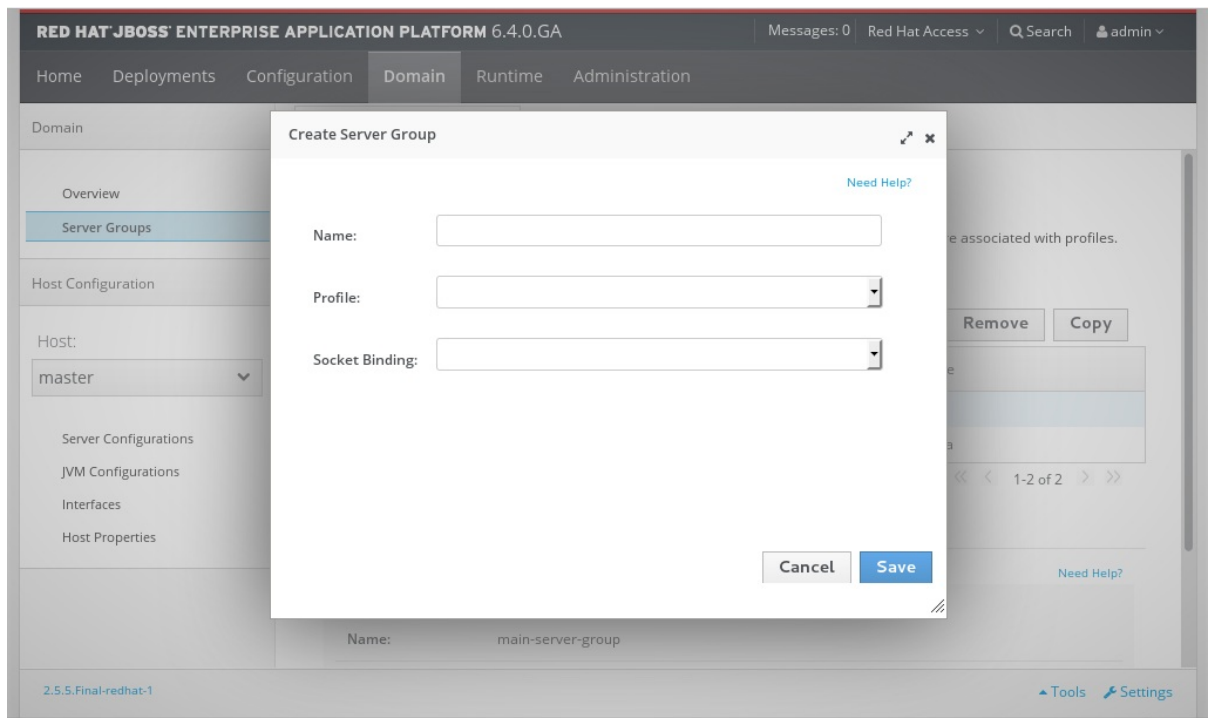


Figure 3.13. The Create Server Group dialog

Result

The new server group is visible in the Management Console.

[Report a bug](#)

3.3.12. Viewing Logs in the Management Console

You can view server and application logs in the JBoss EAP 6 Management Console in order to help diagnose errors, performance problems, and other issues. For a log to be viewable in the Management Console Log Viewer, it must be located in the server's **jboss.server.log.dir** directory. The JBoss EAP 6 Log Viewer also respects user RBAC role assignments, so a user logged in to the Management Console can only view logs that they are authorized to access.

Prerequisites

- [Section 3.3.2, “Log in to the Management Console”](#)

Procedure 3.6. View JBoss EAP 6 Logs in the Management Console

1. Select the **Runtime** tab from the top of the Management Console.
 - a. If you are using a Managed Domain, use the **Change Server** button on the left menu to select the JBoss EAP 6 server that you want to view the logs of.
2. Expand the **Platform** menu on the left, and select **Log Viewer**.
3. Select a log file from the list, and click the **View** button.

You can also click **Download** to download the log file to your local machine.



NOTE

The Management Console Log Viewer displays a confirmation if you attempt to open a log file that is larger than 15MB.

The Management Console Log Viewer is not intended to be a text editor replacement for viewing very large log files (>100MB). Opening very large log files in the Management Console Log Viewer could crash your web browser, so you should always download large log files separately and open them in a text editor.

4. The selected log will open as a new tab within the Management Console. You can open multiple log files in other tabs by returning to the **LOG FILES** tab and repeating the previous step.

[Report a bug](#)

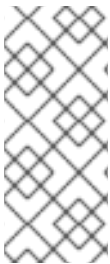
3.3.13. Customer Portal Integration in the Management Console

You can use the access.redhat.com interface to browse sections of the Red Hat Customer Portal without leaving the Management Console of your JBoss EAP installation.

The top navigation bar of the Management Console contains a drop-down menu: **Red Hat Access**. Clicking on this menu will reveal three task-specific links to the Customer Portal:

- **Search Customer Portal**
- **Open Case**
- **Modify Case**

The features of each of these links are discussed in more detail below.



NOTE

If you are not already logged in to the Customer Portal when you click one of these links, a dialogue box will appear, prompting you to log in. You must be logged into the Customer Portal in the browser session that you are using to access the Management Console. If you are logged in to the Customer Portal in one browser but use a different browser to access the Management Console, you will be prompted to log in.

Search Customer Portal

Clicking on **Search Customer Portal** presents a page containing a search box. You can enter search terms or phrases to find Knowledge Base articles.

Once you have performed a search, you can select an item from the list of results and see the entire article displayed in a separate pane.

Open Case

The **Open Case** page allows you to open a new support case.

You will be presented with a form to complete in order to open a new support case. A list of recommended Knowledge Base articles is provided beside the form. This list refreshes based on the details provided for the support case.

Modify Case

The **Modify Case** page allows you to view and modify existing support cases.

You can refine the results by limiting your search to grouped or ungrouped cases, and by the state of the case (open, closed, or either).

After selecting a specific support case, you can view or update the details of the support case, as well as add comments.

[Report a bug](#)

3.4. THE MANAGEMENT CLI

3.4.1. About the Management Command Line Interface (CLI)

The Management Command Line Interface (CLI) is a command line administration tool for JBoss EAP 6.

Use the Management CLI to start and stop servers, deploy and undeploy applications, configure system settings, and perform other administrative tasks. Operations can be performed in batch mode, allowing multiple tasks to be run as a group.

[Report a bug](#)

3.4.2. Launch the Management CLI

Prerequisites:

- [Section 2.2.2, "Start JBoss EAP 6 as a Standalone Server"](#)
- [Section 2.2.4, "Start JBoss EAP 6 as a Managed Domain"](#)

Procedure 3.7. Launch CLI in Linux or Microsoft Windows Server

- ◦ **Launch the CLI in Linux**

Run the ***EAP_HOME/bin/jboss-cli.sh*** file by entering the following at a command line:

```
$ EAP_HOME/bin/jboss-cli.sh
```

- ◦ **Launch the CLI in Microsoft Windows Server**

Run the ***EAP_HOME\bin\jboss-cli.bat*** file by double-clicking it, or by entering the following at a command line:

```
C:\>EAP_HOME\bin\jboss-cli.bat
```

[Report a bug](#)

3.4.3. Quit the Management CLI

From the Management CLI, enter the **quit** command:

```
[domain@localhost:9999 /] quit
```

[Report a bug](#)

3.4.4. Connect to a Managed Server Instance Using the Management CLI

Prerequisites

- [Section 3.4.2, “Launch the Management CLI”](#)

Procedure 3.8. Connect to a Managed Server Instance

- **Run the `connect` command**

From the Management CLI, enter the **connect** command:

```
[disconnected /] connect
Connected to domain controller at localhost:9999
```

- Alternatively, to connect to a managed server when starting the Management CLI on a Linux system, use the **--connect** parameter:

```
$ EAP_HOME/bin/jboss-cli.sh --connect
```

- The **--connect** parameter can be used to specify the host and port of the server. To connect to the address **192.168.0.1** with the port value **9999** the following would apply:

```
$ EAP_HOME/bin/jboss-cli.sh --connect --controller=192.168.0.1:9999
```

[Report a bug](#)

3.4.5. Obtain Help with the Management CLI

Summary

Sometimes you might need guidance if you need to learn a CLI command or feel unsure about what to do. The Management CLI features a help dialog with general and context-sensitive options. (Note that the help commands dependent on the operation context require an established connection to either a standalone or domain controller. These commands will not appear in the listing unless the connection has been established.)

Prerequisites

- [Section 3.4.2, “Launch the Management CLI”](#)

1. **For general help**

From the Management CLI, enter the **help** command:

```
[standalone@localhost:9999 /] help
```

2. **Obtain context-sensitive help**

From the Management CLI, enter the **help -commands** extended command:

```
[standalone@localhost:9999 /] help --commands
```

3. For a more detailed description of a specific command, enter the command, followed by **--help**.

```
[standalone@localhost:9999 /] deploy --help
```

Result

The CLI help information is displayed.

[Report a bug](#)

3.4.6. Use the Management CLI in Batch Mode

Summary

Batch processing allows a number of operation requests to be grouped in a sequence and executed together as a unit. If any of the operation requests in the sequence fail, the entire group of operations is rolled back.



NOTE

Batch mode does not support conditional statements.

Prerequisites

- [Section 3.4.2, "Launch the Management CLI"](#)
- [Section 3.4.4, "Connect to a Managed Server Instance Using the Management CLI"](#)

Procedure 3.9. Batch Mode Commands and Operations

1. **Enter batch mode**

Enter batch mode with the **batch** command.

```
[standalone@localhost:9999 /] batch
```

Batch mode is indicated by the hash symbol (**#**) in the prompt.

2. **Add operation requests to the batch**

Once in batch mode, enter operation requests as normal. The operation requests are added to the batch in the order they are entered.

Refer to [Section 3.4.8, "Use Operations and Commands in the Management CLI"](#) for details on formatting operation requests.

3. **Run the batch**

Once the entire sequence of operation requests is entered, run the batch with the **run-batch** command.

```
[standalone@localhost:9999 / #] run-batch  
The batch executed successfully.
```

Refer to [Section 3.4.7, "CLI Batch Mode Commands"](#) for a full list of commands available for working with batches.

4. Batch commands stored in external files

Frequently run batch commands can be stored in an external text file and can either be loaded by passing the full path to the file as an argument to the **batch** command or executed directly by being an argument to the **run-batch** command.

You can create a batch command file using a text editor. Each command must be on a line by itself and the CLI should be able to access it.

The following command will load a **myscript.txt** file in the batch mode. All commands in this file will now be accessible to be edited or removed. New commands can be inserted. Changes made in this batch session do not persist to the **myscript.txt** file.

```
[standalone@localhost:9999 /] batch --file=myscript.txt
```

The following will instantly run the batch commands stored in the file **myscript.txt**

```
[standalone@localhost:9999 /] run-batch --file=myscript.txt
```

Result

The entered sequence of operation requests is completed as a batch.

[Report a bug](#)

3.4.7. CLI Batch Mode Commands

This table provides a list of valid batch commands that can be used in the JBoss EAP 6 CLI. These commands can only be used to work with batches.

Table 3.2. CLI Batch Mode Commands

Command Name	Description
list-batch	List of the commands and operations in the current batch.
edit-batch-line line-number edited-command	Edit a line in the current batch by providing the line number to edit and the edited command. Example: edit-batch-line 2 data-source disable --name=ExampleDS .
move-batch-line fromline toline	Re-order the lines in the batch by specifying the line number you want to move as the first argument and its new position as the second argument. Example: move-batch-line 3 1 .
remove-batch-line linenummer	Remove the batch command at the specified line. Example: remove-batch-line 3 .

Command Name	Description
holdback-batch [batchname]	<p>You can postpone or store a current batch by using this command. Use this if you want to suddenly execute something in the CLI outside the batch. To return to this heldback batch, simply type batch again at the CLI command line.</p> <p>If you provide a batchname while using holdback-batch command the batch will be stored under that name. To return to the named batch, use the command batch batchname. Calling the batch command without a batchname will start a new (unnamed) batch. There can be only one unnamed heldback batch.</p> <p>To see a list of all heldback batches, use the batch -l command.</p>
discard-batch	Dicards the currently active batch.

[Report a bug](#)

3.4.8. Use Operations and Commands in the Management CLI

Prerequisites

- [Section 3.4.2, "Launch the Management CLI"](#)
- [Section 3.4.4, "Connect to a Managed Server Instance Using the Management CLI"](#)

Procedure 3.10. Create, Configure and Execute Requests

1. Construct the operation request

Operation requests allow for low-level interaction with the management model. They provide a controlled way to edit server configurations. An operation request consists of three parts:

- an *address*, prefixed with a slash (/).
- an *operation name*, prefixed with a colon (:).
- an optional set of *parameters*, contained within parentheses (()).

a. Determine the address

The configuration is presented as a hierarchical tree of addressable resources. Each resource node offers a different set of operations. The address specifies which resource node to perform the operation on. An address uses the following syntax:

```
/node-type=node-name
```

- *node-type* is the resource node type. This maps to an element name in the configuration XML.

- *node-name* is the resource node name. This maps to the **name** attribute of the element in the configuration XML.
- Separate each level of the resource tree with a slash (/).

Refer to the configuration XML files to determine the required address. The ***EAP_HOME/standalone/configuration/standalone.xml*** file holds the configuration for a standalone server and the ***EAP_HOME/domain/configuration/domain.xml*** and ***EAP_HOME/domain/configuration/host.xml*** files hold the configuration for a managed domain.



NOTE

Running the CLI commands in Domain Mode requires host and server specification. For example, ***/host=master/server=server-one/subsystem=logging***

Example 3.5. Example operation addresses

To perform an operation on the logging subsystem, use the following address in an operation request:

```
/subsystem=logging
```

To perform an operation on the Java datasource, use the following address in an operation request:

```
/subsystem=datasources/data-source=java
```

b. Determine the operation

Operations differ for each different type of resource node. An operation uses the following syntax:

```
:operation-name
```

- *operation-name* is the name of the operation to request.

Use the **read-operation-names** operation on any resource address in a standalone server to list the available operations.

Example 3.6. Available operations

To list all available operations for the logging subsystem, enter the following request for a standalone server:

```
[standalone@localhost:9999 /] /subsystem=logging:read-operation-names
{
  "outcome" => "success",
  "result" => [
    "add",
    "read-attribute",
    "read-children-names",
```

```

    "read-children-resources",
    "read-children-types",
    "read-operation-description",
    "read-operation-names",
    "read-resource",
    "read-resource-description",
    "remove",
    "undefine-attribute",
    "whoami",
    "write-attribute"
  ]
}

```

c. Determine any parameters

Each operation may require different parameters.

Parameters use the following syntax:

```
(parameter-name=parameter-value)
```

- *parameter-name* is the name of the parameter.
- *parameter-value* is the value of the parameter.
- Multiple parameters are separated by commas (,).

To determine any required parameters, perform the **read-operation-description** command on a resource node, passing the operation name as a parameter. Refer to [Example 3.7, "Determine operation parameters"](#) for details.

Example 3.7. Determine operation parameters

To determine any required parameters for the **read-children-types** operation on the logging subsystem, enter the **read-operation-description** command as follows:

```

[standalone@localhost:9999 /] /subsystem=logging:read-operation-
description(name=read-children-types)
{
  "outcome" => "success",
  "result" => {
    "operation-name" => "read-children-types",
    "description" => "Gets the type names of all the children under the selected
resource",
    "reply-properties" => {
      "type" => LIST,
      "description" => "The children types",
      "value-type" => STRING
    },
    "read-only" => true
  }
}

```


2. Enter the full operation request

Once the address, operation, and any parameters have been determined, enter the full operation request.

Example 3.8. Example operation request

```
[standalone@localhost:9999 /] /subsystem=web/connector=http:read-
resource(recursive=true)
```

Result

The management interface performs the operation request on the server configuration.

[Report a bug](#)

3.4.9. Use if-else Control Flow with the Management CLI

The Management CLI supports **if-else** control flow, which allows you to choose which set of commands and operations to execute based on a condition. The **if** condition is a boolean expression which evaluates the response of the management command or operation specified after the **of** keyword.

Expressions can contain any of the following items:

- Conditional operators (&&, ||)
- Comparison operators (>, >=, <, <=, ==, !=)
- Parentheses to group and prioritize expressions

Example 3.9. Using an if statement with Management CLI commands

This example attempts to read the system property **test**. If **outcome** is not **success** (meaning that the property does not exist), then the system property will be added and set to **true**.

```
if (outcome != success) of /system-property=test:read-resource
  /system-property=test:add(value=true)
end-if
```

The condition above uses **outcome**, which is returned when the CLI command after the **of** keyword is executed, as shown below:

```
[standalone@localhost:9999 /] /system-property=test:read-resource
{
  "outcome" => "failed",
  "failure-description" => "JBAS014807: Management resource '[(\"system-property\" => \"test\")]'
not found",
  "rolled-back" => true
}
```

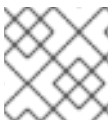
Example 3.10. Using an if-else statement with Management CLI commands

This example checks the launch type of the server process (**STANDALONE** or **DOMAIN**) and issues the appropriate CLI command to enable the **ExampleDS** datasource.

```
if (result == STANDALONE) of /:read-attribute(name=launch-type)
  /subsystem=datasources/data-source=ExampleDS:write-attribute(name=enabled, value=true)
else
  /profile=full/subsystem=datasources/data-source=ExampleDS:write-attribute(name=enabled,
value=true)
end-if
```

Management CLI commands with **if-else** control flow can be specified in a file (one per line) and passed to the **jboss-cli.sh** script to be executed non-interactively.

```
EAP_HOME/bin/jboss-cli.sh --connect --file=CLI_FILE
```



NOTE

The use of nested **if-else** statements is not supported.

[Report a bug](#)

3.4.10. Management CLI Configuration Options

The Management CLI configuration file - **jboss-cli.xml** - is loaded each time the CLI is started. It must be located either in the directory ***\$EAP_HOME*/bin** or a directory specified in the system property **jboss.cli.config**.

default-controller

Configuration of the controller to which to connect if the **connect** command is executed without any parameters.

default-controller Parameters

host

Hostname of the controller. Default: **localhost**.

port

Port number on which to connect to the controller. Default: 9999.

validate-operation-requests

Indicates whether the parameter list of the operation requests is to be validated before the requests are sent to the controller for execution. Type: Boolean. Default: **true**.

history

This element contains the configuration for the commands and operations history log.

history Parameters

enabled

Indicates whether or not the **history** is enabled. Type: Boolean. Default: **true**.

file-name

Name of the file in which the history is to be stored. Default = **.jboss-cli-history**.

file-dir

Directory in which the history is to be stored. Default = **\$USER_HOME**

max-size

Maximum size of the history file. Default: 500.

resolve-parameter-values

Whether to resolve system properties specified as command argument (or operation parameter) values before sending the operation request to the controller or let the resolution happen on the server side. Type: Boolean. Default = **false**.

connection-timeout

The time allowed to establish a connection with the controller. Type: Integer. Default: 5000 seconds.

ssl

This element contains the configuration for the Key and Trust stores used for SSL.



WARNING

Red Hat recommends that you explicitly disable SSL in favor of TLSv1.1 or TLSv1.2 in all affected packages.

ssl Parameters

vault

Type: **vaultType**

key-store

Type: string.

key-store-password

Type: string.

alias

Type: string

key-password

Type: string

trust-store

Type: string.

trust-store-password

Type: string.

modify-trust-store

If set to **true**, the CLI will prompt the user when unrecognised certificates are received and allow them to be stored in the truststore. Type: Boolean. Default: **true**.

vaultType

If neither **code** nor **module** are specified, the default implementation will be used. If **code** is specified but not **module**, it will look for the specified class in the Picketbox module. If **module and code** are specified, it will look for the class specified by **code** in the module specified by 'module'.

code

Type: String.

module

Type: String

silent

Specifies if informational and error messages are to be output to the terminal. Even if the **false** is specified, the messages will still be logged using the logger if its configuration allows and/or if the output target was specified as part of the command line using >. Default: **False**.

[Report a bug](#)

3.4.11. Reference of Management CLI Commands

Prerequisites

- [Section 3.4.2, "Launch the Management CLI"](#)

Summary

The topic [Section 3.4.5, "Obtain Help with the Management CLI"](#) describes how to access the Management CLI help features, including a help dialogue with general and context sensitive options. The help commands are dependent on the operation context and require an established connection to either a standalone or domain controller. These commands will not appear in the listing unless the connection has been established.

Table 3.3.

Command	Description
---------	-------------

Command	Description
batch	Starts the batch mode by creating a new batch or, depending on the existing held back batches, re-activates one. If there are no held back batches this command, when invoked without arguments, will start a new batch. If there is an unnamed held back batch, this command will re-activate it. If there are named held back batches, they can be activated by executing this command with the name of the held back batch as the argument.
cd	Changes the current node path to the argument. The current node path is used as the address for operation requests that do not contain the address part. If an operation request does include the address, the included address is considered relative to the current node path. The current node path may end on a node-type. In that case, to execute an operation specifying a node-name would be sufficient, such as logging:read-resource.
clear	Clears the screen.
command	Allows you to add new, remove and list existing generic type commands. A generic type command is a command that is assigned to a specific node type and which allows you to perform any operation available for an instance of that type. It can also modify any of the properties exposed by the type on any existing instance.
connect	Connects to the controller on the specified host and port.
connection-factory	Defines a connection factory.
data-source	Manages JDBC datasource configurations in the datasource subsystem.
deploy	Deploys the application designated by the file path or enables an application that is pre-existing but disabled in the repository. If executed without arguments, this command will list all the existing deployments.
echo	Available from JBoss EAP 6.4, the echo command outputs to the console the specified text. The text is output verbatim so the use of variables is not available. Example: <pre> echo Phase one complete </pre>
help	Displays the help message. Can be used with the --commands argument to provide context sensitive results for the given commands.
history	Displays the CLI command history in memory and displays a status of whether the history expansion is enabled or disabled. Can be used with arguments to clear, disable and enable the history expansion as required.

Command	Description
jms-queue	Defines a JMS queue in the messaging subsystem.
jms-topic	Defines a JMS topic in the messaging subsystem.
ls	List the contents of the node path. By default the result is printed in columns using the whole width of the terminal. Using the -l switch will print results on one name per line.
pwd	Prints the full node path of the current working node.
quit	Terminates the command line interface.
read-attribute	Prints the value and, depending on the arguments, the description of the attribute of a managed resource.
read-operation	Displays the description of a specified operation, or lists all available operations if none is specified.
undeploy	Undeploys an application when run with the name of the intended application. Can be run with arguments to remove the application from the repository also. Prints the list of all existing deployments when executed without an application specified.
version	Prints the application server version and environment information.
xa-data-source	Manages JDBC XA datasource configuration in the datasource subsystem.

[Report a bug](#)

3.4.12. Reference of Management CLI Operations

Exposing operations in the Management CLI

Operations in the Management CLI can be exposed by using the **read-operation-names** operation described in the topic [Section 3.5.5, "Display the Operation Names using the Management CLI"](#) . The operation descriptions can be exposed by using the **read-operation-descriptions** operation described in the topic [Section 3.5.4, "Display an Operation Description using the Management CLI"](#) .

Table 3.4. Management CLI operations

Operation Name	Description
add-namespace	Adds a namespace prefix mapping to the namespaces attribute's map.
add-schema-location	Adds a schema location mapping to the schema-locations attribute's map.

Operation Name	Description
delete-snapshot	Deletes a snapshot of the server configuration from the snapshots directory.
full-replace-deployment	Add previously uploaded deployment content to the list of content available for use, replace existing content of the same name in the runtime, and remove the replaced content from the list of content available for use. Refer to link for further information.
list-snapshots	Lists the snapshots of the server configuration saved in the snapshots directory.
read-attribute	Displays the value of an attribute for the selected resource.
read-children-names	Displays the names of all children under the selected resource with the given type.
read-children-resources	Displays information about all of a resource's children that are of a given type.
read-children-types	Displays the type names of all the children under the selected resource.
read-config-as-xml	Reads the current configuration and displays it in XML format.
read-operation-description	Displays the details of an operation on the given resource.
read-operation-names	Displays the names of all the operations for the given resource.
read-resource	Displays a model resource's attribute values along with either basic or complete information about any child resources.
read-resource-description	Displays the description of a resource's attributes, types of children and operations.
reload	Reloads the server by shutting all services down and restarting.
remove-namespace	Removes a namespace prefix mapping from the namespaces attribute map.
remove-schema-location	Removes a schema location mapping from the schema-locations attribute map.
replace-deployment	Replace existing content in the runtime with new content. The new content must have been previously uploaded to the deployment content repository.

Operation Name	Description
resolve-expression	Operation that accepts an expression as input or a string that can be parsed into an expression, and resolves it against the local system properties and environment variables.
resolve-internet-address	Takes a set of interface resolution criteria and finds an IP address on the local machine that matches the criteria, or fails if no matching IP address can be found.
server-set-restart-required	Puts the server into a restart-required mode
shutdown	Shuts down the server via a call to System.exit(0) .
start-servers	Starts all configured servers in a Managed Domain that are not currently running.
stop-servers	Stops all servers currently running in a Managed Domain.
take-snapshot	Takes a snapshot of the server configuration and saves it to the snapshots directory.
upload-deployment-bytes	Indicates that the deployment content in the included byte array should be added to the deployment content repository. Note that this operation does not indicate the content should be deployed into the runtime.
upload-deployment-stream	Indicates that the deployment content available at the included input stream index should be added to the deployment content repository. Note that this operation does not indicate the content should be deployed into the runtime.
upload-deployment-url	Indicates that the deployment content available at the included URL should be added to the deployment content repository. Note that this operation does not indicate the content should be deployed into the runtime.
validate-address	Validates the operation's address.
write-attribute	Sets the value of an attribute for the selected resource.

[Report a bug](#)

3.4.13. Property Substitution in the Management CLI

JBoss EAP 6 supports the use of preset element and property expressions in the Management Command Line Interface. These expressions will be resolved to their defined values during the execution of the command.

The following properties can be substituted with expressions:

- the operation address part of the operation request (as node types and/or names);
- operation name;
- operation parameter names;
- header names and values;
- command names;
- command argument names.

By default, the CLI performs property substitution for every line except for argument or parameter values. Argument and parameter values are resolved in the server at runtime. If you require property substitution for argument or parameter values to occur in the Management CLI client and have it send the resolved values to the server, complete the following procedure.

Procedure 3.11. Enable Property Substitution in the Management CLI

1. Open the file **EAP_HOME/bin/jboss-cli.xml**.
2. Locate the ***resolve-parameter-values*** parameter and change the value to **true** (the default is **false**).

```
<!-- whether to resolve system properties specified as command argument or operation
parameter values in the Management CLI VM before sending the operation requests to the
controller -->
<resolve-parameter-values>true</resolve-parameter-values>
```

This element only affects operation request parameter values and command argument values. It does not impact the rest of the command line. This means system properties present on the command line will be resolved during the parsing of the line regardless of what the value of ***resolve-parameter-values*** element is, unless it is a parameter/argument value.

Refer to [Section 3.4.10, “Management CLI Configuration Options”](#) for other Management CLI configuration options.

Be aware that system values used in Management CLI commands must have already been defined. You must include the **--properties=/path/to/file.properties** argument or one or more **-Dkey=VALUE** parameters, when starting your Management CLI instance. The properties file uses a standard *key=value* syntax.

Property keys are denoted in your Management CLI commands using the syntax **#{MY_VAR}**.

Example 3.11. Example: Using properties in Management CLI commands

```
/subsystem=datasources/data-source=#{datasourcename}:add(connection-
url=jdbc:oracle:thin:@server:1521:ora1, jndi-name=java:/jboss/#{name}, driver-
name=#{drivername})
```

[Report a bug](#)

3.5. MANAGEMENT CLI OPERATIONS

3.5.1. Display the Attributes of a Resource with the Management CLI

Prerequisites

- [Section 3.4.2, "Launch the Management CLI"](#)

Summary

The **read-attribute** operation is a global operation used to read the current runtime value of a selected attribute. It can be used to expose only the values that have been set by the user, ignoring any default or undefined values. The request properties include the following parameters.

Request Properties

name

The name of the attribute to get the value for under the selected resource.

include-defaults

A Boolean parameter that can be set to **false** to restrict the operation results to only show attributes set by the user and ignore default values.

Procedure 3.12. Display the Current Runtime Value of a Selected Attribute

- **Run the `read-attribute` operation**
From the Management CLI, use the **read-attribute** operation to display the value of a resource attribute. For more details on operation requests, refer to the topic [Section 3.4.8, "Use Operations and Commands in the Management CLI"](#).

```
[standalone@localhost:9999 /]:read-attribute(name=name-of-attribute)
```

An advantage of the **read-attribute** operation is the ability to expose the current runtime value of a specific attribute. Similar results can be achieved with the **read-resource** operation, but only with the addition of the **include-runtime** request property, and only as part of a list of all available resources for that node. The **read-attribute** operation is intended for fine-grained attribute queries, as the following example shows.

Example 3.12. Run the `read-attribute` operation to expose the public interface IP

If you know the name of the attribute that you would like to expose, you can use the **read-attribute** to return the exact value in the current runtime.

```
[standalone@localhost:9999 /] /interface=public:read-attribute(name=resolved-address)
{
  "outcome" => "success",
  "result" => "127.0.0.1"
}
```

The **resolved-address** attribute is a runtime value, so it is not displayed in the results of the standard **read-resource** operation.

```
[standalone@localhost:9999 /] /interface=public:read-resource
{
  "outcome" => "success",
```

```

"result" => {
  "any" => undefined,
  "any-address" => undefined,
  "any-ipv4-address" => undefined,
  "any-ipv6-address" => undefined,
  "inet-address" => expression "${jboss.bind.address:127.0.0.1}",
  "link-local-address" => undefined,
  "loopback" => undefined,
  "loopback-address" => undefined,
  "multicast" => undefined,
  "name" => "public",
  "nic" => undefined,
  "nic-match" => undefined,
  "not" => undefined,
  "point-to-point" => undefined,
  "public-address" => undefined,
  "site-local-address" => undefined,
  "subnet-match" => undefined,
  "up" => undefined,
  "virtual" => undefined
}
}

```

To display **resolved-address** and other runtime values, you must use the **include-runtime** request property.

```

[standalone@localhost:9999 /] /interface=public:read-resource(include-runtime=true)
{
  "outcome" => "success",
  "result" => {
    "any" => undefined,
    "any-address" => undefined,
    "any-ipv4-address" => undefined,
    "any-ipv6-address" => undefined,
    "inet-address" => expression "${jboss.bind.address:127.0.0.1}",
    "link-local-address" => undefined,
    "loopback" => undefined,
    "loopback-address" => undefined,
    "multicast" => undefined,
    "name" => "public",
    "nic" => undefined,
    "nic-match" => undefined,
    "not" => undefined,
    "point-to-point" => undefined,
    "public-address" => undefined,
    "resolved-address" => "127.0.0.1",
    "site-local-address" => undefined,
    "subnet-match" => undefined,
    "up" => undefined,
    "virtual" => undefined
  }
}

```

Result

The current runtime attribute value is displayed.

[Report a bug](#)

3.5.2. Display the Active User in the Management CLI

Prerequisites

- [Section 3.4.2, "Launch the Management CLI"](#)

Summary

The **whoami** operation is a global operation used to identify the attributes of the current active user. The operation exposes the identity of the username and the realm that they are assigned to. The **whoami** operation is useful for administrators managing multiple users accounts across multiple realms, or to assist in keeping track of active users across domain instances with multiple terminal session and users accounts.

Procedure 3.13. Display the Active User in the Management CLI Using the **whoami** Operation

- **Run the **whoami** operation**
From the Management CLI, use the **whoami** operation to display the active user account.

```
[standalone@localhost:9999 /] :whoami
```

The following example uses the **whoami** operation in a standalone server instance to show that the active user is *username*, and that the user is assigned to the **ManagementRealm** realm.

Example 3.13. Use the **whoami** in a standalone instance

```
[standalone@localhost:9999 /]:whoami
{
  "outcome" => "success",
  "result" => {"identity" => {
    "username" => "username",
    "realm" => "ManagementRealm"
  }}
}
```

Result

Your current active user account is displayed.

[Report a bug](#)

3.5.3. Display System and Server Information in the Management CLI

Prerequisites

- [Section 3.4.2, "Launch the Management CLI"](#)

Procedure 3.14. Display System and Server Information in the Management CLI

- **Run the `version` command**

From the Management CLI, enter the **version** command:

```
[domain@localhost:9999 /] version
```

Result

Your application server version and environment information is displayed.

[Report a bug](#)

3.5.4. Display an Operation Description using the Management CLI

Prerequisites

- [Section 3.4.2, "Launch the Management CLI"](#)

Procedure 3.15. Execute the Command in Management CLI

- **Run the `read-operation-description` operation**

From the Management CLI, use **read-operation-description** to display information about the operation. The operation requires additional parameters in the format of a key-value pair to indicate which operation to display. For more details on operation requests, refer to the topic [Section 3.4.8, "Use Operations and Commands in the Management CLI"](#).

```
[standalone@localhost:9999 /]:read-operation-description(name=name-of-operation)
```

Example 3.14. Display the `list-snapshots` operation description

The following example shows the method for describing the **list-snapshots** operation.

```
[standalone@localhost:9999 /] :read-operation-description(name=list-snapshots)
{
  "outcome" => "success",
  "result" => {
    "operation-name" => "list-snapshots",
    "description" => "Lists the snapshots",
    "request-properties" => {},
    "reply-properties" => {
      "type" => OBJECT,
      "value-type" => {
        "directory" => {
          "type" => STRING,
          "description" => "The directory where the snapshots are stored",
          "expressions-allowed" => false,
          "required" => true,
          "nillable" => false,
          "min-length" => 1L,
          "max-length" => 2147483647L
        },
        "names" => {
          "type" => LIST,
          "description" => "The names of the snapshots within the snapshots directory",
          "expressions-allowed" => false,
```

```

        "required" => true,
        "nullable" => false,
        "value-type" => STRING
    }
},
"access-constraints" => {"sensitive" => {"snapshots" => {"type" => "core"}}},
"read-only" => false
}
}

```

Result

The description is displayed for the chosen operation.

[Report a bug](#)

3.5.5. Display the Operation Names using the Management CLI

Prerequisites

- [Section 3.4.2, "Launch the Management CLI"](#)

Procedure 3.16. Execute the Command in Management CLI

- **Run the `read-operation-names` operation**

From the Management CLI, use the **`read-operation-names`** operation to display the names of the available operations. For more details on operation requests, refer to the topic [Section 3.4.8, "Use Operations and Commands in the Management CLI"](#).

```
[standalone@localhost:9999 /]:read-operation-names
```

Example 3.15. Display the operation names using the Management CLI

The following example shows the method for describing the **`read-operation-names`** operation.

```

[standalone@localhost:9999 /]:read-operation-names
{
  "outcome" => "success",
  "result" => [
    "add-namespace",
    "add-schema-location",
    "delete-snapshot",
    "full-replace-deployment",
    "list-snapshots",
    "read-attribute",
    "read-children-names",
    "read-children-resources",
    "read-children-types",
    "read-config-as-xml",
    "read-operation-description",
    "read-operation-names",
    "read-resource",

```

```

"read-resource-description",
"reload",
"remove-namespace",
"remove-schema-location",
"replace-deployment",
"resolve-expression",
"resolve-internet-address",
"server-set-restart-required",
"shutdown",
"take-snapshot",
"undefine-attribute",
"upload-deployment-bytes",
"upload-deployment-stream",
"upload-deployment-url",
"validate-address",
"validate-operation",
"whoami",
"write-attribute"
]
}

```

Result

The available operation names are displayed.

[Report a bug](#)

3.5.6. Display Available Resources using the Management CLI

Prerequisites

- [Section 3.4.2, "Launch the Management CLI"](#)

Summary

The **read-resource** operation is a global operation used to read resource values. It can be used to expose either basic or complete information about the resources of the current or child nodes, along with a range of request properties to expand or limit the scope of the operation results. The request properties include the following parameters.

Request Properties**recursive**

Whether to recursively include complete information about child resources.

recursive-depth

The depth to which information about child resources should be included.

proxies

Whether to include remote resources in a recursive query. For example including the host level resources from slave Host Controllers in a query of the Domain Controller.

include-runtime

Whether to include runtime attributes in the response, such as attribute values that do not come from the persistent configuration. This request property is set to false by default.

include-defaults

A boolean request property that serves to enable or disable the reading of default attributes. When set to false only the attributes set by the user are returned, ignoring any that remain undefined.

Execute the Command in Management CLI

Run the read-resource operation

From the Management CLI, use the **read-resource** operation to display the available resources.

```
[standalone@localhost:9999 /]:read-resource
```

The following example shows how you might use the **read-resource** operation on a standalone server instance to expose general resource information. The results resemble the **standalone.xml** configuration file, displaying the system resources, extensions, interfaces and subsystems installed or configured for the server instance. These can be further queried directly.

Example 3.16. Using the read-resource operation at the root level

```
[standalone@localhost:9999 /]:read-resource
{
  "outcome" => "success",
  "result" => {
    "management-major-version" => 1,
    "management-micro-version" => 0,
    "management-minor-version" => 7,
    "name" => "localhost",
    "namespaces" => [],
    "product-name" => "EAP",
    "product-version" => "6.4.0.GA",
    "profile-name" => undefined,
    "release-codename" => "Janus",
    "release-version" => "7.5.0.Final-redhat-17",
    "schema-locations" => [],
    "core-service" => {
      "service-container" => undefined,
      "server-environment" => undefined,
      "module-loading" => undefined,
      "platform-mbean" => undefined,
      "management" => undefined,
      "patching" => undefined
    },
    "deployment" => undefined,
    "deployment-overlay" => undefined,
    "extension" => {
      "org.jboss.as.clustering.infinispan" => undefined,
      "org.jboss.as.connector" => undefined,
      "org.jboss.as.deployment-scanner" => undefined,
      "org.jboss.as.ee" => undefined,
      "org.jboss.as.ejb3" => undefined,
      "org.jboss.as.jaxrs" => undefined,

```



```

"org.jboss.as.jdr" => undefined,
"org.jboss.as.jmx" => undefined,
"org.jboss.as.jpa" => undefined,
"org.jboss.as.jsf" => undefined,
"org.jboss.as.logging" => undefined,
"org.jboss.as.mail" => undefined,
"org.jboss.as.naming" => undefined,
"org.jboss.as.pojo" => undefined,
"org.jboss.as.remoting" => undefined,
"org.jboss.as.sar" => undefined,
"org.jboss.as.security" => undefined,
"org.jboss.as.threads" => undefined,
"org.jboss.as.transactions" => undefined,
"org.jboss.as.web" => undefined,
"org.jboss.as.webservices" => undefined,
"org.jboss.as.weld" => undefined
},
"interface" => {
  "management" => undefined,
  "public" => undefined,
  "unsecure" => undefined
},
"path" => {
  "jboss.server.temp.dir" => undefined,
  "user.home" => undefined,
  "jboss.server.base.dir" => undefined,
  "java.home" => undefined,
  "user.dir" => undefined,
  "jboss.server.data.dir" => undefined,
  "jboss.home.dir" => undefined,
  "jboss.server.log.dir" => undefined,
  "jboss.server.config.dir" => undefined,
  "jboss.controller.temp.dir" => undefined
},
"socket-binding-group" => {"standard-sockets" => undefined},
"subsystem" => {
  "jaxrs" => undefined,
  "jsf" => undefined,
  "jca" => undefined,
  "jmx" => undefined,
  "threads" => undefined,
  "webservices" => undefined,
  "sar" => undefined,
  "remoting" => undefined,
  "infinispan" => undefined,
  "weld" => undefined,
  "ejb3" => undefined,
  "transactions" => undefined,
  "datasources" => undefined,
  "deployment-scanner" => undefined,
  "logging" => undefined,
  "jdr" => undefined,
  "pojo" => undefined,
  "jpa" => undefined,
  "naming" => undefined,
  "ee" => undefined,

```

```

    "mail" => undefined,
    "web" => undefined,
    "resource-adapters" => undefined,
    "security" => undefined
  },
  "system-property" => undefined
}
}

```

Run the **read-resource** operation against a child node

The **read-resource** operation can be run to query child nodes from the root. The structure of the operation first defines the node to expose, and then appends the operation to run against it.

```
[standalone@localhost:9999 /] /subsystem=web/connector=http:read-resource
```

In the following example, specific resource information about a web subsystem component can be exposed by directing the **read-resource** operation towards the specific web subsystem node.

Example 3.17. Expose child node resources from the root node

```

[standalone@localhost:9999 /] /subsystem=web/connector=http:read-resource
{
  "outcome" => "success",
  "result" => {
    "configuration" => undefined,
    "enable-lookups" => false,
    "enabled" => true,
    "executor" => undefined,
    "max-connections" => undefined,
    "max-post-size" => 2097152,
    "max-save-post-size" => 4096,
    "name" => "http",
    "protocol" => "HTTP/1.1",
    "proxy-name" => undefined,
    "proxy-port" => undefined,
    "redirect-port" => 443,
    "scheme" => "http",
    "secure" => false,
    "socket-binding" => "http",
    "ssl" => undefined,
    "virtual-server" => undefined
  }
}

```

The same results are possible by using the **cd** command to navigate into the child nodes and run the **read-resource** operation directly.

Example 3.18. Expose child node resources by changing directories

```
[standalone@localhost:9999 /] cd subsystem=web
```

```
[standalone@localhost:9999 subsystem=web] cd connector=http
```

```
[standalone@localhost:9999 connector=http] :read-resource
{
  "outcome" => "success",
  "result" => {
    "configuration" => undefined,
    "enable-lookups" => false,
    "enabled" => true,
    "executor" => undefined,
    "max-connections" => undefined,
    "max-post-size" => 2097152,
    "max-save-post-size" => 4096,
    "name" => "http",
    "protocol" => "HTTP/1.1",
    "proxy-name" => undefined,
    "proxy-port" => undefined,
    "redirect-port" => 443,
    "scheme" => "http",
    "secure" => false,
    "socket-binding" => "http",
    "ssl" => undefined,
    "virtual-server" => undefined
  }
}
```

Use the recursive parameter to include active values in results

The recursive parameter can be used to expose the values of all attributes, including non-persistent values, those passed at startup, or other attributes otherwise active in the runtime model.

```
[standalone@localhost:9999 /] /interface=public:read-resource(include-runtime=true)
```

Compared to the previous example, the inclusion of the **include-runtime** request property exposes additional active attributes, such as the bytes sent and bytes received by the HTTP connector.

Example 3.19. Expose additional and active values with the **include-runtime** parameter

```
[standalone@localhost:9999 /] /subsystem=web/connector=http:read-resource(include-
runtime=true)
{
  "outcome" => "success",
  "result" => {
    "any" => undefined,
    "any-address" => undefined,
    "any-ipv4-address" => undefined,
    "any-ipv6-address" => undefined,
    "inet-address" => expression "${jboss.bind.address:127.0.0.1}",
    "link-local-address" => undefined,
    "loopback" => undefined,
    "loopback-address" => undefined,
    "multicast" => undefined,
    "name" => "public",
```

```

"nic" => undefined,
"nic-match" => undefined,
"not" => undefined,
"point-to-point" => undefined,
"public-address" => undefined,
"resolved-address" => "127.0.0.1",
"site-local-address" => undefined,
"subnet-match" => undefined,
"up" => undefined,
"virtual" => undefined
}
}

```

[Report a bug](#)

3.5.7. Display Available Resource Descriptions using the Management CLI

Prerequisites

- [Section 3.4.2, “Launch the Management CLI”](#)

Execute the Command in Management CLI

Run the **read-resource-description** operation

From the Management CLI, use the **read-resource-description** operation to read and display the available resources. For more details on operation requests, refer to the topic [Section 3.4.8, “Use Operations and Commands in the Management CLI”](#).

```
[standalone@localhost:9999 /]:read-resource-description
```

Use optional parameters

The **read-resource-description** operation allows the use of the additional parameters.

- Use the **operations** parameter to include descriptions of the resource's operations.

```
[standalone@localhost:9999 /]:read-resource-description(operations=true)
```

- Use the **inherited** parameter to include or exclude descriptions of the resource's inherited operations. The default state is true.

```
[standalone@localhost:9999 /]:read-resource-description(inherited=false)
```

- Use the **recursive** parameter to include recursive descriptions of the child resources.

```
[standalone@localhost:9999 /]:read-resource-description(recursive=true)
```

- Use the **locale** parameter to get the resource description in. If null, the default locale will be used.

```
[standalone@localhost:9999 /]:read-resource-description(locale=true)
```

- Use the **access-control** parameter to get the information about the permissions the current caller has for this resource.

```
[standalone@localhost:9999 /]:read-resource-description(access-control=none)
```

Result

Descriptions of the available resources are displayed.

[Report a bug](#)

3.5.8. Reload the Application Server using the Management CLI

Prerequisites

- [Section 3.4.2, “Launch the Management CLI”](#)

From the Management CLI, use the **reload** operation to shut down all services and restart the JBoss EAP instance. Note that the JVM itself is not restarted. When the **reload** is complete the Management CLI will automatically reconnect.

For more details on operation requests, see [Section 3.4.8, “Use Operations and Commands in the Management CLI”](#).

Example 3.20. Reload the Application Server

```
[standalone@localhost:9999 /]reload
{"outcome" => "success"}
```

[Report a bug](#)

3.5.9. Shut the Application Server down using the Management CLI

Prerequisites

- [Section 3.4.2, “Launch the Management CLI”](#)

Procedure 3.17. Shut down the Application Server

- **Run the shutdown operation**
 - From the Management CLI, use the **shutdown** operation to shut the server down via the **System.exit(0)** system call. For more details on operation requests, refer to the topic [Section 3.4.8, “Use Operations and Commands in the Management CLI”](#).

- In the standalone mode, use the following command:

```
[standalone@localhost:9999 /]shutdown
```

- In the domain mode, use the following command with the appropriate host name:

```
[domain@localhost:9999 /]shutdown --host=master
```

- To connect to a detached CLI instance and shut down the server, execute the following command:

```
jboss-cli.sh --connect command=shutdown
```

- To connect to a remote CLI instance and shut down the server, execute the following command:

```
[disconnected /] connect IP_ADDRESS
[standalone@IP_ADDRESS:9999 /] shutdown
```

Replace *IP_ADDRESS* with the IP address of your instance.



NOTE

Appending the **--restart=true** argument to the **shutdown** command (as shown below) will prompt the server to restart.

```
[standalone@localhost:9999 /]shutdown --restart=true
```

Result

The application server is shut down. The Management CLI will be disconnected as the runtime is unavailable.

[Report a bug](#)

3.5.10. Configure an Attribute with the Management CLI

Prerequisites

- [Section 3.4.2, “Launch the Management CLI”](#)

Summary

The **write-attribute** operation is a global operation used to write or modify a selected resource attribute. You can use the operation to make persistent changes and to modify the configuration settings of your managed server instances. The request properties include the following parameters.

Request Properties

name

The name of the attribute to set the value for under the selected resource.

value

The desired value of the attribute under the selected resource. May be null if the underlying model supports null values.

Procedure 3.18. Configure a Resource Attribute with the Management CLI

- Run the **write-attribute** operation

From the Management CLI, use the **write-attribute** operation to modify the value of a resource attribute. The operation can be run at the child node of the resource or at the root node of the Management CLI where the full resource path is specified.

Example 3.21. Disable the deployment scanner with the **write-attribute** operation

The following example uses the **write-attribute** operation to disable the deployment scanner. The operation is run from the root node, using tab completion to aid in populating the correct resource path.

```
[standalone@localhost:9999 /] /subsystem=deployment-scanner/scanner=default:write-attribute(name=scan-enabled,value=false)
{"outcome" => "success"}
```

The results of the operation can be confirmed directly with the **read-attribute** operation.

```
[standalone@localhost:9999 /] /subsystem=deployment-scanner/scanner=default:read-attribute(name=scan-enabled)
{
  "outcome" => "success",
  "result" => false
}
```

The results can also be confirmed by listing all of the node's available resource attributes with the **read-resource** operation. In the following example, this particular configuration shows the **scan-enabled** attribute is now set to **false**.

```
[standalone@localhost:9999 /] /subsystem=deployment-scanner/scanner=default:read-resource
{
  "outcome" => "success",
  "result" => {
    "auto-deploy-exploded" => false,
    "auto-deploy-xml" => true,
    "auto-deploy-zipped" => true,
    "deployment-timeout" => 600,
    "path" => "deployments",
    "relative-to" => "jboss.server.base.dir",
    "scan-enabled" => false,
    "scan-interval" => 5000
  }
}
```

Result

The resource attribute is updated.

[Report a bug](#)

3.5.11. Configure System Properties Using the Management CLI

Procedure 3.19. Configure System Properties Using the Management CLI

1. Start the JBoss EAP server.

2. Launch the Management CLI using the command for your operating system.

For Linux:

```
EAP_HOME/bin/jboss-cli.sh --connect
```

For Windows:

```
EAP_HOME\bin\jboss-cli.bat --connect
```

3. Add a system property.

The command you use depends on whether you are running a standalone server or a managed domain. If you are running a managed domain, you can add system properties to any or all of the servers running in that domain.

- Add a system property on a standalone server using the following syntax:

```
/system-property=PROPERTY_NAME:add(value=PROPERTY_VALUE)
```

Example 3.22. Add a system property to a standalone server

```
[standalone@localhost:9999 /] /system-  
property=property.mybean.queue:add(value=java:/queue/MyBeanQueue)  
{"outcome" => "success"}
```

- Add a system property to all hosts and servers in a managed domain using the following syntax:

```
/system-property=PROPERTY_NAME:add(value=PROPERTY_VALUE)
```

Example 3.23. Add a system property to all servers in a managed domain

```
[domain@localhost:9999 /] /system-  
property=property.mybean.queue:add(value=java:/queue/MyBeanQueue)  
{  
  "outcome" => "success",  
  "result" => undefined,  
  "server-groups" => {"main-server-group" => {"host" => {"master" => {  
    "server-one" => {"response" => {"outcome" => "success"}},  
    "server-two" => {"response" => {"outcome" => "success"}}}  
  }  
}}}
```

- Add a system property to a host and its server instances in a managed domain using the following syntax:

```
/host=master/system-property=PROPERTY_NAME:add(value=PROPERTY_VALUE)
```

Example 3.24. Add a system property to a host and its servers in a domain


```
[domain@localhost:9999 /] /host=master/system-
property=property.mybean.queue:add(value=java:/queue/MyBeanQueue)
{
  "outcome" => "success",
  "result" => undefined,
  "server-groups" => {"main-server-group" => {"host" => {"master" => {
    "server-one" => {"response" => {"outcome" => "success"}},
    "server-two" => {"response" => {"outcome" => "success"}}
  }}}
}
```

- Add a system property to a server instance in a managed domain using the following syntax:

```
/host=master/server-config=server-one/system-
property=PROPERTY_NAME:add(value=PROPERTY_VALUE)
```

Example 3.25. Add a system property to a server instance in a managed domain

```
[domain@localhost:9999 /] /host=master/server-config=server-one/system-
property=property.mybean.queue:add(value=java:/queue/MyBeanQueue)
{
  "outcome" => "success",
  "result" => undefined,
  "server-groups" => {"main-server-group" => {"host" => {"master" => {"server-one"
=> {"response" => {"outcome" => "success"}}}}}
}
```

4. Read a system property.

The command you use depends on whether you are running a standalone server or a managed domain.

- Read a system property from a standalone server using the following syntax:

```
/system-property=PROPERTY_NAME:read-resource
```

Example 3.26. Read a system property from a standalone server

```
[standalone@localhost:9999 /] /system-property=property.mybean.queue:read-
resource
{
  "outcome" => "success",
  "result" => {"value" => "java:/queue/MyBeanQueue"}
}
```

- Read a system property from all hosts and servers in a managed domain using the following syntax:

```
/system-property=PROPERTY_NAME:read-resource
```

Example 3.27. Read a system property from all servers in a managed domain

```
[domain@localhost:9999 /] /system-property=property.mybean.queue:read-resource
{
  "outcome" => "success",
  "result" => {
    "boot-time" => true,
    "value" => "java:/queue/MyBeanQueue"
  }
}
```

- Read a system property from a host and its server instances in a managed domain using the following syntax:

```
/host=master/system-property=PROPERTY_NAME:read-resource
```

Example 3.28. Read a system property from a host and its servers in a domain

```
[domain@localhost:9999 /] /host=master/system-
property=property.mybean.queue:read-resource
{
  "outcome" => "success",
  "result" => {
    "boot-time" => true,
    "value" => "java:/queue/MyBeanQueue"
  }
}
```

- Read a system property from a server instance in a managed domain using the following syntax:

```
/host=master/server-config=server-one/system-property=PROPERTY_NAME:read-
resource
```

Example 3.29. Read a system property from a server instance in a managed domain

```
[domain@localhost:9999 /] /host=master/server-config=server-one/system-
property=property.mybean.queue:read-resource
{
  "outcome" => "success",
  "result" => {
    "boot-time" => true,
    "value" => "java:/queue/MyBeanQueue"
  }
}
```

5. Remove a system property.

The command you use depends on whether you are running a standalone server or a managed domain.

- Remove a system property from a standalone server using the following syntax:

```
/system-property=PROPERTY_NAME:remove
```

Example 3.30. Remove a system property from a standalone server

```
[standalone@localhost:9999 /] /system-property=property.mybean.queue:remove
{"outcome" => "success"}
```

- Remove a system property from all hosts and servers in a managed domain using the following syntax:

```
/system-property=PROPERTY_NAME:remove
```

Example 3.31. Remove a system property from all hosts and servers in a domain

```
[domain@localhost:9999 /] /system-property=property.mybean.queue:remove
{
  "outcome" => "success",
  "result" => undefined,
  "server-groups" => {"main-server-group" => {"host" => {"master" => {
    "server-one" => {"response" => {"outcome" => "success"}},
    "server-two" => {"response" => {"outcome" => "success"}}
  }}}
}
```

- Remove a system property from a host and its server instances in a managed domain using the following syntax:

```
/host=master/system-property=PROPERTY_NAME:remove
```

Example 3.32. Remove a system property from a host and its instances in a domain

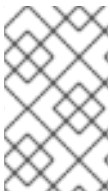
```
[domain@localhost:9999 /] /host=master/system-
property=property.mybean.queue:remove
{
  "outcome" => "success",
  "result" => undefined,
  "server-groups" => {"main-server-group" => {"host" => {"master" => {
    "server-one" => {"response" => {"outcome" => "success"}},
    "server-two" => {"response" => {"outcome" => "success"}}
  }}}
}
```

- Remove a system property from a server instance in a managed domain using the following syntax:

```
/host=master/server-config=server-one/system-property=PROPERTY_NAME:remove
```

Example 3.33. Remove a system property from a server in a managed domain

```
[domain@localhost:9999 /] /host=master/server-config=server-one/system-
property=property.mybean.queue:remove
{
  "outcome" => "success",
  "result" => undefined,
  "server-groups" => {"main-server-group" => {"host" => {"master" => {"server-one"
=> {"response" => {"outcome" => "success"}}}}}}}}
}
```



NOTE

Any system property which contains the text password (regardless of case) is replaced with the text **redacted** when output via logging. This improves security by avoiding having passwords output in plain text in log files.

[Report a bug](#)

3.5.12. Create a New Server with the Management CLI

The following example creates a new server on the host **master** and adds it to **clustered-server-group**:

```
[domain@localhost:9999 /] /host=master/server-config=clustered-server-1:add(group=clustered-
server-group)
```

[Report a bug](#)

3.6. THE MANAGEMENT CLI COMMAND HISTORY

3.6.1. About the Management CLI Command History

The Management CLI features a command history functionality that is enabled by default in the application server installation. The history is kept both as a record in the volatile memory of the active CLI session, and appended to a log file that saves automatically in the user's home directory as **.jboss-cli-history**. This history file is configured by default to record up to a maximum of 500 CLI commands.

The **history** command by itself will return the history of the current session, or with additional arguments will disable, enable or clear the history from the session memory. The Management CLI also features the ability to use your keyboard's arrow keys to go back and forth in the history of commands and operations.

Functions of the Management CLI history

- [Section 3.6.2, "View Management CLI Command History"](#)
- [Section 3.6.3, "Clear the Management CLI Command History"](#)

- [Section 3.6.4, "Disable the Management CLI Command History"](#)
- [Section 3.6.5, "Enable the Management CLI Command History"](#)

[Report a bug](#)

3.6.2. View Management CLI Command History

Prerequisites

- [Section 3.4.2, "Launch the Management CLI"](#)

Procedure 3.20. View the Management CLI Command History

- **Run the `history` command**

From the Management CLI, enter the **history** command:

```
[standalone@localhost:9999 /] history
```

Result

The CLI command history stored in memory since the CLI startup or the history clear command is displayed.

[Report a bug](#)

3.6.3. Clear the Management CLI Command History

Prerequisites

- [Section 3.4.2, "Launch the Management CLI"](#)

Procedure 3.21. Clear the Management CLI Command History

- **Run the `history --clear` command**

From the Management CLI, enter the **history --clear** command:

```
[standalone@localhost:9999 /] history --clear
```

Result

The history of commands recorded since the CLI startup is cleared from the session memory. The command history is still present in the **.jboss-cli-history** file saved to the user's home directory.

[Report a bug](#)

3.6.4. Disable the Management CLI Command History

Prerequisites

- [Section 3.4.2, "Launch the Management CLI"](#)

Procedure 3.22. Disable the Management CLI Command History

- **Run the `history --disable` command**

From the Management CLI, enter the **history --disable** command:

```
[standalone@localhost:9999 /] history --disable
```

Result

Commands made in the CLI will not be recorded either in memory or in the **.jboss-cli-history** file saved to the user's home directory.

[Report a bug](#)

3.6.5. Enable the Management CLI Command History

Prerequisites

- [Section 3.4.2, "Launch the Management CLI"](#)

Procedure 3.23. Enable the Management CLI Command History

- **Run the `history --enable` command**

From the Management CLI, enter the **history --enable** command:

```
[standalone@localhost:9999 /] history --enable
```

Result

Commands made in the CLI are recorded in memory and in the **.jboss-cli-history** file saved to the user's home directory.

[Report a bug](#)

3.7. MANAGEMENT INTERFACE AUDIT LOGGING

3.7.1. About Management Interface Audit Logging

When audit logging is enabled, all operations performed using the Management Console, Management CLI interface, or a custom-written management application, are subject to audit logging.

The audit log entries are stored in JSON format and, based on your configuration, can be stored in files, sent to a syslog server or both. Audit logging can only be configured using the Management CLI and is disabled by default.

Login and logout events cannot be audited as there is no 'authenticated session' in EAP. Instead, audit messages are logged when an operation is received from the user.



NOTE

By default, audit logging is not active. Audit logging can only be configured using the Management CLI.

To list all available management interface audit logging configuration options and their current values, enter the following Management CLI command.

**NOTE**

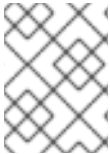
Add the prefix **/host=HOST_NAME** to the command for a managed domain.

```
[... /] /core-service=management/access=audit:read-resource(recursive=true)
```

[Report a bug](#)

3.7.2. Enable Management Interface Audit Logging to a File

To enable audit logging output to a file, enter the following Management CLI command.

**NOTE**

If the change is to be applied to a managed domain, add the prefix **/host=HOST_NAME** to the following command.

```
/core-service=management/access=audit/logger=audit-log:write-attribute(name=enabled,value=true)
```

Management operations are now logged to a file:

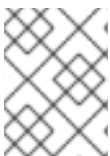
- Standalone mode: **EAP_HOME/standalone/data/audit-log.log**
- Domain mode: **EAP_HOME/domain/data/audit-log.log**

For details of all file handler attributes, see [Section A.3, “Management Interface Audit Logging Reference”](#).

[Report a bug](#)

3.7.3. Enable Management Interface Audit Logging to a Syslog Server

By default, audit logging is preconfigured to output to a file when enabled. This procedure configures output to a syslog server and enables audit logging to a file. For details of all syslog handler attributes see [Section A.3, “Management Interface Audit Logging Reference”](#).

**NOTE**

If the change is to be applied to a managed domain, add the prefix **/host=HOST_NAME** to the **/core-service** commands.

Procedure 3.24. Enable Audit Logging to a Syslog Server

1. **Enable Audit Logging**

Execute the following command:

```
[.. /] /core-service=management/access=audit/logger=audit-log:write-attribute(name=enabled,value=true)
```

2. **Create a syslog Handler**

In this example the **syslog** server is running on the same server as the JBoss EAP instance, on port 514. Replace the values of the **host** attribute with values appropriate to your environment.

Example 3.34. Example syslog handler

```
[.. /]batch
[.. / #]/core-service=management/access=audit/syslog-
handler=mysyslog:add(formatter=json-formatter)
[.. / #]/core-service=management/access=audit/syslog-
handler=mysyslog/protocol=udp:add(host=localhost,port=514)
[.. /]run-batch
```

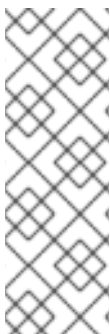
3. Add a Reference to the `syslog` Handler

Execute the following:

```
[.. /]/core-service=management/access=audit/logger=audit-log/handler=mysyslog:add
```

Result

Management interface audit log entries are logged on the **syslog** server.

**NOTE**

Enabling audit logging to a Syslog Server in JBoss EAP will not work unless logging is enabled in the operating system as well.

For more information on **rsyslog** configurations on Red Hat Enterprise Linux, refer to the "**Basic Configuration of rsyslog**" section in the **System Administrator's Guide** for Red Hat Enterprise Linux in https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/

[Report a bug](#)

3.7.4. Disable Management Interface Audit Logging

The audit logging to a file or a **syslog** server can be disabled by executing the following command:

```
/core-service=management/access=audit/logger=audit-log:write-attribute(name=enabled,value=false)
```

[Report a bug](#)

3.7.5. Read a Management Interface Audit Log

Audit log entries output to file(s) are best viewed with a text *viewer*, while those output to a syslog server are best viewed using a syslog viewer application.

**NOTE**

Using a text *editor* for viewing log files is not recommended as some may prevent further log entries being written to the log file.

The management interface audit logs are output in JSON format. Each log entry begins with an optional timestamp, then the fields listed in the *Management Interface Audit Log Fields* table.

Table 3.5. Management Interface Audit Log Fields

Field Name	Description
type	This can have the values core , meaning it is a management operation, or jmx meaning it comes from the JMX subsystem (see the JMX subsystem for configuration of the JMX subsystem's audit logging).
r/o	Has the value true if the operation does not change the management model, false otherwise.
booting	Has the value true if the operation was executed during the bootup process, false if it was executed once the server is up and running.
version	The version number of the JBoss EAP instance.
user	The username of the authenticated user. If the operation occurs via the Management CLI on the same machine as the running server, the special user \$local is used.
domainUUID	An ID to link together all operations as they are propagated from the domain controller to its servers, slave host controllers, and slave host controller servers.
access	This can have one of the following values: <ul style="list-style-type: none"> ● NATIVE - The operation came in through the native management interface, for example the Management CLI. ● HTTP - The operation came in through the domain HTTP interface, for example the Management Console. ● JMX - The operation came in through the JMX subsystem. See JMX for how to configure audit logging for JMX.
remote-address	The address of the client executing this operation.
success	Has the value true if the operation is successful, false if it was rolled back.
ops	The operations being executed. This is a list of the operations serialized to JSON. At boot this is the operations resulting from parsing the XML. Once booted the list typically contains a single entry.

[Report a bug](#)

CHAPTER 4. USER MANAGEMENT

4.1. ABOUT JBOSS EAP USER MANAGEMENT

Depending on how JBoss EAP 6 is installed, there may be no user accounts initially available to access the management interfaces.

If you install the platform using a graphical installer, only one user account with the necessary privileges to access the JBoss EAP 6 management interfaces is created during the installation.

If you install the platform manually using a ZIP archive no user accounts with appropriate privileges are created during installation.

If you install the platform manually using the JAR installer at a console, one user account with appropriate privileges is created during installation.

HTTP-based communication with JBoss EAP 6 is considered remote access, even if the traffic originates on the localhost. Therefore, you must create at least one administrative user to use the management console. If you attempt to access the management console before adding a user, you will receive an error because it does not even deploy until the user is added.

This guide covers simple user management for JBoss EAP 6 using the **add-user.sh** script. For more advanced authentication and authorization options, such as LDAP or Role-Based Access Control (RBAC), see the *Core Management Authentication* section of the *JBoss EAP Security Architecture* document.

[Report a bug](#)

4.2. USER CREATION

4.2.1. Add the User for the Management Interfaces

The following procedure documents how to create the initial administrative user, in the event such a user is not created by the chosen installation method. This initial administrative user can use the web-based Management Console and remote instances of the Management CLI to configure and administer JBoss EAP 6 from remote systems.

Procedure 4.1. Create the Initial Administrative User for the Remote Management Interfaces

1. Run the **add-user.sh** or **add-user.bat** script.

Change to the **EAP_HOME/bin/** directory. Invoke the appropriate script for your operating system.

Red Hat Enterprise Linux

```
[user@host bin]$ ./add-user.sh
```

Microsoft Windows Server

```
C:\bin> add-user.bat
```

2. Choose to add a Management user.

Press **ENTER** to select the default option **a** to add a Management user.

This user is added to the **ManagementRealm** and is authorized to perform management operations using the web-based Management Console or command-line based Management CLI. The other choice, **b**, adds a user to the **ApplicationRealm**, and provides no particular permissions. That realm is provided for use with applications.

3. **Enter the desired username and password.**

When prompted, enter the username and password. You will be prompted to confirm the password.

4. **Enter group information.**

Add the group or groups to which the user belongs. If the user belongs to multiple groups, enter a comma-separated list. Leave it blank if you do not want the user to belong to any groups.

5. **Review the information and confirm.**

You are prompted to confirm the information. If you are satisfied, type **yes**.

6. **Choose whether the user represents a remote JBoss EAP 6 server instance.**

Besides administrators, the other type of user which occasionally needs to be added to JBoss EAP 6 in the **ManagementRealm** is a user representing another instance of JBoss EAP 6, which must be able to authenticate to join a cluster as a member. The next prompt allows you to designate your added user for this purpose. If you select **yes**, you will be given a hashed **secret** value, representing the user's password, which would need to be added to a different configuration file. For the purposes of this task, answer **no** to this question.

7. **Enter additional users.**

You can enter additional users if desired, by repeating the procedure. You can also add them at any time on a running system. Instead of choosing the default security realm, you can add users to other realms to fine-tune their authorizations.

8. **Create users non-interactively.**

You can create users non-interactively, by passing in each parameter at the command line. This approach is not recommended on shared systems, because the passwords will be visible in log and history files. The syntax for the command, using the management realm, is:

```
[user@host bin]$ ./add-user.sh username password
```

To use the application realm, use the **-a** parameter.

```
[user@host bin]$ ./add-user.sh -a username password
```

9. You can suppress the normal output of the add-user script by passing the **--silent** parameter. This applies only if the minimum parameters **username** and **password** have been specified. Error messages will still be shown.

Result

Any users you add are activated within the security realms you have specified. Users active within the **ManagementRealm** realm are able to manage JBoss EAP 6 from remote systems.

[Report a bug](#)

4.2.2. Pass Arguments to the User Management add-user Script

You can run the **add-user.sh** or **add-user.bat** command interactively or you can pass the arguments on the command line. This section describes the options available when passing command line arguments to the add-user script.

For a comprehensive list of the command line arguments available for the **add-user.sh** or **add-user.bat** command, see [Section 4.2.3, "Add-user Command Arguments"](#).

For information on how to specify an alternate properties file and location, see [Section 4.2.4, "Specify Alternate Properties Files for User Management Information"](#).

For examples that demonstrate how to pass arguments on the **add-user.sh** or **add-user.bat** command, see [Section 4.3.1, "Create a User Belonging to a Single Group Using the Default Properties Files"](#), [Section 4.3.2, "Create a User Belonging to Multiple Groups Using the Default Properties Files"](#), [Section 4.3.3, "Create a User With Administrator Privileges in the Default Realm Using the Default Properties Files"](#) and [Section 4.3.4, "Create a User Belonging to Single Group Using Alternate Properties Files to Store the Information"](#).

[Report a bug](#)

4.2.3. Add-user Command Arguments

The following table describes the arguments available for the **add-user.sh** or **add-user.bat** command.

Table 4.1. Add-user Command Arguments

Command Line Argument	Argument Value	Description
-a	N/A	This argument specifies to create a user in the application realm. If omitted, the default is to create a user in the management realm.
-dc	<i>DOMAIN_CONFIGURATION_DIRECTORY</i>	This argument specifies the domain configuration directory that will contain the properties files. If it is omitted, the default directory is EAP_HOME/domain/configuration/ .
-sc	<i>SERVER_CONFIGURATION_DIRECTORY</i>	This argument specifies an alternate standalone server configuration directory that will contain the properties files. If it is omitted, the default directory is EAP_HOME/standalone/configuration/ .
-up --user-properties	<i>USER_PROPERTIES_FILE</i>	This argument specifies the name of the alternate user properties file. It can be an absolute path or it can be a file name used in conjunction with the -sc or -dc argument that specifies the alternate configuration directory.
-g --group	<i>GROUP_LIST</i>	A comma-separated list of groups to assign to this user.
-gp --group-properties	<i>GROUP_PROPERTIES_FILE</i>	This argument specifies the name of the alternate group properties file. It can be an absolute path or it can be a file name used in conjunction with the -sc or -dc argument that specifies the alternate configuration directory.

Command Line Argument	Argument Value	Description
-p --password	<i>PASSWORD</i>	The password of the user. The password must satisfy the following requirements: <ul style="list-style-type: none"> ● It must contain at least 8 characters. ● It must contain at least one alphabetic character. ● It must contain at least one digit. ● It must contain at least one non-alphanumeric symbol
-u --user	<i>USER_NAME</i>	The name of the user. Only alphanumeric characters and the following symbols are valid: ./=@\.
-r --realm	<i>REALM_NAME</i>	The name of the realm used to secure the management interfaces. If omitted, the default is ManagementRealm .
-s --silent	N/A	Run the add-user script with no output to the console.
-h --help	N/A	Display usage information for the add-user script.

[Report a bug](#)

4.2.4. Specify Alternate Properties Files for User Management Information

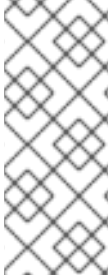
Overview

By default, user and role information created using the **add-user.sh** or **add-user.bat** script are stored in properties files located in the server configuration directory. The server configuration information is stored in the **EAP_HOME/standalone/configuration/** directory and the domain configuration information is stored in the **EAP_HOME/domain/configuration/** directory. This topic describes how to override the default file names and locations.

Specify Alternate Properties Files

- To specify an alternate directory for the server configuration, use the **-sc** argument. This argument specifies an alternate directory that will contain the server configuration properties files.
- To specify an alternate directory for the domain configuration, use the **-dc** argument. This argument specifies an alternate directory that will contain the domain configuration properties files.

- To specify an alternate user configuration properties file, use the **-up** or **--user-properties** argument. It can be an absolute path or it can be a file name used in conjunction with the **-sc** or **-dc** argument that specifies the alternate configuration directory.
- To specify an alternate group configuration properties file, use the **-gp** or **--group-properties** argument. It can be an absolute path or it can be a file name used in conjunction with the **-sc** or **-dc** argument that specifies the alternate configuration directory.



NOTE

The **add-user** command is intended to operate on existing properties files. Any alternate properties files specified in command line arguments must exist or you will see the following error:

```
JBAS015234: No appusers.properties files found
```

For more information about command arguments, see [Section 4.2.3, "Add-user Command Arguments"](#).

[Report a bug](#)

4.3. ADD-USER SCRIPT COMMAND LINE EXAMPLES

4.3.1. Create a User Belonging to a Single Group Using the Default Properties Files

Example 4.1. Create a user belonging to a single group

```
EAP_HOME/bin/add-user.sh -a -u 'appuser1' -p 'password1!' -g 'guest'
```

This example gives the following results.

- The user **appuser1** is added to the following default properties files that store user information.

```
EAP_HOME/standalone/configuration/application-users.properties
```

```
EAP_HOME/domain/configuration/application-users.properties
```

- The user **appuser1** with group **guest** is added to the default properties files that store group information.

```
EAP_HOME/standalone/configuration/application-roles.properties
```

```
EAP_HOME/domain/configuration/application-roles.properties
```

[Report a bug](#)

4.3.2. Create a User Belonging to Multiple Groups Using the Default Properties Files

Example 4.2. Create a user belonging to multiple groups

```
EAP_HOME/bin/add-user.sh -a -u 'appuser1' -p 'password1!' -g 'guest,app1group,app2group'
```

This example gives the following results.

- The user **appuser1** is added to the following default properties files that store user information.

EAP_HOME/standalone/configuration/application-users.properties

EAP_HOME/domain/configuration/application-users.properties

- The user **appuser1** with groups **guest**, **app1group**, and **app2group** is added to the default properties files that store group information.

EAP_HOME/standalone/configuration/application-roles.properties

EAP_HOME/domain/configuration/application-roles.properties

[Report a bug](#)

4.3.3. Create a User With Administrator Privileges in the Default Realm Using the Default Properties Files

Example 4.3. Create a user with Administrator privileges in the Default Realm

```
EAP_HOME/bin/add-user.sh -u 'adminuser1' -p 'password1!' -g 'admin'
```

This example gives the following results.

- The user **adminuser1** is added to the following default properties files that store user information.

EAP_HOME/standalone/configuration/mgmt-users.properties

EAP_HOME/domain/configuration/mgmt-users.properties

- The user **adminuser1** with group **admin** is added to the default properties files that store group information.

EAP_HOME/standalone/configuration/mgmt-groups.properties

EAP_HOME/domain/configuration/mgmt-groups.properties

[Report a bug](#)

4.3.4. Create a User Belonging to Single Group Using Alternate Properties Files to Store the Information

Example 4.4. Create a user belonging to single group using alternate properties files

```
EAP_HOME/bin/add-user.sh -a -u appuser1 -p password1! -g app1group -sc /home/someusername/userconfigs/ -up appusers.properties -gp appgroups.properties
```

This example gives the following results.

- The user **appuser1** is added to the following properties file and that file is now the default file

to store user information.

/home/someusername/userconfigs/appusers.properties

- The user **appuser1** with group **app1group** is added to the following properties file and that file is now the default file to store group information.

/home/someusername/userconfigs/appgroups.properties

[Report a bug](#)

CHAPTER 5. NETWORK AND PORT CONFIGURATION

5.1. INTERFACES

5.1.1. About Interfaces

JBoss EAP uses named interface references throughout the configuration. This gives the configuration the ability to reference individual interface declarations with logical names, rather than requiring the full details of the interface at each use.

The use of logical names also allows for consistency in group references to named interfaces, where server instances on a managed domain may contain varying interface details across multiple machines. With logical names, each server instance can correspond to a logical name group, allowing for easier interface group administration.

Network interfaces are declared by specifying a logical name and a selection criteria for the physical interface.

The JBoss EAP default configuration includes both a **management** and **public** interface names. The **management** interface name can be used for all components and services that require the management layer, including the HTTP Management Endpoint. The **public** interface name can be used for all application-related network communications, including Web and Messaging.

The use of default names is not compulsory. New logical names can be created and substituted for default names.

The **domain.xml**, **host.xml** and **standalone.xml** configuration files all include interface declarations. The declaration criteria can reference a wildcard address or specify a set of one or more characteristics that an interface or address must have in order to be a valid match.

The three configuration files remain directly editable but manual edits are no longer required. The Management CLI and Management Console provide a safe, controlled and persistent environment for configuration changes.

The following examples show multiple possible configurations of interface declarations, typically defined in either the **standalone.xml** or **host.xml** configuration files. Using these files allow remote host groups to maintain specific interface attributes, while still allowing references to domain controller interfaces.

The following example shows a specific **inet-address** value specified for both the management and public relative name groups.

Example 5.1. An interface group created with an **inet-address** value

```
<interfaces>
  <interface name="management">
    <inet-address value="127.0.0.1"/>
  </interface>
  <interface name="public">
    <inet-address value="127.0.0.1"/>
  </interface>
</interfaces>
```

The following example shows a global interface group. It uses the **any-address** element to declare a wildcard address.

Example 5.2. A global group created with a wildcard declaration

```
<interface name="global">
  <!-- Use the wild-card address -->
  <any-address/>
</interface>
```

The following example declares a network interface card (eth0) under a relative group called **external**.

Example 5.3. An external group created with an NIC value

```
<interface name="external">
  <nic name="eth0"/>
</interface>
```

The following example declares the default group with requirements. These requirements set the conditions for the interface to be a valid match. This is an example of how JBoss EAP allows for the creation of interface declaration groups with specific properties that can then be referenced using the interface's name. This helps in reducing configuration complexity and administration overhead across multiple server instances.

Example 5.4. A default group created with specific conditional values

```
<interface name="default">
  <!-- Match any interface/address on the right subnet if it's
  up, supports multicast, and isn't point-to-point -->
  <subnet-match value="192.168.0.0/16"/>
  <up/>
  <multicast/>
  <not>
    <point-to-point/>
  </not>
</interface>
```

[Report a bug](#)

5.1.2. Configure Interfaces

The default interface configurations in the **standalone.xml** and **host.xml** configuration files offer three named interfaces with relative interface tokens for each. Use the Management Console or Management CLI to configure additional attributes and values, as listed in the table below. The relative interface bindings can be replaced with specific values as required but note that if you do so, you will be unable to pass an interface value at server runtime, as the **-b** switch can only override a relative value.

Example 5.5. Default Interface Configurations

```

<interfaces>
  <interface name="management">
    <inet-address value="{jboss.bind.address.management:127.0.0.1}"/>
  </interface>
  <interface name="public">
    <inet-address value="{jboss.bind.address:127.0.0.1}"/>
  </interface>
  <interface name="unsecure">
    <inet-address value="{jboss.bind.address.unsecure:127.0.0.1}"/>
  </interface>
</interfaces>

```

While running multiple servers in a managed domain, interface binding can be assigned to individual servers in their respective **host.xml** files. For example:

```

<servers>
  <server name="server-name" group="main-server-group">
    <interfaces>
      <interface name="public">
        <inet-address value="ip-address"/>
      </interface>
    </interfaces>
  </server>
</servers>

```



NOTE

For the above example, substitute *server-name* with your actual server name and substitute *ip-address* with your actual IP address.

Table 5.1. Interface Attributes and Values

Interface Element	Description
any	Element indicating that part of the selection criteria for an interface should be that it meets at least one, but not necessarily all, of the nested set of criteria.
any-address	<p>Empty element indicating that sockets using this interface should be bound to a wildcard address.</p> <p>The IPv6 wildcard address (::) will be used unless the <code>java.net.preferIPv4Stack</code> system property is set to true, in which case the IPv4 wildcard address (0.0.0.0) will be used.</p> <p>If a socket is bound to an IPv6 anylocal address on a dual-stack machine, it can accept both IPv6 and IPv4 traffic; if it is bound to an IPv4 (IPv4-mapped) anylocal address, it can only accept IPv4 traffic.</p>
any-ipv4-address	Empty element indicating that sockets using this interface should be bound to the IPv4 wildcard address (0.0.0.0).

Interface Element	Description
any-ipv6-address	Empty element indicating that sockets using this interface should be bound to the IPv6 wildcard address (::).
inet-address	Either an IP address in IPv6 or IPv4 dotted decimal notation, or a hostname that can be resolved to an IP address.
link-local-address	Empty element indicating that part of the selection criteria for an interface should be whether or not an address associated with it is link-local.
loopback	Empty element indicating that part of the selection criteria for an interface should be whether or not it is a loopback interface.
loopback-address	A loopback address that may not actually be configured on the machine's loopback interface. Differs from inet-address type in that the given value will be used even if no NIC can be found that has the IP address associated with it.
multicast	Empty element indicating that part of the selection criteria for an interface should be whether or not it supports multicast.
nic	The name of a network interface (e.g. eth0, eth1, lo).
nic-match	A regular expression against which the names of the network interfaces available on the machine can be matched to find an acceptable interface.
not	Element indicating that part of the selection criteria for an interface should be that it does not meet any of the nested set of criteria.
point-to-point	Empty element indicating that part of the selection criteria for an interface should be whether or not it is a point-to-point interface.
public-address	Empty element indicating that part of the selection criteria for an interface should be whether or not it has a publicly routable address.
site-local-address	Empty element indicating that part of the selection criteria for an interface should be whether or not an address associated with it is site-local.
subnet-match	A network IP address and the number of bits in the address' network prefix, written in "slash notation"; e.g. "192.168.0.0/16".
up	Empty element indicating that part of the selection criteria for an interface should be whether or not it is currently up.
virtual	Empty element indicating that part of the selection criteria for an interface should be whether or not it is a virtual interface.

Interface Element	Description
-------------------	-------------

- **Configure Interface Attributes**

- **Configure Interface Attributes with the Management CLI**

You can use tab completion to complete the command string as you type, as well as to expose the available attributes.

Use the Management CLI to add a new server and configure instances to it, effectively adding the same piece of configuration to the XML. Substitute *server-name* with your actual server name and substitute *ip-address* with your actual IP address.

```
/host=master/server-config=server-name:add(group=main-server-group)
/host=master/server-config=server-name/interface=public:add(inet-address=ip-address)
```

Use the Management CLI to add new interfaces and write new values to the interface attributes.

- Add a New Interface**

The **add** operation creates new interfaces as required. The **add** command runs from the root of the Management CLI session, and in the following example it creates a new interface name title *interfacename*, with an **inet-address** declared as *12.0.0.2*.

```
/interface=interfacename/:add(inet-address=12.0.0.2)
```

- Edit Interface Attributes**

The **write-attribute** operation writes new values to an attribute. The following example updates the **inet-address** value to *12.0.0.8*.

```
/interface=interfacename/:write-attribute(name=inet-address, value=12.0.0.8)
```

- Verify Interface Attributes**

Confirm that the attribute values have changed by running the **read-resource** operation with the **include-runtime=true** parameter to expose all current values active in the server model. For example:

```
[standalone@localhost:9999 interface=public] :read-resource(include-runtime=true)
```

- **Configure Interface Attributes with the Management Console**

- Log into the Management Console.**

Log into the Management Console of your Managed Domain or Standalone Server instance.

- Navigate to Configuration tab**

Select the **Configuration** tab from the top of the screen.

**NOTE**

For Domain Mode, select a profile from the **Profile** drop-down menu at the top left of the screen.

- c. **Select Interfaces from the Navigation Menu.**
Select the **Interfaces** menu item from the navigation menu.
- d. **Add a New Interface**
 - i. Click **Add**.
 - ii. Enter required values for **Name**, **Inet Address** and **Address Wildcard**.
 - iii. Click **Save**.
- e. **Edit Interface Attributes**
 - i. Select the interface that you need to edit from the **Available Interfaces** list and click **Edit**.
 - ii. Enter required values for **Name**, **Inet Address** and **Address Wildcard**.
 - iii. Click **Save**.

[Report a bug](#)

5.2. SOCKET BINDING GROUPS

5.2.1. About Socket Binding Groups

Socket bindings and socket binding groups allow you to define network ports and their relationship to the networking interfaces required for your JBoss EAP 6 configuration.

A socket binding is a named configuration for a socket. The declarations for these named configurations can be found in both the **domain.xml** and **standalone.xml** configuration files. Other sections of the configuration can then reference those sockets by their logical name, rather than having to include the full details of the socket configuration. This allows you to reference relative socket configurations which may otherwise vary on different machines.

Socket bindings are collected under a socket binding group. A socket binding group is a collection of socket binding declarations that are grouped under a logical name. The named group can then be referenced throughout the configuration. A standalone server contains only one such group, while a managed domain instance can contain multiple groups. You can create a socket binding group for each server group in the managed domain, or share a socket binding group between multiple server groups.

The naming groups allow for simplified references to be used for particular groups of socket bindings when configuring server groups in the case of a managed domain. Another common use is for the configuration and management of multiple instances of the standalone server on the one system. The following examples show the default socket binding groups in the configuration files for the standalone and domain instances.

Example 5.6. Default socket bindings for the standalone configuration

The default socket binding groups in the **standalone.xml** configuration file are grouped under **standard-sockets**. This group is also referenced to the **public** interface, using the same logical referencing methodology.

```
<socket-binding-group name="standard-sockets" default-interface="public" port-
offset="${jboss.socket.binding.port-offset:0}">
  <socket-binding name="management-native" interface="management" port="${jboss
.management.native.port:9999}"/>
  <socket-binding name="management-http" interface="management" port="${jboss
.management.http.port:9990}"/>
  <socket-binding name="management-https" interface="management" port="${jboss
.management.https.port:9443}"/>
  <socket-binding name="ajp" port="8009"/>
  <socket-binding name="http" port="8080"/>
  <socket-binding name="https" port="8443"/>
  <socket-binding name="remoting" port="4447"/>
  <socket-binding name="txn-recovery-environment" port="4712"/>
  <socket-binding name="txn-status-manager" port="4713"/>
  <outbound-socket-binding name="mail-smtp">
    <remote-destination host="localhost" port="25"/>
  </outbound-socket-binding>
</socket-binding-group>
```

Example 5.7. Default socket bindings for the domain configuration

The default socket binding groups in the **domain.xml** configuration file contain four groups: the **standard-sockets**, **ha-sockets**, **full-sockets** and the **full-ha-sockets** groups. These groups are also referenced to an interface called **public**.

```
<socket-binding-groups>
  <socket-binding-group name="standard-sockets" default-interface="public">
    <!-- Needed for server groups using the 'default' profile -->
    <socket-binding name="ajp" port="8009"/>
    <socket-binding name="http" port="8080"/>
    <socket-binding name="https" port="8443"/>
    <socket-binding name="remoting" port="4447"/>
    <socket-binding name="txn-recovery-environment" port="4712"/>
    <socket-binding name="txn-status-manager" port="4713"/>
    <outbound-socket-binding name="mail-smtp">
      <remote-destination host="localhost" port="25"/>
    </outbound-socket-binding>
  </socket-binding-group>
  <socket-binding-group name="ha-sockets" default-interface="public">
    <!-- Needed for server groups using the 'ha' profile -->
    <socket-binding name="ajp" port="8009"/>
    <socket-binding name="http" port="8080"/>
    <socket-binding name="https" port="8443"/>
    <socket-binding name="jgroups-mping" port="0" multicast-address="${jboss
.default.multicast.address:230.0.0.4}" multicast-port="45700"/>
    <socket-binding name="jgroups-tcp" port="7600"/>
    <socket-binding name="jgroups-tcp-fd" port="57600"/>
    <socket-binding name="jgroups-udp" port="55200" multicast-address="${jboss
.default.multicast.address:230.0.0.4}" multicast-port="45688"/>
  </socket-binding-group>
  <socket-binding-group name="full-sockets" default-interface="public">
    <!-- Needed for server groups using the 'full' profile -->
    <socket-binding name="ajp" port="8009"/>
    <socket-binding name="http" port="8080"/>
    <socket-binding name="https" port="8443"/>
    <socket-binding name="jgroups-mping" port="0" multicast-address="${jboss
.default.multicast.address:230.0.0.4}" multicast-port="45700"/>
    <socket-binding name="jgroups-tcp" port="7600"/>
    <socket-binding name="jgroups-tcp-fd" port="57600"/>
    <socket-binding name="jgroups-udp" port="55200" multicast-address="${jboss
.default.multicast.address:230.0.0.4}" multicast-port="45688"/>
  </socket-binding-group>
  <socket-binding-group name="full-ha-sockets" default-interface="public">
    <!-- Needed for server groups using the 'full-ha' profile -->
    <socket-binding name="ajp" port="8009"/>
    <socket-binding name="http" port="8080"/>
    <socket-binding name="https" port="8443"/>
    <socket-binding name="jgroups-mping" port="0" multicast-address="${jboss
.default.multicast.address:230.0.0.4}" multicast-port="45700"/>
    <socket-binding name="jgroups-tcp" port="7600"/>
    <socket-binding name="jgroups-tcp-fd" port="57600"/>
    <socket-binding name="jgroups-udp" port="55200" multicast-address="${jboss
.default.multicast.address:230.0.0.4}" multicast-port="45688"/>
  </socket-binding-group>
</socket-binding-groups>
```

```

    <socket-binding name="jgroups-udp-fd" port="54200"/>
    <socket-binding name="modcluster" port="0" multicast-address="224.0.1.105"
multicast-port="23364"/>
    <socket-binding name="remoting" port="4447"/>
    <socket-binding name="txn-recovery-environment" port="4712"/>
    <socket-binding name="txn-status-manager" port="4713"/>
    <outbound-socket-binding name="mail-smtp">
    <remote-destination host="localhost" port="25"/>
    </outbound-socket-binding>
</socket-binding-group>
<socket-binding-group name="full-sockets" default-interface="public">
    <!-- Needed for server groups using the 'full' profile -->
    <socket-binding name="ajp" port="8009"/>
    <socket-binding name="http" port="8080"/>
    <socket-binding name="https" port="8443"/>
    <socket-binding name="jacob" interface="unsecure" port="3528"/>
    <socket-binding name="jacob-ssl" interface="unsecure" port="3529"/>
    <socket-binding name="messaging" port="5445"/>
    <socket-binding name="messaging-group" port="0" multicast-address="{jboss
.messaging.group.address:231.7.7.7}" multicast-port="{jboss.messaging.group.port:9876}"/>
    <socket-binding name="messaging-throughput" port="5455"/>
    <socket-binding name="remoting" port="4447"/>
    <socket-binding name="txn-recovery-environment" port="4712"/>
    <socket-binding name="txn-status-manager" port="4713"/>
    <outbound-socket-binding name="mail-smtp">
    <remote-destination host="localhost" port="25"/>
    </outbound-socket-binding>
</socket-binding-group>
<socket-binding-group name="full-ha-sockets" default-interface="public">
    <!-- Needed for server groups using the 'full-ha' profile -->
    <socket-binding name="ajp" port="8009"/>
    <socket-binding name="http" port="8080"/>
    <socket-binding name="https" port="8443"/>
    <socket-binding name="jacob" interface="unsecure" port="3528"/>
    <socket-binding name="jacob-ssl" interface="unsecure" port="3529"/>
    <socket-binding name="jgroups-mping" port="0" multicast-address="{jboss
.default.multicast.address:230.0.0.4}" multicast-port="45700"/>
    <socket-binding name="jgroups-tcp" port="7600"/>
    <socket-binding name="jgroups-tcp-fd" port="57600"/>
    <socket-binding name="jgroups-udp" port="55200" multicast-address="{jboss
.default.multicast.address:230.0.0.4}" multicast-port="45688"/>
    <socket-binding name="jgroups-udp-fd" port="54200"/>
    <socket-binding name="messaging" port="5445"/>
    <socket-binding name="messaging-group" port="0" multicast-address="{jboss
.messaging.group.address:231.7.7.7}" multicast-port="{jboss.messaging.group.port:9876}"/>
    <socket-binding name="messaging-throughput" port="5455"/>
    <socket-binding name="modcluster" port="0" multicast-address="224.0.1.105"
multicast-port="23364"/>
    <socket-binding name="remoting" port="4447"/>
    <socket-binding name="txn-recovery-environment" port="4712"/>
    <socket-binding name="txn-status-manager" port="4713"/>
    <outbound-socket-binding name="mail-smtp">
    <remote-destination host="localhost" port="25"/>
    </outbound-socket-binding>
</socket-binding-group>
</socket-binding-groups>

```


The socket binding instances can be created and edited in the **standalone.xml** and **domain.xml** source files in the application server directory. The recommended method of managing bindings is to use either the Management Console or the Management CLI. The advantages of using the Management Console include a graphical user interface with a dedicated Socket Binding Group screen under the **General Configuration** section. The Management CLI offers an API and workflow based around a command line approach that allows for batch processing and the use of scripts across the higher and lower levels of the application server configuration. Both interfaces allow for changes to be persisted or otherwise saved to the server configuration.

[Report a bug](#)

5.2.2. Configure Socket Bindings

Socket bindings can be defined in unique socket binding groups. A standalone server contains one such group, the **standard-sockets** group, and is unable to create any further groups. Instead you can create alternate standalone server configuration files. For a managed domain however, you can create multiple socket binding groups and configure the socket bindings that they contain as you require. The following table shows the available attributes for each socket binding.

Table 5.2. Socket Binding Attributes

Attribute	Description	Role
name	Logical name of the socket configuration that should be used elsewhere in the configuration.	Required
port	Base port to which a socket based on this configuration should be bound. Note that servers can be configured to override this base value by applying an increment or decrement to all port values.	Required
interface	Logical name of the interface to which a socket based on this configuration should be bound. If not defined, the value of the default-interface attribute from the enclosing socket binding group will be used.	Optional
multicast-address	If the socket will be used for multicast, the multicast address to use.	Optional
multicast-port	If the socket will be used for multicast, the multicast port to use.	Optional

Attribute	Description	Role
fixed-port	If true , declares that the value of port must always be used for the socket and should not be overridden by applying an increment or decrement.	Optional

- **Configure Socket Bindings in Socket Binding Groups**

Choose either the Management CLI or the management console to configure your socket bindings as required.

- **Configure Socket Bindings Using the Management CLI**

Use the Management CLI to configure socket bindings.

- a. **Add a New Socket Binding**

Use the **add** operation to create a new address setting if required. You can run this command from the root of the Management CLI session, which in the following examples creates a new socket binding titled *newsocket*, with a **port** attribute declared as *1234*. The examples apply for both a standalone server and a managed domain editing on the **standard-sockets** socket binding group as shown.

```
[domain@localhost:9999 /] /socket-binding-group=standard-sockets/socket-binding=newsocket/:add(port=1234)
```

- b. **Edit Pattern Attributes**

Use the **write-attribute** operation to write a new value to an attribute. You can use tab completion to help complete the command string as you type, as well as to expose the available attributes. The following example updates the **port** value to *2020*

```
[domain@localhost:9999 /] /socket-binding-group=standard-sockets/socket-binding=newsocket/:write-attribute(name=port,value=2020)
```

- c. **Confirm Pattern Attributes**

Confirm the values are changed by running the **read-resource** operation with the **include-runtime=true** parameter to expose all current values active in the server model.

```
[domain@localhost:9999 /] /socket-binding-group=standard-sockets/socket-binding=newsocket/:read-resource
```

- **Configure Socket Bindings Using the Management Console**

Use the management console to configure socket bindings.

- a. **Log into the Management Console.**

Log into the management console of your managed domain or standalone server.

- b. **Navigate to the Configuration tab.**

Select the **Configuration** tab at the top of the screen.

- c. **Select the Socket Binding item from the navigation menu.**

Expand the **General Configuration** menu. Select **Socket Binding**. If you are using a managed domain, select the desired group in the **Socket Binding Groups** list.

d. **Add a New Socket Binding**

- i. Click the **Add** button.
- ii. Enter any required values for **Name**, **Port** and **Binding Group**.
- iii. Click **Save** to finish.

e. **Edit Socket Binding**

- i. Select a socket binding from the list and click **Edit**.
- ii. Enter any required values such as **Name**, **Interface** or **Port**.
- iii. Click **Save** to finish.

[Report a bug](#)

5.2.3. Network Ports Used By JBoss EAP 6

The ports used by the JBoss EAP 6 default configuration depend on several factors:

- Whether your server groups use one of the default socket binding groups, or a custom group.
- The requirements of your individual deployments.



NOTE

A numerical port offset can be configured, to alleviate port conflicts when you run multiple servers on the same physical server. If your server uses a numerical port offset, add the offset to the default port number for its server group's socket binding group. For instance, if the HTTP port of the socket binding group is **8080**, and your server uses a port offset of **100**, its HTTP port is **8180**.

Unless otherwise stated, the ports use the TCP protocol.

The default socket binding groups

- **full-ha-sockets**
- **full-sockets**
- **ha-sockets**
- **standard-sockets**

These socket binding groups are available only in **domain.xml**. The standalone server profiles contain only standard socket binding group. This group corresponds to standard-sockets in **standalone.xml**, **ha-sockets** for **standalone-ha.xml**, **full-sockets** for **standalone-full.xml**, and **full-ha-sockets** for **standalone-full-ha.xml**. Standalone profiles contain some more socket bindings, for example, `management-{native,http,https}`.

Table 5.3. Reference of the default socket bindings

Name	Port	Multicast Port	Description	full-ha-sockets	full-sockets	ha-socket	standard-socket
ajp	8009		Apache JServ Protocol. Used for HTTP clustering and load balancing.	Yes	Yes	Yes	Yes
http	8080		The default port for deployed web applications.	Yes	Yes	Yes	Yes
https	8443		SSL-encrypted connection between deployed web applications and clients.	Yes	Yes	Yes	Yes
jacorb	3528		CORBA services for JTS transactions and other ORB-dependent services.	Yes	Yes	No	No
jacorb-ssl	3529		SSL-encrypted CORBA services.	Yes	Yes	No	No
jgroups-diagnostics		7500	Multicast. Used for peer discovery in HA clusters. Not configurable using the Management Interfaces.	Yes	No	Yes	No
jgroups-mping		45700	Multicast. Used to discover initial membership in a HA cluster.	Yes	No	Yes	No
jgroups-tcp	7600		Unicast peer discovery in HA clusters using TCP.	Yes	No	Yes	No
jgroups-tcp-fd	57600		Used for HA failure detection over TCP.	Yes	No	Yes	No
jgroups-udp	55200	45688	Multicast peer discovery in HA clusters using UDP.	Yes	No	Yes	No

Name	Port	Multicast Port	Description	full-ha-sockets	full-sockets	ha-socket	standard-socket
jgroups-udp-fd	54200		Used for HA failure detection over UDP.	Yes	No	Yes	No
messaging	5445		JMS service.	Yes	Yes	No	No
messaging-group	0	9876	Referenced by HornetQ JMS broadcast and discovery groups.	Yes	Yes	No	No
messaging-throughput	5455		Used by JMS Remoting.	Yes	Yes	No	No
mod_cluster		23364	Multicast port for communication between JBoss EAP 6 and the HTTP load balancer.	Yes	No	Yes	No
remoting	4447		Used for remote EJB invocation.	Yes	Yes	Yes	Yes
txn-recovery-environment	4712		The JTA transaction recovery manager.	Yes	Yes	Yes	Yes
txn-status-manager	4713		The JTA / JTS transaction manager.	Yes	Yes	Yes	Yes

Management Ports

In addition to the socket binding groups, each host controller opens two more ports for management purposes:

- **9990** - The Web Management Console port
- **9999** - The port used by the Management Console and Management API

Additionally, if HTTPS is enabled for the Management Console, 9443 is also opened as the default port.

[Report a bug](#)

5.2.4. About Port Offsets for Socket Binding Groups

Port offsets are a numeric offset added to the port values given by the socket binding group for that server. This allows a single server to inherit the socket bindings of the server group that it belongs to, with an offset to ensure that it does not clash with the other servers in the group. For instance, if the HTTP port of the socket binding group is 8080, and your server uses a port offset of 100, its HTTP port is 8180.

[Report a bug](#)

5.2.5. Configure Port Offsets

- **Configure Port Offsets**

Choose either the Management CLI or the Management Console to configure your port offsets.

- **Configure Port Offsets Using the Management CLI**

Use the Management CLI to configure port offsets.

- a. **Edit Port Offsets**

Use the **write-attribute** operation to write a new value to the port offset attribute. The following example updates the **socket-binding-port-offset** value of *server-two* to 250. This server is a member of the default local host group. A restart is required for the changes to take effect.

```
[domain@localhost:9999 /] /host=master/server-config=server-two/:write-attribute(name=socket-binding-port-offset,value=250)
```

- b. **Confirm Port Offset Attributes**

Confirm the values are changed by running the **read-resource** operation with the **include-runtime=true** parameter to expose all current values active in the server model.

```
[domain@localhost:9999 /] /host=master/server-config=server-two/:read-resource(include-runtime=true)
```

- **Configure Port Offsets Using the Management Console**

Use the Management Console to configure port offsets.

- a. **Log into the Management Console.**

Log into the Management Console of your Managed Domain.

- b. **Select the Domain tab**

Select the **Domain** tab at the top of the screen.

- c. **Edit Port Offset Attributes**

- i. Select the server under the **Available Server Configurations** list and click **Edit** at the top of the attributes list below.

- ii. Enter any desired values in the **Port Offset** field.

- iii. Click **Save** to finish.

[Report a bug](#)

5.3. IPV6

5.3.1. Configure JVM Stack Preferences for IPv6 Networking

Summary

This topic covers enabling IPv6 networking for the JBoss EAP 6 installation.

Procedure 5.1. Disable the IPv4 Stack Java Property

1. Open the relevant file for the installation:
 - **For a Standalone Server:**
Open ***EAP_HOME/bin/standalone.conf***.
 - **For a Managed Domain:**
Open ***EAP_HOME/bin/domain.conf***.
2. Change the IPv4 Stack Java property to false:

```
-Djava.net.preferIPv4Stack=false
```

For example:

Example 5.8. JVM options

```
# Specify options to pass to the Java VM.
#
if [ "$JAVA_OPTS" = "x" ]; then
  JAVA_OPTS="-Xms64m -Xmx512m -XX:MaxPermSize=256m -
Djava.net.preferIPv4Stack=false
-Dorg.jboss.resolver.warning=true -Dsun.rmi.dgc.client.gcInterval=3600000
-Dsun.rmi.dgc.server.gcInterval=3600000 -Djava.net.preferIPv6Addresses=true"
fi
```

[Report a bug](#)

5.3.2. Configure the Interface Declarations for IPv6 Networking

Summary

Follow these steps to configure the interface inet address to the IPv6 default:

Prerequisites

- [Section 2.2.1, "Start JBoss EAP 6"](#)
- [Section 3.3.2, "Log in to the Management Console"](#)

Procedure 5.2. Configure the Interface for IPv6 Networking

1. Select the **Configuration** tab at the top of the screen.

2. Expand the **General Configuration** menu and select **Interfaces**.
3. Select the interface from the **Available Interfaces** list.
4. Click **Edit** in the detail list.
5. Set the inet address to:

```

|  ${jboss.bind.address.management:[ADDRESS]}

```

6. Click **Save** to finish.
7. Restart the server to implement the changes.

[Report a bug](#)

5.3.3. Configure JVM Stack Preferences for IPv6 Addresses

Summary

This topic covers configuring the JBoss EAP 6 installation to prefer IPv6 addresses through the configuration files.

Procedure 5.3. Configure the JBoss EAP 6 Installation to Prefer IPv6 Addresses

1. Open the relevant file for the installation:
 - **For a Standalone Server:**
Open ***EAP_HOME/bin/standalone.conf***.
 - **For a Managed Domain:**
Open ***EAP_HOME/bin/domain.conf***.
2. Append the following Java property to the Java VM options:

```

| -Djava.net.preferIPv6Addresses=true

```

For example:

Example 5.9. JVM options

```

| # Specify options to pass to the Java VM.
| #
| if [ "x$JAVA_OPTS" = "x" ]; then
|   JAVA_OPTS="-Xms64m -Xmx512m -XX:MaxPermSize=256m -
|   Djava.net.preferIPv4Stack=false
|   -Dorg.jboss.resolver.warning=true -Dsun.rmi.dgc.client.gcInterval=3600000
|   -Dsun.rmi.dgc.server.gcInterval=3600000 -Djava.net.preferIPv6Addresses=true"
| fi

```

[Report a bug](#)

5.4. REMOTING

5.4.1. Configuration of Message Size in Remoting

The remoting subsystem provides the option to limit the size of the messages for remoting protocols. You can set the maximum inbound message size (**MAX_INBOUND_MESSAGE_SIZE**) and the maximum outbound message size (**MAX_OUTBOUND_MESSAGE_SIZE**) to ensure that messages are received and sent within appropriate size limits.

The **MAX_INBOUND_MESSAGE_SIZE** and **MAX_OUTBOUND_MESSAGE_SIZE** can be set in `ejb3` subsystem, and the value is in byte.

```
<subsystem xmlns="urn:jboss:domain:ejb3:1.5">
.....
<remote connector-ref="remoting-connector" thread-pool-name="default">
  <channel-creation-options>
    <option name="MAX_INBOUND_MESSAGE_SIZE" value="xxxxx" type="remoting"/>
    <option name="MAX_OUTBOUND_MESSAGE_SIZE" value="xxxxx" type="remoting"/>
  </channel-creation-options>
</remote>
.....
</subsystem>
```

Configuring the size of messages in remoting protocols helps in effective utilization of system memory and prevents it from reaching an out of memory state while performing important operations.

If the sender sends a message which exceeds the maximum allowable limit (**MAX_OUTBOUND_MESSAGE_SIZE**), the server throws an exception and cancels the transmission of data. However the connection remains open and the sender can choose to close the message if needed.

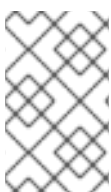
If a message received exceeds the maximum allowable limit (**MAX_INBOUND_MESSAGE_SIZE**) the message is closed asynchronously with the connection still open.

[Report a bug](#)

5.4.2. Configure the Remoting Subsystem

Overview

JBoss Remoting has three top-level configurable elements: the worker thread pool, one or more connectors, and a series of local and remote connection URLs. This topic presents an explanation of each configurable item, example CLI commands for how to configure each item, and an XML example of a fully-configured subsystem. This configuration only applies to the server. Most people will not need to configure the Remoting subsystem at all, unless they use custom connectors for their own applications. Applications which act as Remoting clients, such as EJBs, need separate configuration to connect to a specific connector.



NOTE

The Remoting subsystem configuration is not exposed to the web-based Management Console, but it is fully configurable from the command-line based Management CLI. Editing the XML by hand is not recommended.

Adapting the CLI Commands

The CLI commands are formulated for a managed domain, when configuring the **default** profile. To configure a different profile, substitute its name. For a standalone server, omit the **/profile=default** part of the command.

Configuration Outside the Remoting Subsystem

There are a few configuration aspects which are outside of the **remoting** subsystem:

Network Interface

The network interface used by the **remoting** subsystem is the **public** interface defined in the **domain/configuration/domain.xml** or **standalone/configuration/standalone.xml**.

```
<interfaces>
  <interface name="management"/>
  <interface name="public"/>
  <interface name="unsecure"/>
</interfaces>
```

The per-host definition of the **public** interface is defined in the **host.xml** in the same directory as the **domain.xml** or **standalone.xml**. This interface is also used by several other subsystems. Exercise caution when modifying it.

```
<interfaces>
  <interface name="management">
    <inet-address value="{jboss.bind.address.management:127.0.0.1}"/>
  </interface>
  <interface name="public">
    <inet-address value="{jboss.bind.address:127.0.0.1}"/>
  </interface>
  <interface name="unsecure">
    <!-- Used for IIOP sockets in the standard configuration.
         To secure JacORB you need to setup SSL -->
    <inet-address value="{jboss.bind.address.unsecure:127.0.0.1}"/>
  </interface>
</interfaces>
```

socket-binding

The default socket-binding used by the **remoting** subsystem binds to TCP port 4447. Refer to the documentation about socket bindings and socket binding groups for more information if you need to change this.

Information about socket binding and socket binding groups can be found in the *Socket Binding Groups* chapter of JBoss EAP's *Administration and Configuration Guide* available at https://access.redhat.com/documentation/en-us/red_hat_jboss_enterprise_application_platform/?version=6.4

Remoting Connector Reference for EJB

The EJB subsystem contains a reference to the remoting connector for remote method invocations. The following is the default configuration:

```
<remote-connector-ref="remoting-connector" thread-pool-name="default"/>
```

Secure Transport Configuration

Remoting transports use StartTLS to use a secure (HTTPS, Secure Servlet, etc) connection if the client requests it. The same socket binding (network port) is used for secured and unsecured connections, so no additional server-side configuration is necessary. The client requests the secure or unsecured transport, as its needs dictate. JBoss EAP 6 components which use Remoting, such as EJBs, the ORB, and the JMS provider, request secured interfaces by default.



WARNING

StartTLS works by activating a secure connection if the client requests it, and otherwise defaulting to an unsecured connection. It is inherently susceptible to a *Man in the Middle* style exploit, wherein an attacker intercepts the client's request and modifies it to request an unsecured connection. Clients must be written to fail appropriately if they do not receive a secure connection, unless an unsecured connection actually is an appropriate fall-back.

Worker Thread Pool

The worker thread pool is the group of threads which are available to process work which comes in through the Remoting connectors. It is a single element `<worker-thread-pool>`, and takes several attributes. Tune these attributes if you get network timeouts, run out of threads, or need to limit memory usage. Specific recommendations depend on your specific situation. Contact Red Hat Global Support Services for more information.

Table 5.4. Worker Thread Pool Attributes

Attribute	Description	CLI Command
read-threads	The number of read threads to create for the remoting worker. Defaults to 1 .	<code>/profile=default/subsystem=remoting:/write-attribute(name=worker-read-threads,value=1)</code>
write-threads	The number of write threads to create for the remoting worker. Defaults to 1 .	<code>/profile=default/subsystem=remoting:/write-attribute(name=worker-write-threads,value=1)</code>
task-keepalive	The number of milliseconds to keep non-core remoting worker task threads alive. Defaults to 60 .	<code>/profile=default/subsystem=remoting:/write-attribute(name=worker-task-keepalive,value=60)</code>
task-max-threads	The maximum number of threads for the remoting worker task thread pool. Defaults to 16 .	<code>/profile=default/subsystem=remoting:/write-attribute(name=worker-task-max-threads,value=16)</code>

Attribute	Description	CLI Command
task-core-threads	The number of core threads for the remoting worker task thread pool. Defaults to 4 .	/profile=default/subsystem=remoting/:write-attribute(name=worker-task-core-threads,value=4)
task-limit	The maximum number of remoting worker tasks to allow before rejecting. Defaults to 16384 .	/profile=default/subsystem=remoting/:write-attribute(name=worker-task-limit,value=16384)

Connector

The connector is the main Remoting configuration element. Multiple connectors are allowed. Each consists of a element **<connector>** element with several sub-elements, as well as a few possible attributes. The default connector is used by several subsystems of JBoss EAP 6. Specific settings for the elements and attributes of your custom connectors depend on your applications, so contact Red Hat Global Support Services for more information.

Table 5.5. Connector Attributes

Attribute	Description	CLI Command
socket-binding	The name of the socket binding to use for this connector.	/profile=default/subsystem=remoting/connector=remoting-connector/:write-attribute(name=socket-binding,value=remoting)
authentication-provider	The Java Authentication Service Provider Interface for Containers (JASPIC) module to use with this connector. The module must be in the classpath.	/profile=default/subsystem=remoting/connector=remoting-connector/:write-attribute(name=authentication-provider,value=myProvider)
security-realm	Optional. The security realm which contains your application's users, passwords, and roles. An EJB or Web Application can authenticate against a security realm. ApplicationRealm is available in a default JBoss EAP 6 installation.	/profile=default/subsystem=remoting/connector=remoting-connector/:write-attribute(name=security-realm,value=ApplicationRealm)

Table 5.6. Connector Elements

Attribute	Description	CLI Command
sasl	Enclosing element for Simple Authentication and Security Layer (SASL) authentication mechanisms	N/A

Attribute	Description	CLI Command
properties	Contains one or more <property> elements, each with a name attribute and an optional value attribute.	/profile=default/subsystem=remoting/connector=remoting - connector/property=myProp: add(value=myPropValue)

Outbound Connections

You can specify three different types of outbound connection:

- Outbound connection to a URI.
- Local outbound connection – connects to a local resource such as a socket.
- Remote outbound connection – connects to a remote resource and authenticates using a security realm.

All of the outbound connections are enclosed in an **<outbound-connections>** element. Each of these connection types takes an **outbound-socket-binding-ref** attribute. The outbound-connection takes a **uri** attribute. The remote outbound connection takes optional **username** and **security-realm** attributes to use for authorization.

Table 5.7. Outbound Connection Elements

Attribute	Description	CLI Command
outbound-connection	Generic outbound connection.	/profile=default/subsystem=remoting/outbound-connection=my-connection/:add(uri=http://my-connection)
local-outbound-connection	Outbound connection with a implicit local:// URI scheme.	/profile=default/subsystem=remoting/local-outbound-connection=my-connection/:add(outbound-socket-binding-ref=remoting2)
remote-outbound-connection	Outbound connections for remote:// URI scheme, using basic/digest authentication with a security realm.	/profile=default/subsystem=remoting/remote-outbound-connection=my-connection/:add(outbound-socket-binding-ref=remoting,username=myUser,security-realm=ApplicationRealm)

SASL Elements

Before defining the SASL child elements, you need to create the initial SASL element. Use the following command:

```
/profile=default/subsystem=remoting/connector=remoting-connector/security=sasl:add
```

The child elements of the SASL element are described in the table below.

Table 5.8. SASL child elements

Attribute	Description	CLI Command
include-mechanisms	Contains a value attribute, which is a list of SASL mechanisms.	<pre>/profile=default/subsystem=r emoting/connector=remoting - connector/security=sasl:write -attribute(name=include- mechanisms,value= ["DIGEST","PLAIN","GSSA PI"])</pre>
qop	Contains a value attribute, which is a list of SASL Quality of protection values, in decreasing order of preference.	<pre>/profile=default/subsystem=r emoting/connector=remoting - connector/security=sasl:write -attribute(name=qop,value= ["auth"])</pre>
strength	Contains a value attribute, which is a list of SASL cipher strength values, in decreasing order of preference.	<pre>/profile=default/subsystem=r emoting/connector=remoting - connector/security=sasl:write - attribute(name=strength,valu e=["medium"])</pre>
reuse-session	Contains a value attribute which is a boolean value. If true, attempt to reuse sessions.	<pre>/profile=default/subsystem=r emoting/connector=remoting - connector/security=sasl:write -attribute(name=reuse- session,value=false)</pre>
server-auth	Contains a value attribute which is a boolean value. If true, the server authenticates to the client.	<pre>/profile=default/subsystem=r emoting/connector=remoting - connector/security=sasl:write -attribute(name=server- auth,value=false)</pre>

policy Attribute	Description	CLI Command
	<p>An enclosing element which contains zero or more of the following elements, which each take a single value.</p> <ul style="list-style-type: none"> • forward-secrecy – whether mechanisms are required to implement forward secrecy (breaking into one session will not automatically provide information for breaking into future sessions) • no-active – whether mechanisms susceptible to non-dictionary attacks are permitted. A value of false permits, and true denies. • no-anonymous – whether mechanisms that accept anonymous login are permitted. A value of false permits, and true denies. • no-dictionary – whether mechanisms susceptible to passive dictionary attacks are allowed. A value of false permits, and true denies. • no-plain-text – whether mechanisms which are susceptible to simple plain passive attacks are allowed. A value of false permits, and true denies. • pass-credentials – whether mechanisms which pass client credentials are allowed. 	<pre> /profile=default/subsystem=remoting/connector=remoting - connector/security=sasl/sasl-policy=policy:add /profile=default/subsystem=remoting/connector=remoting - connector/security=sasl/sasl-policy=policy:write-attribute(name=forward-secrecy,value=true) /profile=default/subsystem=remoting/connector=remoting - connector/security=sasl/sasl-policy=policy:write-attribute(name=no-active,value=false) /profile=default/subsystem=remoting/connector=remoting - connector/security=sasl/sasl-policy=policy:write-attribute(name=no-anonymous,value=false) /profile=default/subsystem=remoting/connector=remoting - connector/security=sasl/sasl-policy=policy:write-attribute(name=no-dictionary,value=true) /profile=default/subsystem=remoting/connector=remoting - connector/security=sasl/sasl-policy=policy:write-attribute(name=no-plain-text,value=false) /profile=default/subsystem=remoting/connector=remoting - connector/security=sasl/sasl-policy=policy:write-attribute(name=pass-credentials,value=true) </pre>

Attribute properties	Description Contains one or more	CLI Command
	<p><property> elements, each with a name attribute and an optional value attribute.</p>	<pre data-bbox="1034 197 1441 607">/profile=default/subsystem=remoting/connector=remoting - connector/security=sasl/property=myprop:add(value=1) /profile=default/subsystem=remoting/connector=remoting - connector/security=sasl/property=myprop2:add(value=2)</pre>

Example 5.10. Example Configurations

This example shows the default remoting subsystem that ships with JBoss EAP 6.

```
<subsystem xmlns="urn:jboss:domain:remoting:1.1">
  <connector name="remoting-connector" socket-binding="remoting" security-
  realm="ApplicationRealm"/>
</subsystem>
```

This example contains many hypothetical values, and is presented to put the elements and attributes discussed previously into context.

```
<subsystem xmlns="urn:jboss:domain:remoting:1.1">
  <worker-thread-pool read-threads="1" task-keepalive="60" task-max-threads="16" task-core-
  thread="4" task-limit="16384" write-threads="1" />
  <connector name="remoting-connector" socket-binding="remoting" security-
  realm="ApplicationRealm">
    <sasl>
      <include-mechanisms value="GSSAPI PLAIN DIGEST-MD5" />
      <qop value="auth" />
      <strength value="medium" />
      <reuse-session value="false" />
      <server-auth value="false" />
      <policy>
        <forward-secrecy value="true" />
        <no-active value="false" />
        <no-anonymous value="false" />
        <no-dictionary value="true" />
        <no-plain-text value="false" />
        <pass-credentials value="true" />
      </policy>
      <properties>
        <property name="myprop1" value="1" />
        <property name="myprop2" value="2" />
      </properties>
    </sasl>
  <authentication-provider name="myprovider" />
</properties>
```



```
<property name="myprop3" value="propValue" />
</properties>
</connector>
<outbound-connections>
  <outbound-connection name="my-outbound-connection" uri="http://myhost:7777"/>
  <remote-outbound-connection name="my-remote-connection" outbound-socket-binding-
ref="my-remote-socket" username="myUser" security-realm="ApplicationRealm"/>
  <local-outbound-connection name="myLocalConnection" outbound-socket-binding-ref="my-
outbound-socket"/>
</outbound-connections>
</subsystem>
```

Configuration Aspects Not Yet Documented

- JNDI and Multicast Automatic Detection

[Report a bug](#)

CHAPTER 6. DATASOURCE MANAGEMENT

6.1. INTRODUCTION

6.1.1. About JDBC

The JDBC API is the standard that defines how databases are accessed by Java applications. An application configures a datasource that references a JDBC driver. Application code can then be written against the driver, rather than the database. The driver converts the code to the database language. This means that if the correct driver is installed, an application can be used with any supported database.

The JDBC 4.0 specification is defined here: <http://jcp.org/en/jsr/detail?id=221>.

Procedure 6.1. Test datasource pools connection using Management Console

1. [Section 3.3.2, “Log in to the Management Console”](#) .
2. Select the **Runtime** tab from the top of the Management Console.
3. Expand the **Subsystems** menu on the left, and select **Datasources**.
4. Click the **Test Connection** button to test the connection to the datasource and verify the settings are correct.

Procedure 6.2. Test datasource connection using Management CLI

1. Launch the CLI tool and connect to your server.
2. Run the following Management CLI command to test the connection:
 - In Standalone mode, enter the following command:

```
/subsystem=datasources/data-source=ExampleDS:test-connection-in-pool
```

- In Domain mode, enter the following command:

```
host=master/server=server-one/subsystem=datasources/data-source=ExampleDS:test-connection-in-pool
```

To get started with JDBC and datasources, refer to the JDBC Driver section of the Administration and Configuration Guide for JBoss EAP 6.

[Report a bug](#)

6.1.2. JBoss EAP 6 Supported Databases

The list of JDBC compliant databases supported by JBoss EAP 6 is available here: <https://access.redhat.com/articles/111663>.

[Report a bug](#)

6.1.3. Types of Datasources

The two general types of resources are referred to as **datasources** and **XA datasources**.

Non-XA datasources are used for applications which do not use transactions, or applications which use transactions with a single database.

XA datasources are used by applications whose transactions are distributed across multiple databases. XA datasources introduce additional overhead.

You specify the type of your datasource when you create it in the Management Console or Management CLI.

[Report a bug](#)

6.1.4. The Example Datasource

JBoss EAP 6 includes a H2 database. It is a lightweight, relational database management system that provides developers with the ability to quickly build applications, and is the example datasource for the platform.



WARNING

However, it should not be used in a production environment. It is a very small, self-contained datasource that supports all of the standards needed for testing and building applications, but is not robust or scalable enough for production use.

For a list of supported and certified datasources, refer here: [Section 6.1.2, "JBoss EAP 6 Supported Databases"](#).

[Report a bug](#)

6.1.5. Deployment of `-ds.xml` files

In JBoss EAP 6, datasources are defined as a resource of the server subsystem. In previous versions, a ***-ds.xml** datasource configuration file was required in the deployment directory of the server configuration. ***-ds.xml** files can still be deployed in JBoss EAP 6, following the 1.1 data sources schema available under *Schemas* here: <http://www.ironjacamar.org/documentation.html>.



WARNING

This feature should only be used for development. It is not recommended for production environments because it is not supported by the JBoss administrative and management tools. This feature is deprecated in JBoss EAP 6.4 and will not be supported in the next major release of the product.



IMPORTANT

It is mandatory to use a reference to an already deployed / defined <driver> entry when deploying ***-ds.xml** files.

[Report a bug](#)

6.2. JDBC DRIVERS

6.2.1. Install a JDBC Driver with the Management Console

Summary

Before your application can connect to a JDBC datasource, your datasource vendor's JDBC drivers need to be installed in a location where JBoss EAP 6 can use them. JBoss EAP 6 allows you to deploy these drivers like any other deployment. This means that you can deploy them across multiple servers in a server group, if you use a managed domain.

Prerequisites

Before performing this task, you need to meet the following prerequisites:

- Download the JDBC driver from your database vendor.



NOTE

Any JDBC 4-compliant driver is automatically recognized and installed in the system by name and version. A JDBC JAR is identified using the Java service provider mechanism. Such JARs contain the META-INF/services/java.sql.Driver text, which contains the name of the Driver classes in that JAR.

Procedure 6.3. Modify the JDBC Driver JAR

If the JDBC driver JAR is not JDBC 4-compliant, it can be made deployable using the following method.

1. Change to, or create, an empty temporary directory.
2. Create a META-INF subdirectory.
3. Create a META-INF/services subdirectory.
4. Create a META-INF/services/java.sql.Driver file, which contains one line indicating the fully-qualified class name of the JDBC driver.
5. Use the JAR command-line tool to update the JAR like this:

```
jar -uf jdbc-driver.jar META-INF/services/java.sql.Driver
```

6. [Section 3.3.2, "Log in to the Management Console"](#)
7. If you use a managed domain, deploy the JAR file to a server group. Otherwise, deploy it to your server. See [Section 10.2.2, "Enable a Deployed Application Using the Management Console"](#).

Result:

The JDBC driver is deployed, and is available for your applications to use.

[Report a bug](#)

6.2.2. Install a JDBC Driver as a Core Module

Prerequisites

Before performing this task, you need to meet the following prerequisites:

- Download the JDBC driver from your database vendor. JDBC driver download locations are listed here: [Section 6.2.3, “JDBC Driver Download Locations”](#).
- Extract the archive.

Procedure 6.4. Install a JDBC Driver as a Core Module Using the Management CLI

1. Start the Server.
2. Start the Management CLI, but do not use the **--connect** or **-c** argument to connect to the running instance.
3. Use the **module add** CLI command to add the new module.

Example 6.1. Example CLI command to add a MySQL JDBC driver module

```
module add --name=com.mysql --resources=/path/to/mysql.jar --
dependencies=javax.api,javax.transaction.api
```

NOTE

Using the **module** management CLI command to add and remove modules is provided as [technology preview](#) only, and should not be used to add modules to a remote JBoss EAP instance. Modules should be added and removed manually in a production environment.

Perform the following steps to add a module manually.

1. A file path structure will be created under the **EAP_HOME/modules/** directory. For example, for a MySQL JDBC driver, the following will be created: **EAP_HOME/modules/com/mysql/main/**.
2. The JAR files specified as resources will be copied to the **main/** subdirectory.
3. A **module.xml** file with the specified dependencies will be created in the **main/** subdirectory. See [Section 7.1.1, “Modules”](#) for an example of a **module.xml** file.

Execute **module --help** for more details on using this command to add and remove modules.

4. Use the **connect** CLI command to connect to the running instance.
5. Run the CLI command to add the JDBC driver module to the server configuration.

The command you choose depends on the number of classes listed in the **/META-INF/services/java.sql.Driver** file located in the JDBC driver JAR. For example, the **/META-INF/services/java.sql.Driver** file in the MySQL 5.1.20 JDBC JAR lists two classes:

- **com.mysql.jdbc.Driver**
- **com.mysql.fabric.jdbc.FabricMySQLDriver**

When there is more than one entry, you must also specify the name of the driver class. Failure to do so results in an error similar to the following:

Example 6.2. Driver class error

```
JBAS014749: Operation handler failed: Service jboss.jdbc-driver.mysql is already registered
```

- Run the CLI command for JDBC JARs containing one class entry.

```
/subsystem=datasources/jdbc-driver=DRIVER_NAME:add(driver-name=DRIVER_NAME,driver-module-name=MODULE_NAME,driver-xa-datasource-class-name=XA_DATASOURCE_CLASS_NAME)
```

Example 6.3. CLI Command for Standalone Mode for JDBC JARs with one driver class

```
/subsystem=datasources/jdbc-driver=mysql:add(driver-name=mysql,driver-module-name=com.mysql,driver-xa-datasource-class-name=com.mysql.jdbc.jdbc2.optional.MysqlXADataSource)
```

Example 6.4. CLI Command for Domain Mode for JDBC JARs with one driver class

```
/profile=ha/subsystem=datasources/jdbc-driver=mysql:add(driver-name=mysql,driver-module-name=com.mysql,driver-xa-datasource-class-name=com.mysql.jdbc.jdbc2.optional.MysqlXADataSource)
```

- Run the CLI command for JDBC JARs containing multiple class entries.

```
/subsystem=datasources/jdbc-driver=DRIVER_NAME:add(driver-name=DRIVER_NAME,driver-module-name=MODULE_NAME,driver-xa-datasource-class-name=XA_DATASOURCE_CLASS_NAME, driver-class-name=DRIVER_CLASS_NAME)
```

Example 6.5. CLI Command for Standalone Mode for JDBC JARs with multiple driver class entries

```
/subsystem=datasources/jdbc-driver=mysql:add(driver-name=mysql,driver-module-name=com.mysql,driver-xa-datasource-class-name=com.mysql.jdbc.jdbc2.optional.MysqlXADataSource, driver-class-name=com.mysql.jdbc.Driver)
```

Example 6.6. CLI Command for Domain Mode for JDBC JARs with multiple driver class entries

```
/profile=ha/subsystem=datasources/jdbc-driver=mysql:add(driver-name=mysql,driver-
module-name=com.mysql,driver-xa-datasource-class-
name=com.mysql.jdbc.jdbc2.optional.MysqlXADataSource, driver-class-
name=com.mysql.jdbc.Driver)
```

Example 6.7. CLI Command for Domain Mode for JDBC JARs with multiple non-XA driver class entries

```
/profile=ha/subsystem=datasources/jdbc-driver=oracle/:add(driver-module-
name=com.oracle,driver-name=oracle,jdbc-compliant=true,driver-datasource-class-
name=oracle.jdbc.OracleDriver)
```

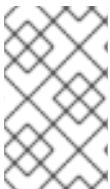
Result

The JDBC driver is now installed and set up as a core module, and is available to be referenced by application datasources.

[Report a bug](#)

6.2.3. JDBC Driver Download Locations

The following table gives the standard download locations for JDBC drivers of common databases used with JBoss EAP 6.

**NOTE**

These links point to third-party websites which are not controlled or actively monitored by Red Hat. For the most up-to-date drivers for your database, check your database vendor's documentation and website.

Table 6.1. JDBC driver download locations

Vendor	Download Location
MySQL	http://www.mysql.com/products/connector/
PostgreSQL	http://jdbc.postgresql.org/
Oracle	http://www.oracle.com/technetwork/database/features/jdbc/index-091264.html
IBM	http://www-306.ibm.com/software/data/db2/java/
Sybase	http://www.sybase.com/products/allproductsa-z/softwaredeveloperkit/jconnect

Vendor	Download Location
Microsoft	http://msdn.microsoft.com/data/jdbc/

[Report a bug](#)

6.2.4. Access Vendor Specific Classes

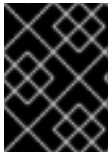
Summary

This topic covers the steps required to use the JDBC specific classes. This is necessary when an application needs to use vendor specific functionality that is not part of the JDBC API.



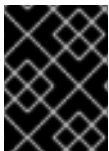
WARNING

This is advanced usage. Only applications that need functionality not found in the JDBC API should implement this procedure.



IMPORTANT

This process is required when using the reauthentication mechanism, and accessing vendor specific classes.



IMPORTANT

Follow the vendor specific API guidelines closely, as the connection is being controlled by the IronJacamar container.

Prerequisites

- [Section 6.2.2, "Install a JDBC Driver as a Core Module"](#) .

Procedure 6.5. Add a Dependency to the Application

You can add a dependency to an application using either of the following methods. Choose whichever method you prefer.

- ◦ **Configure the MANIFEST.MF file**
 - a. Open the application's **META-INF/MANIFEST.MF** file in a text editor.
 - b. Add a dependency for the JDBC module and save the file.

Dependencies: *MODULE_NAME*

Example 6.8. MANIFEST.MF file with `com.mysql` declared as a dependency

Dependencies: com.mysql

- o a. **Create a `jboss-deployment-structure.xml` file**
Create a file called **`jboss-deployment-structure.xml`** in the **`META-INF/`** or **`WEB-INF`** folder of the application.

Example 6.9. `jboss-deployment-structure.xml` file with `com.mysql` declared as a dependency

```
<jboss-deployment-structure>
  <deployment>
    <dependencies>
      <module name="com.mysql" />
    </dependencies>
  </deployment>
</jboss-deployment-structure>
```

Example 6.10. Access the Vendor Specific API

The example below accesses the MySQL API.

```
import java.sql.Connection;
import org.jboss.jca.adapters.jdbc.WrappedConnection;

Connection c = ds.getConnection();
WrappedConnection wc = (WrappedConnection)c;
com.mysql.jdbc.Connection mc = wc.getUnderlyingConnection();
```

[Report a bug](#)

6.3. NON-XA DATASOURCES

6.3.1. Create a Non-XA Datasource with the Management Interfaces

Summary

This topic covers the steps required to create a non-XA datasource, using either the Management Console or the Management CLI.

Prerequisites

- The JBoss EAP 6 server must be running.



NOTE

Prior to version 10.2 of the Oracle datasource, the `<no-tx-separate-pools/>` parameter was required, as mixing non-transactional and transactional connections would result in an error. This parameter may no longer be required for certain applications.

**NOTE**

To prevent issues such as duplication of driver listing, selected driver not available in a profile, or driver not displayed if a server for the profile is not running, in JBoss EAP 6.4 onwards, only JDBC drivers that are installed as modules and correctly referenced from profiles are detectable while creating a datasource using the Management Console in domain mode.

Procedure 6.6. Create a Datasource using either the Management CLI or the Management Console

- ○ **Management CLI**

- a. Launch the CLI tool and connect to your server.
- b. Run the following Management CLI command to create a non-XA datasource, configuring the variables as appropriate:

**NOTE**

The value for *DRIVER_NAME* depends on the number of classes listed in the **/META-INF/services/java.sql.Driver** file located in the JDBC driver JAR. If there is only one class, the value is the name of the JAR. If there are multiple classes, the value is the name of the JAR + *driverClassName* + "_" + *majorVersion* + "_" + *minorVersion*. Failure to do so will result in the following error being logged:

```
JBAS014775: New missing/unsatisfied dependencies
```

For example, the *DRIVER_NAME* value required for the MySQL 5.1.31 driver, is **mysql-connector-java-5.1.31-bin.jarcom.mysql.jdbc.Driver_5_1**.

```
data-source add --name=DATASOURCE_NAME --jndi-name=JNDI_NAME --driver-name=DRIVER_NAME --connection-url=CONNECTION_URL
```

- c. Enable the datasource:

```
data-source enable --name=DATASOURCE_NAME
```

- ○ **Management Console**

- a. Login to the Management Console.
- b. **Navigate to the Datasources panel in the Management Console**
 - i. Select the **Configuration** tab from the top of the console.
 - ii. For Domain mode only, select a profile from the drop-down box in the top left.
 - iii. Expand the **Subsystems** menu on the left of the console, then expand the **Connector** menu.
 - iv. Select **Datasources** from the menu on the left of the console.

c. Create a new datasource

- i. Click **Add** at the top of the **Datasources** panel.
- ii. Enter the new datasource attributes in the **Create Datasource** wizard and proceed with the **Next** button.
- iii. Enter the JDBC driver details in the **Create Datasource** wizard and click **Next** to continue.
- iv. Enter the connection settings in the **Create Datasource** wizard.
- v. Click the **Test Connection** button to test the connection to the datasource and verify the settings are correct.
- vi. Click **Done** to finish

Result

The non-XA datasource has been added to the server. It is now visible in either the **standalone.xml** or **domain.xml** file, as well as the management interfaces.

[Report a bug](#)

6.3.2. Modify a Non-XA Datasource with the Management Interfaces

Summary

This topic covers the steps required to modify a non-XA datasource, using either the Management Console or the Management CLI.

Prerequisites

- [Section 2.2.1, “Start JBoss EAP 6”](#).



NOTE

Non-XA datasources can be integrated with JTA transactions. To integrate the datasource with JTA, ensure that the **jta** parameter is set to **true**.

Procedure 6.7. Modify a Non-XA Datasource

- ○ **Management CLI**
 - a. [Section 3.4.2, “Launch the Management CLI”](#).
 - b. Use the **write-attribute** command to configure a datasource attribute:

```
/subsystem=datasources/data-source=DATASOURCE_NAME:write-attribute(name=ATTRIBUTE_NAME,value=ATTRIBUTE_VALUE)
```

- c. Reload the server to confirm the changes:

```
reload
```

- **Management Console**

- a. [Section 3.3.2, “Log in to the Management Console”](#) .
- b. **Navigate to the Datasources panel in the Management Console**
 - i. Select the **Configuration** tab from the top of the console.
 - ii. For Domain mode only, select a profile from the drop-down box in the top left.
 - iii. Expand the **Subsystems** menu on the left of the console, then expand the **Connector** menu.
 - iv. Select **Datasources** from expanded menu.
- c. **Edit the datasource**
 - i. Select a datasource from the **Available Datasources** list. The datasource attributes are displayed below.
 - ii. Click **Edit** to edit the datasource attributes.
 - iii. Click **Save** to finish.

Result

The non-XA datasource has been configured. The changes are now visible in either the **standalone.xml** or **domain.xml** file, as well as the management interfaces.

- To create a new datasource, refer here: [Section 6.3.1, “Create a Non-XA Datasource with the Management Interfaces”](#).
- To remove the datasource, refer here: [Section 6.3.3, “Remove a Non-XA Datasource with the Management Interfaces”](#).

[Report a bug](#)

6.3.3. Remove a Non-XA Datasource with the Management Interfaces

Summary

This topic covers the steps required to remove a non-XA datasource from JBoss EAP 6, using either the Management Console or the Management CLI.

Prerequisites

- [Section 2.2.1, “Start JBoss EAP 6”](#) .

Procedure 6.8. Remove a Non-XA Datasource

- - **Management CLI**
 - a. [Section 3.4.2, “Launch the Management CLI”](#) .
 - b. Run the following command to remove a non-XA datasource:

```
data-source remove --name=DATASOURCE_NAME
```

- - **Management Console**

- a. [Section 3.3.2, “Log in to the Management Console”](#) .
- b. **Navigate to the Datasources panel in the Management Console**
 - i. Select the **Configuration** tab from the top of the console.
 - ii. For Domain mode only, select a profile from the drop-down box in the top left.
 - iii. Expand the **Subsystems** menu on the left of the console, then expand the **Connector** menu.
 - iv. Select **Datasources**.
- c. Select the datasource to be deleted, then click **Remove**.

Result

The non-XA datasource has been removed from the server.

- To add a new datasource, refer here: [Section 6.3.1, “Create a Non-XA Datasource with the Management Interfaces”](#).

[Report a bug](#)

6.4. XA DATASOURCES

6.4.1. Create an XA Datasource with the Management Interfaces

Prerequisites:

- [Section 2.2.1, “Start JBoss EAP 6”](#)

Summary

This topic covers the steps required to create an XA datasource, using either the Management Console or the Management CLI.



NOTE

Prior to version 10.2 of the Oracle datasource, the `<no-tx-separate-pools/>` parameter was required, as mixing non-transactional and transactional connections would result in an error. This parameter may no longer be required for certain applications.

Procedure 6.9. Create an XA Datasource, Using Either the Management CLI or the Management Console

- ○ **Management CLI**
 - a. [Section 3.4.2, “Launch the Management CLI”](#) .
 - b. Run the following Management CLI command to create an XA datasource, configuring the variables as appropriate:

**NOTE**

The value for *DRIVER_NAME* depends on the number of classes listed in the **/META-INF/services/java.sql.Driver** file located in the JDBC driver JAR. If there is only one class, the value is the name of the JAR. If there are multiple classes, the value is the name of the JAR + *driverClassName* + "_" + *majorVersion* + "_" + *minorVersion*. Failure to do so will result in the following error being logged:

```
JBAS014775: New missing/unsatisfied dependencies
```

For example, the *DRIVER_NAME* value required for the MySQL 5.1.31 driver, is **mysql-connector-java-5.1.31-bin.jarcom.mysql.jdbc.Driver_5_1**.

```
xa-data-source add --name=XA_DATASOURCE_NAME --jndi-name=JNDI_NAME --
driver-name=DRIVER_NAME --xa-datasource-class=XA_DATASOURCE_CLASS
```

c. **Configure the XA datasource properties**

i. **Set the server name**

Run the following command to configure the server name for the host:

```
/subsystem=datasources/xa-data-source=XA_DATASOURCE_NAME/xa-
datasource-properties=ServerName:add(value=HOSTNAME)
```

ii. **Set the database name**

Run the following command to configure the database name:

```
/subsystem=datasources/xa-data-source=XA_DATASOURCE_NAME/xa-
datasource-properties=DatabaseName:add(value=DATABASE_NAME)
```

d. Enable the datasource:

```
xa-data-source enable --name=XA_DATASOURCE_NAME
```

o **Management Console**

a. [Section 3.3.2, "Log in to the Management Console"](#) .

b. **Navigate to the Datasources panel in the Management Console**

i. Select the **Configuration** tab from the top of the console.

ii. For Domain mode only, select a profile from the drop-down box at the top left.

iii. Expand the **Subsystems** menu on the left of the console, then expand the **Connector** menu.

iv. Select **Datasources**.

c. Select the **XA Datasource** tab.

d. **Create a new XA datasource**

- i. Click **Add**.
- ii. Enter the new XA datasource attributes in the **Create XA Datasource** wizard and click **Next**.
- iii. Enter the JDBC driver details in the **Create XA Datasource** wizard and click **Next**.
- iv. Enter the XA properties and click **Next**.
- v. Enter the connection settings in the **Create XA Datasource** wizard.
- vi. Click the **Test Connection** button to test the connection to the XA datasource and verify the settings are correct.
- vii. Click **Done** to finish

Result

The XA datasource has been added to the server. It is now visible in either the **standalone.xml** or **domain.xml** file, as well as the management interfaces.

See Also:

- [Section 6.4.2, “Modify an XA Datasource with the Management Interfaces”](#)
- [Section 6.4.3, “Remove an XA Datasource with the Management Interfaces”](#)

[Report a bug](#)

6.4.2. Modify an XA Datasource with the Management Interfaces

Summary

This topic covers the steps required to modify an XA datasource, using either the Management Console or the Management CLI.

Prerequisites

- [Section 2.2.1, “Start JBoss EAP 6”](#).

Procedure 6.10. Modify an XA Datasource, Using Either the Management CLI or the Management Console

- - **Management CLI**
 - a. [Section 3.4.2, “Launch the Management CLI”](#).
 - b. **Configure XA datasource attributes**
Use the **write-attribute** command to configure a datasource attribute:

```
/subsystem=datasources/xa-data-source=XA_DATASOURCE_NAME:write-attribute(name=ATTRIBUTE_NAME,value=ATTRIBUTE_VALUE)
```

- **Configure XA datasource properties**
Run the following command to configure an XA datasource subresource:

```
/subsystem=datasources/xa-data-source=DATASOURCE_NAME/xa-datasource-properties=PROPERTY_NAME:add(value=PROPERTY_VALUE)
```

- d. Reload the server to confirm the changes:

```
reload
```

- o **Management Console**

- a. [Section 3.3.2, "Log in to the Management Console"](#) .
- b. **Navigate to the Datasources panel in the Management Console**
 - i. Select the **Configuration** tab from the top of the console.
 - ii. For Domain Mode only, select a profile from the drop-down box at top left.
 - iii. Expand the **Subsystems** menu on the left of the console, then expand the **Connector** menu.
 - iv. Select **Datasources**.
- c. Select the **XA Datasource** tab.
- d. **Edit the datasource**
 - i. Select the relevant XA datasource from the **Available XA Datasources** list. The XA datasource attributes are displayed in the **Attributes** panel below it.
 - ii. Select the **Edit** button to edit the datasource attributes.
 - iii. Edit the XA datasource attributes and select the **Save** button when done.

Result

The XA datasource has been configured. The changes are now visible in either the **standalone.xml** or **domain.xml** file, as well as the management interfaces.

- To create a new datasource, refer here: [Section 6.4.1, "Create an XA Datasource with the Management Interfaces"](#).
- To remove the datasource, refer here: [Section 6.4.3, "Remove an XA Datasource with the Management Interfaces"](#).

[Report a bug](#)

6.4.3. Remove an XA Datasource with the Management Interfaces

Summary

This topic covers the steps required to remove an XA datasource from JBoss EAP 6, using either the Management Console or the Management CLI.

Prerequisites

- [Section 2.2.1, "Start JBoss EAP 6"](#) .

Procedure 6.11. Remove an XA Datasource Using Either the Management CLI or the Management Console

- ○ **Management CLI**
 - a. [Section 3.4.2, “Launch the Management CLI”](#) .
 - b. Run the following command to remove an XA datasource:


```
xa-data-source remove --name=XA_DATASOURCE_NAME
```
- **Management Console**
 - a. [Section 3.3.2, “Log in to the Management Console”](#) .
 - b. **Navigate to the Datasources panel in the Management Console**
 - i. Select the **Configuration** tab from the top of the console.
 - ii. For Domain mode only, select a profile from the drop-down box in the top left.
 - iii. Expand the **Subsystems** menu on the left of the console, then expand the **Connector** menu.
 - iv. Select **Datasources**.
 - c. Select the **XA Datasource** tab.
 - d. Select the registered XA datasource to be deleted, and click **Remove** to permanently delete the XA datasource.

Result

The XA datasource has been removed from the server.

- To add a new XA datasource, refer here: [Section 6.4.1, “Create an XA Datasource with the Management Interfaces”](#).

[Report a bug](#)

6.4.4. XA Recovery

6.4.4.1. About XA Recovery Modules

Each XA resource needs a recovery module associated with its configuration. The recovery module must extend class **com.arjuna.ats.jta.recovery.XAResourceRecovery**.

JBoss EAP 6 provides recovery modules for JDBC and JMS XA resources. For these types of resources, recovery modules are automatically registered. If you need to use a custom module, you can register it in your datasource.

[Report a bug](#)

6.4.4.2. Configure XA Recovery Modules

For most JDBC and JMS resources, the recovery module is automatically associated with the resource. In these cases, you only need to configure the options that allow the recovery module to connect to your resource to perform recovery.

For custom resources which are not JDBC or JMS, contact Red Hat Global Support Services for information on supported configurations.

Each of these configuration attributes can be set either during datasource creation, or afterward. You can set them using either the web-based Management Console or the command-line Management CLI. Refer to [Section 6.4.1, “Create an XA Datasource with the Management Interfaces”](#) and [Section 6.4.2, “Modify an XA Datasource with the Management Interfaces”](#) for general information on configuring XA datasources.

Refer to the following tables for general datasource configuration attributes, and for information about configuration details relating to specific database vendors.

Table 6.2. General Configuration Attributes

Attribute	Description
recovery-username	The username the recovery module should use to connect to the resource for recovery.
recovery-password	The password the recovery module should use to connect to the resource for recovery.
recovery-security-domain	The security domain the recovery module should use to connect to the resource for recovery.
recovery-plugin-class-name	If you need to use a custom recovery module, set this attribute to the fully-qualified class name of the module. The module should extend class com.arjuna.ats.jta.recovery.XAResourceRecovery .
recovery-plugin-properties	If you use a custom recovery module which requires properties to be set, set this attribute to the list of comma-separated <i>key=value</i> pairs for the properties.



NOTE

The names used in the [Table 6.2, “General Configuration Attributes”](#) are parameters which are used when the datasource is configured via CLI. They may differ from names used in XML configuration file.

Vendor-Specific Configuration Information

This section describes specific configurations which are required to be done for particular database to cooperate in XA transactions managed by the JBoss EAP transaction manager. For more detailed information, refer to documentation for a particular database.

Oracle

If the Oracle datasource is configured incorrectly, you may see errors like the following in your log output:

Example 6.11. Incorrect configuration error

-

```
WARN [com.arjuna.ats.jta.logging.logger18N]
[com.arjuna.ats.internal.jta.recovery.xarecovery1] Local XARecoveryModule.xaRecovery got
XA exception javax.transaction.xa.XAException, XAException.XAER_RMERR
```

To resolve this error, ensure that the Oracle user configured in **recovery-username** has access to the tables needed for recovery. The following SQL statement shows the correct grants for Oracle 11g or Oracle 10g R2 instances patched for Oracle bug 5945463.

Example 6.12. Grant configuration

```
GRANT SELECT ON sys.dba_pending_transactions TO recovery-username;
GRANT SELECT ON sys.pending_trans$ TO recovery-username;
GRANT SELECT ON sys.dba_2pc_pending TO recovery-username;
GRANT EXECUTE ON sys.dbms_xa TO recovery-username;
```

If you use an Oracle 11 version prior to 11g, change the final **EXECUTE** statement to the following:

```
GRANT EXECUTE ON sys.dbms_system TO recovery-username;
```

PostgreSQL and Postgres Plus Advanced Server

For PostgreSQL to be able to handle XA transaction, change the configuration parameter **max_prepared_transactions** to an value higher than 0.

MySQL

No special configuration is required. For more information, refer to MySQL documentation.

IBM DB2

No special configuration is required. For more information, refer to IBM DB2 documentation.

Sybase

Sybase expects XA transactions to be enabled on the database. Without correct database configuration, XA transactions will not work. **enable xact coordination** enables or disables Adaptive Server transaction coordination services. When this parameter is enabled, Adaptive Server ensures that updates to remote Adaptive Server data commit or roll back with the original transaction. To enable transaction coordination, use:

```
sp_configure 'enable xact coordination', 1
```

MSSQL

For more information, refer to Microsoft documentation. You can also refer <http://msdn.microsoft.com/en-us/library/aa342335.aspx> as a starting point.

Vendor-specific issues and their consequences

This section describes the currently known database specific issues connected with handling XA transactions. These issues refers to database versions and jdbc drivers which are currently supported by particular EAP version.

Oracle

There are no known issues related to database.

PostgreSQL

- Jdbc driver returns XAER_RMERR when an error occurs during the call of commit method protocol. Database returns this error code and leaves the transaction in **in-doubt** state on the database side. The correct return code should be XAER_RMFAIL or XAER_RETRY. For more information, see <https://github.com/pgjdbc/pgjdbc/issues/236>. This causes the transaction to be left in the **Heuristic** state on the EAP side and holding locks in database which requires user intervention.
- The incorrect error code returned by the jdbc driver can cause data inconsistency in some cases. For more information, refer to <https://developer.jboss.org/thread/251537>, <https://github.com/pgjdbc/pgjdbc/issues/236>

Postgres Plus Advanced Server

- Jdbc driver returns XAER_RMERR when an error occurs during the call of commit method protocol. Database returns this error code and leaves the transaction in **in-doubt** state on the database side. The correct return code should be XAER_RMFAIL or XAER_RETRY. For more information, see <https://github.com/pgjdbc/pgjdbc/issues/236>. This causes the transaction to be left in the **Heuristic** state on the EAP side and holding locks in database which requires user intervention.
- The incorrect error code returned by the jdbc driver can cause data inconsistency in some cases. For more information, refer to <https://developer.jboss.org/thread/251537>
- If **XAResource.rollback** is called for the same XID more than once then **XAException** is thrown. Calling rollback against the same XID several times is compliant with JTS spec and no **XAException** should be thrown. For more information, see <https://github.com/pgjdbc/pgjdbc/issues/78>.

MySQL

MySQL is not capable of handling XA transactions. If a client is disconnected the MySQL then all the information about such transactions is lost. For more information, see <http://bugs.mysql.com/bug.php?id=12161>.

IBM DB2

There are no known issues related to database.

Sybase

- Jdbc driver returns XAER_RMERR when an error occurs during the call of commit method protocol. Database returns this error code and leaves the transaction in **in-doubt** state on the database side. The correct return code should be XAER_RMFAIL or XAER_RETRY. For more information, see <https://github.com/pgjdbc/pgjdbc/issues/236>. This causes the transaction to be left in the **Heuristic** state on the EAP side and holding locks in database which requires user intervention.
- If **XAResource.rollback** is called for the same XID more than once then **XAException** is

thrown. Calling rollback against the same XID several times is compliant with JTS spec and no **XAException** should be thrown. For more information, see <https://github.com/pgjdbc/pgjdbc/issues/78>.

MSSQL

- Jdbc driver returns XAER_RMERR when an error occurs during the call of commit method protocol. Database returns this error code and leaves the transaction in **in-doubt** state on the database side. The correct return code should be XAER_RMFAIL or XAER_RETRY. For more information, see <https://github.com/pgjdbc/pgjdbc/issues/236>. This causes the transaction to be left in the **Heuristic** state on the EAP side and holding locks in database which requires user intervention.

This issue was reported on the Microsoft Connect site at <https://connect.microsoft.com/SQLServer/feedback/details/1207381/jdbc-driver-is-not-xa-compliant-in-case-of-connection-failure-error-code-xaer-rmerr-is-returned-instead-of-xaer-rmretry>.

- If **XAResource.rollback** is called for the same XID more than once then **XAException** is thrown. Calling rollback against the same XID several times is compliant with JTS spec and no **XAException** should be thrown. For more information, see <https://github.com/pgjdbc/pgjdbc/issues/78>.
- The call of **XAResource.rollback** for the same XID causes the following message being logged in server log file.

```
WARN [com.arjuna.ats.jtax] ...: XAResourceRecord.rollback caused an error from
resource ... in transaction ...: java.lang.NullPointerException
```

Messaging

IBM Websphere

- When JTS is used then second call of rollback against the same XID during the periodic recovery can cause an error to be logged in the log file. Second rollback against the same XID is JTS complaint and can be ignored.

When RAR does not know such XID, then XAER_NOTA return code is expected. But IBM Websphere may return an incorrect return code XAER_RMFAIL.

```
The method 'xa_rollback' has failed with errorCode '-7' due to the resource being closed.'
```

ActiveMQ

- Resource adapter returns XAER_RMERR when an error occurs during the call of commit method protocol, for example network disconnection. Resource adapter returns this error code back to transaction manager and it causes the transaction being left in **in-doubt** state on message broker side. This behaviour is against XA specification and the correct return code should be XAER_RMFAIL or XAER_RETRY. The wrong error code can cause data inconsistency in some cases. For more information, refer to <https://developer.jboss.org/thread/251537> .

```
WARN [com.arjuna.ats.jtax] ...: XAResourceRecord.rollback caused an XA error:
ARJUNA016099: Unknown error code:0 from resource ... in transaction ...:
javax.transaction.xa.XAException: Transaction ... has not been started.
```

TIBCO

- When JTS is used then second call of rollback against the same XID during the periodic recovery can cause an error to be logged in the log file. Second rollback against the same XID is JTS complaint and can be ignored.

```
WARN [com.arjuna.ats.jtax] ...: XAResourceRecord.rollback caused an XA error:
XAException.XAER_RMFAIL from resource ... in transaction ...:
javax.transaction.xa.XAException
```

[Report a bug](#)

6.5. DATASOURCE SECURITY

6.5.1. About Datasource Security

Datasource security refers to encrypting or obscuring passwords for datasource connections. These passwords can be stored in plain text in configuration files, however this represents a security risk.

The preferred solution for datasource security is the use of either security domains or password vaults. Examples of each are included below. For more information, refer to the *Security Architecture* and other JBoss EAP security documentation.

Example 6.13. Security Domain Example

```
<security-domain name="DsRealm" cache-type="default">
  <authentication>
    <login-module code="ConfiguredIdentity" flag="required">
      <module-option name="userName" value="sa"/>
      <module-option name="principal" value="sa"/>
      <module-option name="password" value="sa"/>
    </login-module>
  </authentication>
</security-domain>
```

The DsRealm domain is referenced by a datasource like so:

```
<datasources>
  <datasource jndi-name="java:jboss/datasources/securityDs"
    pool-name="securityDs">
    <connection-url>jdbc:h2:mem:test;DB_CLOSE_DELAY=-1</connection-url>
    <driver>h2</driver>
    <new-connection-sql>select current_user()</new-connection-sql>
    <security>
      <security-domain>DsRealm</security-domain>
    </security>
  </datasource>
</datasources>
```



NOTE

If a security domain will be used with multiple datasources, then caching should be disabled on the security domain. This can be accomplished by setting the value of the **cache-type** attribute to **none** or by removing the attribute altogether. However, if caching is desired, then a separate security domain should be used for each datasource.

Example 6.14. Password Vault Example

```
<security>
  <user-name>admin</user-name>

  <password>${VAULT::ds_ExampleDS::password::N2NhZDYzOTMtNWE0OS00ZGQ0LWE4MmEt
MWNIMDMYNDdmNmI2TEIORV9CUkVBS3ZhdWx0}</password>
</security>
```

[Report a bug](#)

6.6. DATABASE CONNECTION VALIDATION

6.6.1. Configure Database Connection Validation Settings

Overview

Database maintenance, network problems, or other outage events may cause JBoss EAP 6 to lose the connection to the database. You enable database connection validation using the **<validation>** element within the **<datasource>** section of the server configuration file. Follow the steps below to configure the datasource settings to enable database connection validation in JBoss EAP 6.

Procedure 6.12. Configure Database Connection Validation Settings

1. Choose a Validation Method

Select one of the following validation methods.

- o **<validate-on-match>true</validate-on-match>**

When the **<validate-on-match>** option is set to **true**, the database connection is validated every time it is checked out from the connection pool using the validation mechanism specified in the next step.

If a connection is not valid, a warning is written to the log and it retrieves the next connection in the pool. This process continues until a valid connection is found. If you prefer not to cycle through every connection in the pool, you can use the **<use-fast-fail>** option. If a valid connection is not found in the pool, a new connection is created. If the connection creation fails, an exception is returned to the requesting application.

This setting results in the quickest recovery but creates the highest load on the database. However, this is the safest selection if the minimal performance hit is not a concern.

- o **<background-validation>true</background-validation>**

When the **<background-validation>** option is set to **true**, it is used in combination with the

<background-validation-millis> value to determine how often background validation runs. The default value for the **<background-validation-millis>** parameter is 0 milliseconds, meaning it is disabled by default. This value should not be set to the same value as your **<idle-timeout-minutes>** setting.

It is a balancing act to determine the optimum **<background-validation-millis>** value for a particular system. The lower the value, the more frequently the pool is validated and the sooner invalid connections are removed from the pool. However, lower values take more database resources. Higher values result in less frequent connection validation checks and use less database resources, but dead connections are undetected for longer periods of time.



NOTE

If the **<validate-on-match>** option is set to **true**, the **<background-validation>** option should be set to **false**. The reverse is also true. If the **<background-validation>** option is set to **true**, the **<validate-on-match>** option should be set to **false**.

2. Choose a Validation Mechanism

Select one of the following validation mechanisms.

o Specify a **<valid-connection-checker>** Class Name

This is the preferred mechanism as it optimized for the particular RDBMS in use. JBoss EAP 6 provides the following connection checkers:

- `org.jboss.jca.adapters.jdbc.extensions.db2.DB2ValidConnectionChecker`
- `org.jboss.jca.adapters.jdbc.extensions.mssql.MSSQLValidConnectionChecker`
- `org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLReplicationValidConnectionChecker`
- `org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLValidConnectionChecker`
- `org.jboss.jca.adapters.jdbc.extensions.novendor.JDBC4ValidConnectionChecker`
- `org.jboss.jca.adapters.jdbc.extensions.novendor.NullValidConnectionChecker`
- `org.jboss.jca.adapters.jdbc.extensions.oracle.OracleValidConnectionChecker`
- `org.jboss.jca.adapters.jdbc.extensions.postgres.PostgreSQLValidConnectionChecker`
- `org.jboss.jca.adapters.jdbc.extensions.sybase.SybaseValidConnectionChecker`

o Specify SQL for **<check-valid-connection-sql>**

You provide the SQL statement used to validate the connection.

The following is an example of how you might specify a SQL statement to validate a connection for Oracle:

```
<check-valid-connection-sql>select 1 from dual</check-valid-connection-sql>
```

For MySQL or PostgreSQL, you might specify the following SQL statement:

```
<check-valid-connection-sql>select 1</check-valid-connection-sql>
```


3. Set the <exception-sorter> Class Name

When an exception is marked as fatal, the connection is closed immediately, even if the connection is participating in a transaction. Use the exception sorter class option to properly detect and clean up after fatal connection exceptions. JBoss EAP 6 provides the following exception sorters:

- `org.jboss.jca.adapters.jdbc.extensions.db2.DB2ExceptionSorter`
- `org.jboss.jca.adapters.jdbc.extensions.informix.InformixExceptionSorter`
- `org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLExceptionSorter`
- `org.jboss.jca.adapters.jdbc.extensions.novendor.NullExceptionSorter`
- `org.jboss.jca.adapters.jdbc.extensions.oracle.OracleExceptionSorter`
- `org.jboss.jca.adapters.jdbc.extensions.postgres.PostgreSQLExceptionSorter`
- `org.jboss.jca.adapters.jdbc.extensions.sybase.SybaseExceptionSorter`
- `org.jboss.jca.adapters.jdbc.extensions.mssql.MSQLExceptionSorter`

[Report a bug](#)

6.7. DATASOURCE CONFIGURATION

6.7.1. Datasource Parameters

Table 6.3. Datasource parameters common to non-XA and XA datasources

Parameter	Description
<code>jndi-name</code>	The unique JNDI name for the datasource.
<code>pool-name</code>	The name of the management pool for the datasource.
<code>enabled</code>	Whether or not the datasource is enabled.
<code>use-java-context</code>	Whether to bind the datasource to global JNDI.
<code>spy</code>	Enable spy functionality on the JDBC layer. This logs all JDBC traffic to the datasource. Note that the logging category jboss.jdbc.spy must also be set to the log level DEBUG in the logging subsystem.
<code>use-ccm</code>	Enable the cached connection manager.
<code>new-connection-sql</code>	A SQL statement which executes when the connection is added to the connection pool.

Parameter	Description
transaction-isolation	One of the following: <ul style="list-style-type: none"> ● TRANSACTION_READ_UNCOMMITTED ● TRANSACTION_READ_COMMITTED ● TRANSACTION_REPEATABLE_READ ● TRANSACTION_SERIALIZABLE ● TRANSACTION_NONE
url-selector-strategy-class-name	A class that implements interface org.jboss.jca.adapters.jdbc.URLSelectorStrategy .
security	Contains child elements which are security settings. See Table 6.8, "Security parameters" .
validation	Contains child elements which are validation settings. See Table 6.9, "Validation parameters" .
timeout	Contains child elements which are timeout settings. See Table 6.10, "Timeout parameters" .
statement	Contains child elements which are statement settings. See Table 6.11, "Statement parameters" .

Table 6.4. Non-XA datasource parameters

Parameter	Description
jta	Enable JTA integration for non-XA datasources. Does not apply to XA datasources.
connection-url	The JDBC driver connection URL.
driver-class	The fully-qualified name of the JDBC driver class.
connection-property	Arbitrary connection properties passed to the method Driver.connect(url,props) . Each connection-property specifies a string name/value pair. The property name comes from the name, and the value comes from the element content.
pool	Contains child elements which are pooling settings. See Table 6.6, "Pool parameters common to non-XA and XA datasources" .

Parameter	Description
url-delimiter	The delimiter for URLs in a connection-url for High Availability (HA) clustered databases.

Table 6.5. XA datasource parameters

Parameter	Description
xa-datasource-property	A property to assign to implementation class XADatasource . Specified by <i>name=value</i> . If a setter method exists, in the format setName , the property is set by calling a setter method in the format of setName(value) .
xa-datasource-class	The fully-qualified name of the implementation class javax.sql.XADatasource .
driver	A unique reference to the class loader module which contains the JDBC driver. The accepted format is <i>driverName#majorVersion.minorVersion</i> .
xa-pool	Contains child elements which are pooling settings. See Table 6.6, "Pool parameters common to non-XA and XA datasources" and Table 6.7, "XA pool parameters".
recovery	Contains child elements which are recovery settings. See Table 6.12, "Recovery parameters".

Table 6.6. Pool parameters common to non-XA and XA datasources

Parameter	Description
min-pool-size	The minimum number of connections a pool holds.
max-pool-size	The maximum number of connections a pool can hold.
prefill	Whether to try to prefill the connection pool. The default is false .
use-strict-min	Whether the idle connection scan should strictly stop marking for closure of any further connections, once the min-pool-size has been reached. The default value is false .

Parameter	Description
flush-strategy	<p>Whether the pool is flushed in the case of an error. Valid values are:</p> <ul style="list-style-type: none"> ● FailingConnectionOnly ● IdleConnections ● EntirePool <p>The default is FailingConnectionOnly.</p>
allow-multiple-users	<p>Specifies if multiple users will access the datasource through the getConnection(user, password) method, and whether the internal pool type accounts for this behavior.</p>

Table 6.7. XA pool parameters

Parameter	Description
is-same-rm-override	<p>Whether the javax.transaction.xa.XAResource.isSameRM(XAResource) class returns true or false.</p>
interleaving	<p>Whether to enable interleaving for XA connection factories.</p>
no-tx-separate-pools	<p>Whether to create separate sub-pools for each context. This is required for Oracle datasources, which do not allow XA connections to be used both inside and outside of a JTA transaction.</p> <p>Using this option will cause your total pool size to be twice max-pool-size, because two actual pools will be created.</p>
pad-xid	<p>Whether to pad the Xid.</p>
wrap-xa-resource	<p>Whether to wrap the XAResource in an org.jboss.tm.XAResourceWrapper instance.</p>

Table 6.8. Security parameters

Parameter	Description
user-name	<p>The username to use to create a new connection.</p>
password	<p>The password to use to create a new connection.</p>

Parameter	Description
security-domain	Contains the name of a JAAS security-manager which handles authentication. This name correlates to the application-policy/name attribute of the JAAS login configuration.
reauth-plugin	Defines a reauthentication plug-in to use to reauthenticate physical connections.

Table 6.9. Validation parameters

Parameter	Description
valid-connection-checker	An implementation of interface org.jboss.jca.adapters.jdbc.ValidConnectionChecker which provides a SQLException.isValidConnection(Connection e) method to validate a connection. An exception means the connection is destroyed. This overrides the parameter check-valid-connection-sql if it is present.
check-valid-connection-sql	An SQL statement to check validity of a pool connection. This may be called when a managed connection is taken from a pool for use.
validate-on-match	Indicates whether connection level validation is performed when a connection factory attempts to match a managed connection for a given set. Specifying "true" for validate-on-match is typically not done in conjunction with specifying "true" for background-validation . Validate-on-match is needed when a client must have a connection validated prior to use. This parameter is false by default.
background-validation	Specifies that connections are validated on a background thread. Background validation is a performance optimization when not used with validate-on-match . If validate-on-match is true, using background-validation could result in redundant checks. Background validation does leave open the opportunity for a bad connection to be given to the client for use (a connection goes bad between the time of the validation scan and prior to being handed to the client), so the client application must account for this possibility.
background-validation-millis	The amount of time, in milliseconds, that background validation runs.

Parameter	Description
use-fast-fail	If true, fail a connection allocation on the first attempt, if the connection is invalid. Defaults to false .
stale-connection-checker	An instance of org.jboss.jca.adapters.jdbc.StaleConnectionChecker which provides a Boolean isStaleConnection(SQLException e) method. If this method returns true , the exception is wrapped in an org.jboss.jca.adapters.jdbc.StaleConnectionException , which is a subclass of SQLException .
exception-sorter	An instance of org.jboss.jca.adapters.jdbc.ExceptionSorter which provides a Boolean isExceptionFatal(SQLException e) method. This method validates whether an exception is broadcast to all instances of javax.resource.spi.ConnectionEventListener as a connectionErrorOccurred message.

Table 6.10. Timeout parameters

Parameter	Description
use-try-lock	Uses tryLock() instead of lock() . This attempts to obtain the lock for the configured number of seconds, before timing out, rather than failing immediately if the lock is unavailable. Defaults to 60 seconds. As an example, to set a timeout of 5 minutes, set <use-try-lock>300</use-try-lock> .
blocking-timeout-millis	The maximum time, in milliseconds, to block while waiting for a connection. After this time is exceeded, an exception is thrown. This blocks only while waiting for a permit for a connection, and does not throw an exception if creating a new connection takes a long time. Defaults to 30000, which is 30 seconds.
idle-timeout-minutes	The maximum time, in minutes, before an idle connection is closed. If not specified, the default is 30 minutes. The actual maximum time depends upon the idleRemover scan time, which is half of the smallest idle-timeout-minutes of any pool.
set-tx-query-timeout	Whether to set the query timeout based on the time remaining until transaction timeout. Any configured query timeout is used if no transaction exists. Defaults to false .
query-timeout	Timeout for queries, in seconds. The default is no timeout.

Parameter	Description
allocation-retry	The number of times to retry allocating a connection before throwing an exception. The default is 0 , so an exception is thrown upon the first failure.
allocation-retry-wait-millis	How long, in milliseconds, to wait before retrying to allocate a connection. The default is 5000, which is 5 seconds.
xa-resource-timeout	If non-zero, this value is passed to method XAResource.setTimeout .

Table 6.11. Statement parameters

Parameter	Description
track-statements	<p>Whether to check for unclosed statements when a connection is returned to a pool and a statement is returned to the prepared statement cache. If false, statements are not tracked.</p> <p>Valid values</p> <ul style="list-style-type: none"> ● true: statements and result sets are tracked, and a warning is issued if they are not closed. ● false: neither statements or result sets are tracked. ● nowarn: statements are tracked but no warning is issued. This is the default.
prepared-statement-cache-size	The number of prepared statements per connection, in a Least Recently Used (LRU) cache.
share-prepared-statements	Whether JBoss EAP should cache, instead of close or terminate, the underlying physical statement when the wrapper supplied to the application is closed by application code. The default is false .


Table 6.12. Recovery parameters

Parameter	Description
recover-credential	A username/password pair or security domain to use for recovery.
recover-plugin	An implementation of the org.jboss.jca.core.spi.recovery.RecoveryPlugin class, to be used for recovery.

[Report a bug](#)

6.7.2. Datasource Connection URLs

Table 6.13. Datasource Connection URLs

Datasource	Connection URL
PostgreSQL	<code>jdbc:postgresql://SERVER_NAME:PORT/DATABASE_NAME</code>
MySQL	<code>jdbc:mysql://SERVER_NAME:PORT/DATABASE_NAME</code>
Oracle	<code>jdbc:oracle:thin:@ORACLE_HOST:PORT:ORACLE_SID</code>
IBM DB2	<code>jdbc:db2://SERVER_NAME:PORT/DATABASE_NAME</code>
Microsoft SQLServer	<p><code>jdbc:sqlserver://SERVER_NAME:PORT;DatabaseName=DATABASE_NAME</code></p> <div style="display: flex; align-items: flex-start;"> <div style="flex: 1;">  </div> <div style="flex: 2;"> <p>NOTE</p> <p>The <code>jdbc:microsoft:sqlserver://SERVER_NAME:PORT;DatabaseName=DATABASE_NAME</code> template does not work with new database.</p> </div> </div>

[Report a bug](#)

6.7.3. Datasource Extensions

Datasource deployments can use several extensions in the JDBC resource adapter to improve the connection validation, and check whether an exception should reestablish the connection. Those extensions are:

Table 6.14. Datasource Extensions

Datasource Extension	Configuration Parameter	Description
<code>org.jboss.jca.adapters.jdbc.spi.ExceptionSorter</code>	<code><exception-sorter></code>	Checks whether an <code>SQLException</code> is fatal for the connection on which it was thrown
<code>org.jboss.jca.adapters.jdbc.spi.StaleConnectionChecker</code>	<code><stale-connection-checker></code>	Wraps stale <code>SQLExceptions</code> in a <code>org.jboss.jca.adapters.jdbc.StaleConnectionException</code>

Datasource Extension	Configuration Parameter	Description
org.jboss.jca.adapters.jdbc.spi.ValidConnection	<valid-connection-checker>	Checks whether a connection is valid for use by the application

JBoss EAP 6 also features implementations of these extensions for several supported databases.

Extension Implementations

Generic

- org.jboss.jca.adapters.jdbc.extensions.novendor.NullExceptionSorter
- org.jboss.jca.adapters.jdbc.extensions.novendor.NullStaleConnectionChecker
- org.jboss.jca.adapters.jdbc.extensions.novendor.NullValidConnectionChecker
- org.jboss.jca.adapters.jdbc.extensions.novendor.JDBC4ValidConnectionChecker

PostgreSQL

- org.jboss.jca.adapters.jdbc.extensions.postgres.PostgreSQLExceptionSorter
- org.jboss.jca.adapters.jdbc.extensions.postgres.PostgreSQLValidConnectionChecker

MySQL

- org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLExceptionSorter
- org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLReplicationValidConnectionChecker
- org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLValidConnectionChecker

IBM DB2

- org.jboss.jca.adapters.jdbc.extensions.db2.DB2ExceptionSorter
- org.jboss.jca.adapters.jdbc.extensions.db2.DB2StaleConnectionChecker
- org.jboss.jca.adapters.jdbc.extensions.db2.DB2ValidConnectionChecker

Sybase

- org.jboss.jca.adapters.jdbc.extensions.sybase.SybaseExceptionSorter
- org.jboss.jca.adapters.jdbc.extensions.sybase.SybaseValidConnectionChecker

Microsoft SQLServer

- org.jboss.jca.adapters.jdbc.extensions.mssql.MSSQLExceptionSorter
- org.jboss.jca.adapters.jdbc.extensions.mssql.MSSQLValidConnectionChecker

Oracle

- org.jboss.jca.adapters.jdbc.extensions.oracle.OracleExceptionSorter
- org.jboss.jca.adapters.jdbc.extensions.oracle.OracleStaleConnectionChecker
- org.jboss.jca.adapters.jdbc.extensions.oracle.OracleValidConnectionChecker

[Report a bug](#)

6.7.4. View Datasource Statistics

You can view statistics from defined datasources for both the **jdbc** and **pool** using appropriately modified versions of the commands below:

Example 6.15. CLI for domain mode:

Change **/host=master/server=server-one** and **data-source=ExampleDS** according to the environment.

```
[domain@localhost:9999 /] /host=master/server=server-one/subsystem=datasources/data-
source=ExampleDS/statistics=pool:read-resource(include-runtime=true)
{
  "outcome" => "success",
  "result" => {
    "ActiveCount" => "0",
    "AvailableCount" => "20",
    "AverageBlockingTime" => "0",
    "AverageCreationTime" => "0",
    "CreatedCount" => "0",
    "DestroyedCount" => "0",
    "MaxCreationTime" => "0",
    "MaxUsedCount" => "0",
    "MaxWaitTime" => "0",
    "TimedOut" => "0",
    "TotalBlockingTime" => "0",
    "TotalCreationTime" => "0"
  }
}
```

Example 6.16. CLI for standalone mode:

Change **data-source=ExampleDS** according to the environment.

```
[standalone@localhost:9999 /] /subsystem=datasources/data-
source=ExampleDS/statistics=pool:read-resource(include-runtime=true)
{
  "outcome" => "success",
  "result" => {
    "ActiveCount" => "0",
    "AvailableCount" => "20",
    "AverageBlockingTime" => "0",
    "AverageCreationTime" => "0",
```

```

"CreatedCount" => "0",
"DestroyedCount" => "0",
"MaxCreationTime" => "0",
"MaxUsedCount" => "0",
"MaxWaitTime" => "0",
"TimedOut" => "0",
"TotalBlockingTime" => "0",
"TotalCreationTime" => "0"
}
}

```



NOTE

Ensure you specify the ***include-runtime=true*** argument, as all statistics are runtime only information and the default is **false**.

[Report a bug](#)

6.7.5. Datasource Statistics

Core Statistics

The following table contains a list of the supported datasource core statistics:

Table 6.15. Core Statistics

Name	Description
ActiveCount	The number of active connections. Each of the connections is either in use by an application or available in the pool
AvailableCount	The number of available connections in the pool.
AverageBlockingTime	The average time spent blocking on obtaining an exclusive lock on the pool. The value is in milliseconds.
AverageCreationTime	The average time spent creating a connection. The value is in milliseconds.
CreatedCount	The number of connections created.
DestroyedCount	The number of connections destroyed.
InUseCount	The number of connections currently in use.
MaxCreationTime	The maximum time it took to create a connection. The value is in milliseconds.
MaxUsedCount	The maximum number of connections used.
MaxWaitCount	The maximum number of requests waiting for a connection at the same time.

Name	Description
MaxWaitTime	The maximum time spent waiting for an exclusive lock on the pool.
TimedOut	The number of timed out connections.
TotalBlockingTime	The total time spent waiting for an exclusive lock on the pool. The value is in milliseconds.
TotalCreationTime	The total time spent creating connections. The value is in milliseconds.

JDBC Statistics

The following table contains a list of the supported datasource JDBC statistics:

Table 6.16. JDBC Statistics

Name	Description
PreparedStatementCacheAccessCount	The number of times that the statement cache was accessed.
PreparedStatementCacheAddCount	The number of statements added to the statement cache.
PreparedStatementCacheCurrentSize	The number of prepared and callable statements currently cached in the statement cache.
PreparedStatementCacheDeleteCount	The number of statements discarded from the cache.
PreparedStatementCacheHitCount	The number of times that statements from the cache were used.
PreparedStatementCacheMissCount	The number of times that a statement request could not be satisfied with a statement from the cache.

You can enable **Core** and **JDBC** statistics using appropriately modified versions of the following commands:

- `/subsystem=datasources/data-source=ExampleDS/statistics=pool:write-attribute(name=statistics-enabled,value=true)`
- `/subsystem=datasources/data-source=ExampleDS/statistics=jdbc:write-attribute(name=statistics-enabled,value=true)`

[Report a bug](#)

6.8. EXAMPLE DATASOURCES

6.8.1. Example PostgreSQL Datasource

The example below is a PostgreSQL datasource configuration. The datasource has been enabled, a user has been added, and validation options have been set.

Example 6.17. PostSQL datasource configuration

```
<datasources>
  <datasource jndi-name="java:jboss/PostgresDS" pool-name="PostgresDS">
    <connection-url>jdbc:postgresql://localhost:5432/postgresdb</connection-url>
    <driver>postgresql</driver>
    <security>
      <user-name>admin</user-name>
      <password>admin</password>
    </security>
    <validation>
      <validate-on-match>>true</validate-on-match>
      <background-validation>>false</background-validation>
      <valid-connection-checker class-
name="org.jboss.jca.adapters.jdbc.extensions.postgres.PostgreSQLValidConnectionChecker">
</valid-connection-checker>
      <exception-sorter class-
name="org.jboss.jca.adapters.jdbc.extensions.postgres.PostgreSQLExceptionSorter">
</exception-sorter>
    </validation>
  </datasource>
</drivers>
  <driver name="postgresql" module="org.postgresql">
    <xa-datasource-class>org.postgresql.xa.PGXADatasource</xa-datasource-class>
  </driver>
</drivers>
</datasources>
```

The example below is a **module.xml** file for the PostgreSQL datasource above.

Example 6.18. module.xml

```
<module xmlns="urn:jboss:module:1.1" name="org.postgresql">
  <resources>
    <resource-root path="postgresql-9.1-902.jdbc4.jar"/>
  </resources>
  <dependencies>
    <module name="javax.api"/>
    <module name="javax.transaction.api"/>
  </dependencies>
</module>
```

[Report a bug](#)

6.8.2. Example PostgreSQL XA Datasource

The example below is a PostgreSQL XA datasource configuration. The datasource has been enabled, a user has been added, and validation options have been set.

Example 6.19. PostSQL XA datasource

```
<datasources>
  <xa-datasource jndi-name="java:jboss/PostgresXADS" pool-name="PostgresXADS">
    <driver>postgresql</driver>
    <xa-datasource-property name="ServerName">localhost</xa-datasource-property>
    <xa-datasource-property name="PortNumber">5432</xa-datasource-property>
    <xa-datasource-property name="DatabaseName">postgresdb</xa-datasource-property>
    <security>
      <user-name>admin</user-name>
      <password>admin</password>
    </security>
    <validation>
      <validate-on-match>true</validate-on-match>
      <background-validation>>false</background-validation>
      <valid-connection-checker class-
name="org.jboss.jca.adapters.jdbc.extensions.postgres.PostgreSQLValidConnectionChecker">
      </valid-connection-checker>
      <exception-sorter class-
name="org.jboss.jca.adapters.jdbc.extensions.postgres.PostgreSQLExceptionSorter">
      </exception-sorter>
    </validation>
  </xa-datasource>
</drivers>
  <driver name="postgresql" module="org.postgresql">
    <xa-datasource-class>org.postgresql.xa.PGXADDataSource</xa-datasource-class>
  </driver>
</drivers>
</datasources>
```

The example below is a **module.xml** file for the PostgreSQL XA datasource above.

Example 6.20. module.xml

```
<module xmlns="urn:jboss:module:1.1" name="org.postgresql">
  <resources>
    <resource-root path="postgresql-9.1-902.jdbc4.jar"/>
  </resources>
  <dependencies>
    <module name="javax.api"/>
    <module name="javax.transaction.api"/>
  </dependencies>
</module>
```

[Report a bug](#)

6.8.3. Example MySQL Datasource

The example below is a MySQL datasource configuration. The datasource has been enabled, a user has been added, and validation options have been set.

Example 6.21. MySQL datasource configuration

```
<datasources>
  <datasource jndi-name="java:jboss/MySqlDS" pool-name="MySqlDS">
    <connection-url>jdbc:mysql://mysql-localhost:3306/jbossdb</connection-url>
    <driver>mysql</driver>
    <security>
      <user-name>admin</user-name>
      <password>admin</password>
    </security>
    <validation>
      <validate-on-match>true</validate-on-match>
      <background-validation>>false</background-validation>
      <valid-connection-checker class-
name="org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLValidConnectionChecker"></valid-
connection-checker>
      <exception-sorter class-
name="org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLExceptionSorter"></exception-
sorter>
    </validation>
  </datasource>
  <drivers>
    <driver name="mysql" module="com.mysql">
      <xa-datasource-class>com.mysql.jdbc.jdbc2.optional.MysqlXADataSource</xa-datasource-
class>
    </driver>
  </drivers>
</datasources>
```

The example below is a **module.xml** file for the MySQL datasource above.

Example 6.22. module.xml

```
<module xmlns="urn:jboss:module:1.1" name="com.mysql">
  <resources>
    <resource-root path="mysql-connector-java-5.0.8-bin.jar"/>
  </resources>
  <dependencies>
    <module name="javax.api"/>
    <module name="javax.transaction.api"/>
  </dependencies>
</module>
```

[Report a bug](#)

6.8.4. Example MySQL XA Datasource

The example below is a MySQL XA datasource configuration. The datasource has been enabled, a user has been added, and validation options have been set.

Example 6.23. MySQL XA datasource

```

<datasources>
  <xa-datasource jndi-name="java:jboss/MysqlXADS" pool-name="MysqlXADS">
    <driver>mysql</driver>
    <xa-datasource-property name="ServerName">localhost</xa-datasource-property>
    <xa-datasource-property name="DatabaseName">mysql</xa-datasource-property>
    <security>
      <user-name>admin</user-name>
      <password>admin</password>
    </security>
    <validation>
      <validate-on-match>true</validate-on-match>
      <background-validation>>false</background-validation>
      <valid-connection-checker class-
name="org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLValidConnectionChecker"></valid-
connection-checker>
      <exception-sorter class-
name="org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLExceptionSorter"></exception-
sorter>
    </validation>
  </xa-datasource>
</drivers>
  <driver name="mysql" module="com.mysql">
    <xa-datasource-class>com.mysql.jdbc.jdbc2.optional.MysqlXADataSource</xa-datasource-
class>
  </driver>
</drivers>
</datasources>

```

The example below is a **module.xml** file for the MySQL XA datasource above.

Example 6.24. module.xml

```

<module xmlns="urn:jboss:module:1.1" name="com.mysql">
  <resources>
    <resource-root path="mysql-connector-java-5.0.8-bin.jar"/>
  </resources>
  <dependencies>
    <module name="javax.api"/>
    <module name="javax.transaction.api"/>
  </dependencies>
</module>

```

[Report a bug](#)

6.8.5. Example Oracle Datasource

**NOTE**

Prior to version 10.2 of the Oracle datasource, the `<no-tx-separate-pools/>` parameter was required, as mixing non-transactional and transactional connections would result in an error. This parameter may no longer be required for certain applications.

The example below is an Oracle datasource configuration. The datasource has been enabled, a user has been added, and validation options have been set.

Example 6.25. Oracle datasource configuration

```
<datasources>
  <datasource jndi-name="java:/OracleDS" pool-name="OracleDS">
    <connection-url>jdbc:oracle:thin:@localhost:1521:XE</connection-url>
    <driver>oracle</driver>
    <security>
      <user-name>admin</user-name>
      <password>admin</password>
    </security>
    <validation>
      <validate-on-match>>true</validate-on-match>
      <background-validation>>false</background-validation>
      <valid-connection-checker class-
name="org.jboss.jca.adapters.jdbc.extensions.oracle.OracleValidConnectionChecker"></valid-
connection-checker>
      <stale-connection-checker class-
name="org.jboss.jca.adapters.jdbc.extensions.oracle.OracleStaleConnectionChecker"></stale-
connection-checker>
      <exception-sorter class-
name="org.jboss.jca.adapters.jdbc.extensions.oracle.OracleExceptionSorter"></exception-sorter>
    </validation>
  </datasource>
</drivers>
  <driver name="oracle" module="com.oracle">
  </driver>
</drivers>
</datasources>
```

The example below is a **module.xml** file for the Oracle datasource above.

Example 6.26. module.xml

```
<module xmlns="urn:jboss:module:1.1" name="com.oracle">
  <resources>
    <resource-root path="ojdbc6.jar"/>
  </resources>
  <dependencies>
    <module name="javax.api"/>
    <module name="javax.transaction.api"/>
  </dependencies>
</module>
```

[Report a bug](#)

6.8.6. Example Oracle XA Datasource



NOTE

Prior to version 10.2 of the Oracle datasource, the `<no-tx-separate-pools/>` parameter was required, as mixing non-transactional and transactional connections would result in an error. This parameter may no longer be required for certain applications.



IMPORTANT

The following settings must be applied for the user accessing an Oracle XA datasource in order for XA recovery to operate correctly. The value **user** is the user defined to connect from JBoss to Oracle:

- GRANT SELECT ON sys.dba_pending_transactions TO user;
- GRANT SELECT ON sys.pending_trans\$ TO user;
- GRANT SELECT ON sys.dba_2pc_pending TO user;
- GRANT EXECUTE ON sys.dbms_xa TO user; (If using Oracle 10g R2 (patched) or Oracle 11g)

OR

GRANT EXECUTE ON sys.dbms_system TO user; (If using an unpatched Oracle version prior to 11g)

The example below is an Oracle XA datasource configuration. The datasource has been enabled, a user has been added, and validation options have been set.

Example 6.27. Oracle XA datasource

```

<datasources>
  <xa-datasource jndi-name="java:/XAOracleDS" pool-name="XAOracleDS">
    <driver>oracle</driver>
    <xa-datasource-property name="URL">jdbc:oracle:oci8:@tc</xa-datasource-property>
    <security>
      <user-name>admin</user-name>
      <password>admin</password>
    </security>
    <xa-pool>
      <is-same-rm-override>>false</is-same-rm-override>
      <no-tx-separate-pools />
    </xa-pool>
    <validation>
      <validate-on-match>>true</validate-on-match>
      <background-validation>>false</background-validation>
      <valid-connection-checker class-
name="org.jboss.jca.adapters.jdbc.extensions.oracle.OracleValidConnectionChecker"></valid-
connection-checker>
      <stale-connection-checker class-

```

```

name="org.jboss.jca.adapters.jdbc.extensions.oracle.OracleStaleConnectionChecker"></stale-
connection-checker>
  <exception-sorter class-
name="org.jboss.jca.adapters.jdbc.extensions.oracle.OracleExceptionSorter"></exception-sorter>
</validation>
</xa-datasource>
<drivers>
  <driver name="oracle" module="com.oracle">
    <xa-datasource-class>oracle.jdbc.xa.client.OracleXADataSource</xa-datasource-class>
  </driver>
</drivers>
</datasources>

```

The example below is a **module.xml** file for the Oracle XA datasource above.

Example 6.28. module.xml

```

<module xmlns="urn:jboss:module:1.1" name="com.oracle">
  <resources>
    <resource-root path="ojdbc6.jar"/>
  </resources>
  <dependencies>
    <module name="javax.api"/>
    <module name="javax.transaction.api"/>
  </dependencies>
</module>

```

[Report a bug](#)

6.8.7. Example Microsoft SQLServer Datasource

The example below is a Microsoft SQLServer datasource configuration. The datasource has been enabled, a user has been added, and validation options have been set.

Example 6.29. SQLserver datasource configuration

```

<datasources>
  <datasource jndi-name="java:/MSSQLDS" pool-name="MSSQLDS">
    <connection-url>jdbc:sqlserver://localhost:1433;DatabaseName=MyDatabase</connection-url>
    <driver>sqlserver</driver>
    <security>
      <user-name>admin</user-name>
      <password>admin</password>
    </security>
    <validation>
      <validate-on-match>true</validate-on-match>
      <background-validation>>false</background-validation>
      <valid-connection-checker class-
name="org.jboss.jca.adapters.jdbc.extensions.mssql.MSSQLValidConnectionChecker"></valid-
connection-checker>
      <exception-sorter class-
name="org.jboss.jca.adapters.jdbc.extensions.mssql.MSSQLExceptionSorter"></exception-

```

```

sorter>
  </validation>
</datasource>
<drivers>
  <driver name="sqlserver" module="com.microsoft">
</datasources>

```

The example below is a **module.xml** file for the Microsoft SQLServer datasource above.

Example 6.30. module.xml

```

<module xmlns="urn:jboss:module:1.1" name="com.microsoft">
  <resources>
    <resource-root path="sqljdbc4.jar"/>
  </resources>
  <dependencies>
    <module name="javax.api"/>
    <module name="javax.transaction.api"/>
  </dependencies>
</module>

```

[Report a bug](#)

6.8.8. Example Microsoft SQLServer XA Datasource

The example below is a Microsoft SQLServer XA datasource configuration. The datasource has been enabled, a user has been added, and validation options have been set.

Example 6.31. SQLserver XA datasource

```

<datasources>
  <xa-datasource jndi-name="java:/MSSQLXADS" pool-name="MSSQLXADS">
    <driver>sqlserver</driver>
    <xa-datasource-property name="ServerName">localhost</xa-datasource-property>
    <xa-datasource-property name="DatabaseName">mssqlldb</xa-datasource-property>
    <xa-datasource-property name="SelectMethod">cursor</xa-datasource-property>
    <security>
      <user-name>admin</user-name>
      <password>admin</password>
    </security>
    <xa-pool>
      <is-same-rm-override>>false</is-same-rm-override>
    </xa-pool>
    <validation>
      <validate-on-match>true</validate-on-match>
      <background-validation>>false</background-validation>
      <valid-connection-checker class-
name="org.jboss.jca.adapters.jdbc.extensions.mssql.MSSQLValidConnectionChecker"></valid-
connection-checker>
      <exception-sorter class-
name="org.jboss.jca.adapters.jdbc.extensions.mssql.MSSQLExceptionSorter"></exception-
sorter>

```

```

    </validation>
  </xa-datasource>
</drivers>
  <driver name="sqlserver" module="com.microsoft">
    <xa-datasource-class>com.microsoft.sqlserver.jdbc.SQLServerXADataSource</xa-
datasource-class>
  </driver>
</drivers>
</datasources>

```

The example below is a **module.xml** file for the Microsoft SQLServer XA datasource above.

Example 6.32. module.xml

```

<module xmlns="urn:jboss:module:1.1" name="com.microsoft">
  <resources>
    <resource-root path="sqljdbc4.jar"/>
  </resources>
  <dependencies>
    <module name="javax.api"/>
    <module name="javax.transaction.api"/>
  </dependencies>
</module>

```

[Report a bug](#)

6.8.9. Example IBM DB2 Datasource

The example below is an IBM DB2 datasource configuration. The datasource has been enabled, a user has been added, and validation options have been set.

Example 6.33. IBM DB2 datasource configuration

```

<datasources>
  <datasource jndi-name="java:/DB2DS" pool-name="DB2DS">
    <connection-url>jdbc:db2:ibmdb2db</connection-url>
    <driver>ibmdb2</driver>
    <pool>
      <min-pool-size>0</min-pool-size>
      <max-pool-size>50</max-pool-size>
    </pool>
    <security>
      <user-name>admin</user-name>
      <password>admin</password>
    </security>
    <validation>
      <validate-on-match>true</validate-on-match>
      <background-validation>>false</background-validation>
      <valid-connection-checker class-
name="org.jboss.jca.adapters.jdbc.extensions.db2.DB2ValidConnectionChecker"></valid-
connection-checker>
      <stale-connection-checker class-

```

```

name="org.jboss.jca.adapters.jdbc.extensions.db2.DB2StaleConnectionChecker"></stale-
connection-checker>
  <exception-sorter class-
name="org.jboss.jca.adapters.jdbc.extensions.db2.DB2ExceptionSorter"></exception-sorter>
  </validation>
</datasource>
<drivers>
  <driver name="ibmdb2" module="com.ibm">
  </driver>
</drivers>
</datasources>

```

The example below is a **module.xml** file for the IBM DB2 datasource above.

Example 6.34. module.xml

```

<module xmlns="urn:jboss:module:1.1" name="com.ibm">
  <resources>
    <resource-root path="db2jcc4.jar"/>
  </resources>
  <dependencies>
    <module name="javax.api"/>
    <module name="javax.transaction.api"/>
  </dependencies>
</module>

```

[Report a bug](#)

6.8.10. Example IBM DB2 XA Datasource

The example below is an IBM DB2 XA datasource configuration. The datasource has been enabled, a user has been added and validation options have been set.

Example 6.35. IBM DB2 XA datasource configuration

```

<datasources>
  <xa-datasource jndi-name="java:/DB2XADS" pool-name="DB2XADS">
    <driver>ibmdb2</driver>
    <xa-datasource-property name="DatabaseName">ibmdb2db</xa-datasource-property>
    <xa-datasource-property name="ServerName">hostname</xa-datasource-property>
    <xa-datasource-property name="PortNumber">port</xa-datasource-property>
    <xa-datasource-property name="DriverType">4</xa-datasource-property>
    <security>
      <user-name>admin</user-name>
      <password>admin</password>
    </security>
    <xa-pool>
      <is-same-rm-override>>false</is-same-rm-override>
    </xa-pool>
    <validation>
      <validate-on-match>>true</validate-on-match>
      <background-validation>>false</background-validation>

```

```

    <valid-connection-checker class-
name="org.jboss.jca.adapters.jdbc.extensions.db2.DB2ValidConnectionChecker"></valid-
connection-checker>
    <stale-connection-checker class-
name="org.jboss.jca.adapters.jdbc.extensions.db2.DB2StaleConnectionChecker"></stale-
connection-checker>
    <exception-sorter class-
name="org.jboss.jca.adapters.jdbc.extensions.db2.DB2ExceptionSorter"></exception-sorter>
  </validation>
  <recovery>
    <recover-plugin class-name="org.jboss.jca.core.recovery.ConfigurableRecoveryPlugin">
      <config-property name="EnableIsValid">>false</config-property>
      <config-property name="IsValidOverride">>false</config-property>
      <config-property name="EnableClose">>false</config-property>
    </recover-plugin>
  </recovery>
</xa-datasource>
<drivers>
  <driver name="ibmdb2" module="com.ibm">
    <xa-datasource-class>com.ibm.db2.jcc.DB2XADataSource</xa-datasource-class>
  </driver>
</drivers>
</datasources>

```

The example below is a **module.xml** file for the IBM DB2 XA datasource above.

Example 6.36. module.xml

```

<module xmlns="urn:jboss:module:1.1" name="com.ibm">
  <resources>
    <resource-root path="db2jcc4.jar"/>
    <resource-root path="db2jcc_license_cisuz.jar"/>
    <resource-root path="db2jcc_license_cu.jar"/>
  </resources>
  <dependencies>
    <module name="javax.api"/>
    <module name="javax.transaction.api"/>
  </dependencies>
</module>

```

[Report a bug](#)

6.8.11. Example Sybase Datasource

The example below is a Sybase datasource configuration. The datasource has been enabled, a user has been added, and validation options have been set.

Example 6.37. Sybase datasource configuration

```

<datasources>
  <datasource jndi-name="java:jboss/SybaseDB" pool-name="SybaseDB" enabled="true">
    <connection-url>jdbc:sybase:Tds:localhost:5000/DATABASE?

```

```
JCONNECT_VERSION=6</connection-url>
  <security>
    <user-name>admin</user-name>
    <password>admin</password>
  </security>
  <validation>
    <validate-on-match>true</validate-on-match>
    <background-validation>false</background-validation>
    <valid-connection-checker class-
name="org.jboss.jca.adapters.jdbc.extensions.sybase.SybaseValidConnectionChecker"></valid-
connection-checker>
    <exception-sorter class-
name="org.jboss.jca.adapters.jdbc.extensions.sybase.SybaseExceptionSorter"></exception-
sorter>
  </validation>
</datasource>
<drivers>
  <driver name="sybase" module="com.sybase">
    <datasource-class>com.sybase.jdbc4.jdbc.SybDataSource</datasource-class>
    <xa-datasource-class>com.sybase.jdbc4.jdbc.SybXADataSource</xa-datasource-class>
  </driver>
</drivers>
</datasources>
```

The example below is a **module.xml** file for the Sybase datasource above.

Example 6.38. module.xml

```
<module xmlns="urn:jboss:module:1.1" name="com.sybase">
  <resources>
    <resource-root path="jconn2.jar"/>
  </resources>
  <dependencies>
    <module name="javax.api"/>
    <module name="javax.transaction.api"/>
  </dependencies>
</module>
```

[Report a bug](#)

6.8.12. Example Sybase XA Datasource

The example below is a Sybase XA datasource configuration. The datasource has been enabled, a user has been added, and validation options have been set.

Example 6.39. Sybase XA datasource configuration

```
<datasources>
  <xa-datasource jndi-name="java:jboss/SybaseXADS" pool-name="SybaseXADS"
enabled="true">
    <xa-datasource-property name="NetworkProtocol">Tds</xa-datasource-property>
    <xa-datasource-property name="ServerName">myserver</xa-datasource-property>
```



```

<xa-datasource-property name="PortNumber">4100</xa-datasource-property>
<xa-datasource-property name="DatabaseName">mydatabase</xa-datasource-property>
<security>
  <user-name>admin</user-name>
  <password>admin</password>
</security>
<xa-pool>
  <is-same-rm-override>>false</is-same-rm-override>
</xa-pool>
<validation>
  <validate-on-match>>true</validate-on-match>
  <background-validation>>false</background-validation>
  <valid-connection-checker class-
name="org.jboss.jca.adapters.jdbc.extensions.sybase.SybaseValidConnectionChecker"></valid-
connection-checker>
  <exception-sorter class-
name="org.jboss.jca.adapters.jdbc.extensions.sybase.SybaseExceptionSorter"></exception-
sorter>
</validation>
</xa-datasource>
<drivers>
  <driver name="sybase" module="com.sybase">
    <datasource-class>com.sybase.jdbc4.jdbc.SybDataSource</datasource-class>
    <xa-datasource-class>com.sybase.jdbc4.jdbc.SybXADataSource</xa-datasource-class>
  </driver>
</drivers>
</datasources>

```

The example below is a **module.xml** file for the Sybase XA datasource above.

Example 6.40. module.xml

```

<module xmlns="urn:jboss:module:1.1" name="com.sybase">
  <resources>
    <resource-root path="jconn2.jar"/>
  </resources>
  <dependencies>
    <module name="javax.api"/>
    <module name="javax.transaction.api"/>
  </dependencies>
</module>

```

[Report a bug](#)

CHAPTER 7. CONFIGURING MODULES

7.1. INTRODUCTION

7.1.1. Modules

A Module is a logical grouping of classes used for class loading and dependency management. JBoss EAP 6 identifies two different types of modules, sometimes called static and dynamic modules. However the only difference between the two is how they are packaged.

Static Modules

Static Modules are predefined in the **EAP_HOME/modules/** directory of the application server. Each sub-directory represents one module and defines a **main/** subdirectory that contains a configuration file (**module.xml**) and any required JAR files. The name of the module is defined in the **module.xml** file. All the application server provided APIs are provided as static modules, including the Java EE APIs as well as other APIs such as JBoss Logging.

Example 7.1. Example module.xml file

```
<?xml version="1.0" encoding="UTF-8"?>
<module xmlns="urn:jboss:module:1.0" name="com.mysql">
  <resources>
    <resource-root path="mysql-connector-java-5.1.15.jar"/>
  </resources>
  <dependencies>
    <module name="javax.api"/>
    <module name="javax.transaction.api"/>
  </dependencies>
</module>
```

The module name, **com.mysql**, should match the directory structure for the module, excluding the **main/** subdirectory name.

The modules provided in JBoss EAP distributions are located in a **system** directory within the **EAP_HOME/modules** directory. This keeps them separate from any modules provided by third parties.

Any Red Hat provided layered products that layer on top of JBoss EAP 6.1 or later will also install their modules within the **system** directory.

Creating custom static modules can be useful if many applications are deployed on the same server that use the same third-party libraries. Instead of bundling those libraries with each application, a module containing these libraries can be created and installed by the JBoss administrator. The applications can then declare an explicit dependency on the custom static modules.

Users must ensure that custom modules are installed into the **EAP_HOME/modules** directory, using a one directory per module layout. This ensures that custom versions of modules that already exist in the **system** directory are loaded instead of the shipped versions. In this way, user provided modules will take precedence over system modules.

If you use the **JBOSS_MODULEPATH** environment variable to change the locations in which JBoss EAP searches for modules, then the product will look for a **system** subdirectory structure within one of the locations specified. A **system** structure must exist somewhere in the locations specified with

JBOSS_MODULEPATH.

Dynamic Modules

Dynamic Modules are created and loaded by the application server for each JAR or WAR deployment (or subdeployment in an EAR). The name of a dynamic module is derived from the name of the deployed archive. Because deployments are loaded as modules, they can configure dependencies and be used as dependencies by other deployments.

Modules are only loaded when required. This usually only occurs when an application is deployed that has explicit or implicit dependencies.

[Report a bug](#)

7.1.2. Global Modules

A global module is a module that JBoss EAP 6 provides as a dependency to every application. Any module can be made global by adding it to the application server's list of global modules. It does not require changes to the module.

[Report a bug](#)

7.1.3. Module Dependencies

A module dependency is a declaration that one module requires the classes of another module in order to function. Modules can declare dependencies on any number of other modules. When the application server loads a module, the modular class loader parses the dependencies of that module and adds the classes from each dependency to its class path. If a specified dependency cannot be found, the module will fail to load.

Deployed applications (JAR and WAR) are loaded as dynamic modules and make use of dependencies to access the APIs provided by JBoss EAP 6.

There are two types of dependencies: explicit and implicit.

Explicit Dependencies

Explicit dependencies are declared by the developer in the configuration file. Static modules can declare dependencies in the **module.xml** file. Dynamic modules can have dependencies declared in the **MANIFEST.MF** or **jboss-deployment-structure.xml** deployment descriptors of the deployment.

Explicit dependencies can be specified as optional. Failure to load an optional dependency will not cause a module to fail to load. However if the dependency becomes available later it will NOT be added to the module's class path. Dependencies must be available when the module is loaded.

Implicit Dependencies

Implicit dependencies are added automatically by the application server when certain conditions or meta-data are found in a deployment. The Java EE 6 APIs supplied with JBoss EAP 6 are examples of modules that are added by detection of implicit dependencies in deployments.

Deployments can also be configured to exclude specific implicit dependencies. This is done with the **jboss-deployment-structure.xml** deployment descriptor file. This is commonly done when an application bundles a specific version of a library that the application server will attempt to add as an implicit dependency.

A module's class path contains only its own classes and that of its immediate dependencies. A module is not able to access the classes of the dependencies of one of its dependencies. However a module can specify that an explicit dependency is exported. An exported dependency is provided to any module that depends on the module that exports it.

Example 7.2. Module dependencies

Module A depends on Module B and Module B depends on Module C. Module A can access the classes of Module B, and Module B can access the classes of Module C. Module A cannot access the classes of Module C unless:

- Module A declares an explicit dependency on Module C, or
- Module B exports its dependency on Module C.

See the *Class Loading and Modules* chapter of the *Development Guide* for details on using the **jboss-deployment-structure.xml** deployment descriptor.

[Report a bug](#)

7.1.4. Subdeployment Class Loader Isolation

Each subdeployment in an Enterprise Archive (EAR) is a dynamic module with its own class loader. By default a subdeployment can access the resources of other subdeployments.

If a subdeployment is not to be allowed to access the resources of other subdeployments, strict subdeployment isolation can be enabled.

[Report a bug](#)

7.2. DISABLE SUBDEPLOYMENT MODULE ISOLATION FOR ALL DEPLOYMENTS

This task shows server administrators how to disable Subdeployment Module Isolation on the application server. This affects all deployments.



WARNING

This task requires you to edit the XML configuration files of the server. The server must be halted before doing this. This is temporary as the final release administration tools will support this type of configuration.

1. **Stop the server**
Halt the JBoss EAP 6 server.
2. **Open the server configuration file**
Open the server configuration file in a text editor.

This file will be different for a managed domain or standalone server. In addition, non-default locations and file names may be used. The default configuration files are **domain/configuration/domain.xml** and **standalone/configuration/standalone.xml** for managed domains and standalone servers respectively.

3. Locate the EE Subsystem Configuration

Locate the EE Subsystem configuration element in the configuration file. The **<profile>** element of the configuration file contains several subsystem elements. The EE Subsystem element has the namespace of **urn:jboss:domain:ee:1.2**.

```
<profile>
...
<subsystem xmlns="urn:jboss:domain:ee:1.2" />
...
```

The default configuration has a single self-closing tag but a custom configuration may have separate open and closing tags (possibly with other elements within) like this:

```
<subsystem xmlns="urn:jboss:domain:ee:1.2" ></subsystem>
```

4. Replace self-closing tags if necessary

If the EE Subsystem element is a single self-closing tag then replace with appropriate opening and closing tags like this:

```
<subsystem xmlns="urn:jboss:domain:ee:1.2" ></subsystem>
```

5. Add ear-subdeployments-isolated element

Add the **ear-subdeployments-isolated** element as a child of the EE Subsystem element and add the content of **false** like this:

```
<subsystem xmlns="urn:jboss:domain:ee:1.2" ><ear-subdeployments-isolated>>false</ear-
subdeployments-isolated></subsystem>
```

6. Start the server

Relaunch the JBoss EAP 6 server to start it running with the new configuration.

Result:

The server will now be running with Subdeployment Module Isolation disabled for all deployments.

[Report a bug](#)

7.3. ADD A MODULE TO ALL DEPLOYMENTS

This task shows how JBoss administrators can define a list of global modules.

Prerequisites

1. You must know the name of the modules that are to be configured as global modules. Refer to [Section 7.6.1, "Included Modules"](#) for the list of static modules included with JBoss EAP 6. If the module is in another deployment, refer to [Section 7.6.2, "Dynamic Module Naming"](#) to

determine the module name.

Procedure 7.1. Add a module to the list of global modules

1. Login to the management console. [Section 3.3.2, “Log in to the Management Console”](#)
2. Navigate to the **EE Subsystem** panel.
 - a. Select the **Configuration** tab from the top of the console.
 - b. **Domain Mode Only**
 - i. Select the appropriate profile from the drop-down box in the top left.
 - c. Expand the **Subsystems** menu on the left of the console.
 - d. Select **Container** → **EE** from the menu on the left of the console.
3. Click **Add** in the **Subsystem Defaults** section. The **Create Module** dialog appears.
4. Type in the name of the module and optionally the module slot.
5. Click **Save** to add the new global module, or click **Cancel** to abort.
 - If you click **Save**, the dialog will close and the specified module will be added to the list of global modules.
 - If you click **Cancel**, the dialog will close and no changes will be made.

Result

The modules added to the list of global modules will be added as dependencies to every deployment.

[Report a bug](#)

7.4. CREATE A CUSTOM MODULE

The following procedure describes how to create a custom module in order to make properties files and other resources available to all applications running on the JBoss EAP server.

Procedure 7.2. Create a Custom Module

1. Create and populate the **module/** directory structure.
 - a. Create a directory structure under the **EAP_HOME/module** directory to contain the files and JARs. For example:

```
$ cd EAP_HOME/modules/  
$ mkdir -p myorg-conf/main/properties
```

- b. Move the properties files to the **EAP_HOME/modules/myorg-conf/main/properties/** directory you created in the previous step.
- c. Create a **module.xml** file in the **EAP_HOME/modules/myorg-conf/main/** directory containing the following XML:

```
<module xmlns="urn:jboss:module:1.1" name="myorg-conf">
  <resources>
    <resource-root path="properties"/>
  </resources>
</module>
```

2. Modify the **ee** subsystem in the server configuration file. You can use the Management CLI or you can manually edit the file.

- o Follow these steps to modify the server configuration file using the Management CLI.
 - a. Start the server and connect to the Management CLI.

- For Linux, enter the following at the command line:

```
EAP_HOME/bin/jboss-cli.sh --connect
```

- For Windows, enter the following at a command line:

```
C:\>EAP_HOME\bin\jboss-cli.bat --connect
```

You should see the following response:

```
Connected to standalone controller at localhost:9999
```

- b. To create the **myorg-conf** <global-modules> element in the **ee** subsystem, type the following in the command line:

```
/subsystem=ee:write-attribute(name=global-modules, value=[{"name"=>"myorg-conf", "slot"=>"main"}])
```

You should see the following result:

```
{"outcome" => "success"}
```

- o Follow these steps if you prefer to manually edit the server configuration file.
 - a. Stop the server and open the server configuration file in a text editor. If you are running a standalone server, this is the **EAP_HOME/standalone/configuration/standalone.xml** file, or the **EAP_HOME/domain/configuration/domain.xml** file if you are running a managed domain.
 - b. Find the **ee** subsystem and add the global module for **myorg-conf**. The following is an example of the **ee** subsystem element, modified to include the **myorg-conf** element:

Example 7.3. myorg-conf element

```
<subsystem xmlns="urn:jboss:domain:ee:1.0" >
  <global-modules>
    <module name="myorg-conf" slot="main" />
  </global-modules>
</subsystem>
```

- 3. Assuming you copied a file named **my.properties** into the correct module location, you are now able to load properties files using code similar to the following:

Example 7.4. Load properties file

```
Thread.currentThread().getContextClassLoader().getResource("my.properties");
```

[Report a bug](#)

7.5. DEFINE AN EXTERNAL JBOSS MODULE DIRECTORY

Summary

By default, JBoss EAP looks for modules in the **EAP_HOME/modules/** directory. You can direct JBoss EAP to look in one or more external directories by defining a **JBOSS_MODULEPATH** environment variable or by setting the variable in the startup configuration file. This topic describes both methods.

Procedure 7.3. Set the JBOSS_MODULEPATH Environment Variable

- To specify one or more external module directories, define the **JBOSS_MODULEPATH** environment variable.

For Linux, use a colon to delimit a list of directories. For example:

Example 7.5. JBOSS_MODULEPATH environment variable

```
export
JBOSS_MODULEPATH=EAP_HOME/modules/:/home/username/external/modules/directo
ry/
```

For Windows, use a semicolon to delimit a list of directories. For example:

Example 7.6. JBOSS_MODULEPATH environment variable

```
SET JBOSS_MODULEPATH=EAP_HOME\modules\;D:\JBoss-Modules\
```

Procedure 7.4. Set the JBOSS_MODULEPATH Variable in the Startup Configuration File

- If you prefer not to set a global environment variable, you can set the **JBOSS_MODULEPATH** variable in the JBoss EAP startup configuration file. If you are running a standalone server, this is the **EAP_HOME/bin/standalone.conf** file. If the server is running in a managed domain, this is the **EAP_HOME/bin/domain.conf** file.

The following is an example of the command that sets the **JBOSS_MODULEPATH** variable in the **standalone.conf** file:

Example 7.7. standalone.conf entry


```
JBOSS_MODULEPATH="EAP_HOME/modules/:/home/username/external/modules/direct  
ory/"
```

[Report a bug](#)

7.6. REFERENCE

7.6.1. Included Modules

A table listing the JBoss EAP 6 included modules and whether they are supported can be found on the Customer Portal at <https://access.redhat.com/articles/1122333>.

[Report a bug](#)

7.6.2. Dynamic Module Naming

All deployments are loaded as modules by JBoss EAP 6 and named according to the following conventions.

- Deployments of WAR and JAR files are named with the following format:

```
deployment.DEPLOYMENT_NAME
```

For example, **inventory.war** and **store.jar** will have the module names of **deployment.inventory.war** and **deployment.store.jar** respectively.

- Subdeployments within an Enterprise Archive are named with the following format:

```
deployment.EAR_NAME.SUBDEPLOYMENT_NAME
```

For example, the subdeployment of **reports.war** within the enterprise archive **accounts.ear** will have the module name of **deployment.accounts.ear.reports.war**.

[Report a bug](#)

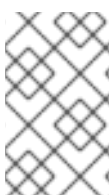
CHAPTER 8. JSVC

8.1. INTRODUCTION

8.1.1. About Jsvc

Jsvc is a set of libraries and applications which allow Java applications run on UNIX and UNIX-like platforms as a background service. It allows an application to perform operations as a privileged user, and then switch identity to a non-privileged user.

Jsvc uses three processes: a launcher process, a controller process and a controlled process. The controlled process is also the main Java thread. If the JVM crashes the controller process will restart it within 60 seconds. Jsvc is a daemon process and for JBoss EAP 6 it must be started by a privileged user.



NOTE

Jsvc is for use on Red Hat Enterprise Linux, Solaris and HP-UX only. For similar functionality on Microsoft Windows, see **prunsvr.exe** in the **Native Utilities for Windows Server** download available from the Red Hat Customer Portal.

[Report a bug](#)

8.1.2. Start and Stop JBoss EAP using Jsvc

The instructions for starting and stopping JBoss EAP using Jsvc vary, depending on which mode it is operating: standalone or domain. Be aware that if JBoss EAP is run in domain mode, Jsvc handles the process of the domain controller only. Whichever command you use to start JBoss EAP using Jsvc, it must be run by a privileged user.

Prerequisites

- If JBoss EAP was installed using the Zip method:
 - Install the *Native Utilities* package for your operating system, available for download from the Red Hat Customer Portal. See *Install Native Components and Native Utilities (Zip, Installer)* in the *Installation Guide*.
 - Create the user account under which the JBoss EAP 6 instance will run. The account used to start and stop the server must have read and write access to the directory in which JBoss EAP was installed.
- If JBoss EAP was installed using the RPM method, install the *apache-commons-daemon-jsvc-eap6* package. See *Install Native Components and Native Utilities (RPM Installation)* in the *Installation Guide*.

The following commands are to start and stop JBoss EAP in either standalone or domain modes. Note that file locations are different depending on the method used to install Jsvc in JBoss EAP 6. Use the tables below to determine which files to use to resolve the variables in the commands.

Standalone Mode

The following instructions are to start or stop JBoss EAP in standalone mode.

Table 8.1. Jsvc File locations For Zip installations - Standalone Mode

File Reference in Instructions	File Location
EAP-HOME	<code>\${eap-installation-location}/jboss-eap-\${version}</code>
JSVC-BIN	<code>EAP_HOME/modules/system/layers/base/native/sbin/jsvc</code>
JSVC-JAR	<code>EAP_HOME/modules/system/layers/base/native/sbin/commons-daemon.jar</code>
CONF-DIR	<code>EAP_HOME/standalone/configuration</code>
LOG-DIR	<code>EAP_HOME/standalone/log</code>

Table 8.2. Jsvc File Locations for RPM Installations - Standalone Mode

File Reference in Instructions	File Location
EAP-HOME	<code>/usr/share/jbossas</code>
JSVC-BIN	<code>/usr/bin/jsvc-eap6/jsvc</code>
JSVC-JAR	<code>EAP_HOME/modules/system/layers/base/native/sbin/commons-daemon.jar</code>
CONF-DIR	<code>/etc/jbossas/standalone</code>
LOG-DIR	<code>/var/log/jbossas/standalone</code>

Start JBoss EAP in Standalone Mode

- ```

JSVC_BIN \
-outfile LOG_DIR/jsvc.out.log \
-errfile LOG_DIR/jsvc.err.log \
-pidfile LOG_DIR/jsvc.pid \
-user jboss \
-D[Standalone] -XX:+UseCompressedOops -Xms1303m \
-Xmx1303m -XX:MaxPermSize=256m \
-Djava.net.preferIPv4Stack=true
-Djboss.modules.system.pkgs=org.jboss.byteman \
-Djava.awt.headless=true \
-Dorg.jboss.boot.log.file=LOG_DIR/server.log \
-Dlogging.configuration=file:CONF_DIR/logging.properties \
-Djboss.modules.policy-permissions \
-cp EAP_HOME/jboss-modules.jar:JSVC_JAR \
-Djboss.home.dir=EAP_HOME \
-Djboss.server.base.dir=EAP_HOME/standalone \
@org.jboss.modules.Main -start-method main \

```

```
-mp EAP_HOME/modules \
-jaxpmodule javax.xml.jaxp-provider \
org.jboss.as.standalone
```

### Stop JBoss EAP in Standalone Mode

- ```
JSVC_BIN \
-stop \
-outfile LOG_DIR/jsvc.out.log \
-errfile LOG_DIR/jsvc.err.log \
-pidfile LOG_DIR/jsvc.pid \
-user jboss \
-D[Standalone] -XX:+UseCompressedOops -Xms1303m \
-Xmx1303m -XX:MaxPermSize=256m \
-Djava.net.preferIPv4Stack=true \
-Djboss.modules.system.pkgs=org.jboss.byteman \
-Djava.awt.headless=true \
-Dorg.jboss.boot.log.file=LOG_DIR/server.log \
-Dlogging.configuration=file:CONF_DIR/logging.properties \
-Djboss.modules.policy-permissions \
-cp EAP_HOME/jboss-modules.jar:JSVC_JAR \
-Djboss.home.dir=EAP_HOME \
-Djboss.server.base.dir=EAP_HOME/standalone \
@org.jboss.modules.Main -start-method main \
-mp EAP_HOME/modules \
-jaxpmodule javax.xml.jaxp-provider \
org.jboss.as.standalone
```

Domain Mode

The following instructions are to start or stop JBoss EAP in domain mode. Note that for domain mode, you must replace the `JAVA_HOME` variable with the Java home directory.

Table 8.3. Jsvc File Locations for Zip Installations - Domain Mode

File Reference in Instructions	File Location
EAP-HOME	<code>\${eap-installation-location}/jboss-eap-\${version}</code>
JSVC-BIN	<code>EAP_HOME/modules/system/layers/base/native/sbin/jsvc</code>
JSVC-JAR	<code>EAP_HOME/modules/system/layers/base/native/sbin/commons-daemon.jar</code>
CONF-DIR	<code>EAP_HOME/domain/configuration</code>
LOG-DIR	<code>EAP_HOME/domain/log</code>

Table 8.4. Jsvc File Locations for RPM Installations - Domain Mode

File Reference in Instructions	File Location
EAP-HOME	/usr/share/jbossas
JSVC-BIN	/usr/bin/jsvc-eap6/jsvc
JSVC-JAR	EAP_HOME/modules/system/layers/base/native/sbin/commons-daemon.jar
CONF-DIR	/etc/jbossas/domain
LOG-DIR	/var/log/jbossas/domain

Start JBoss EAP in Domain Mode

- ```

JSVC_BIN \
-outfile LOG_DIR/jsvc.out.log \
-errfile LOG_DIR/jsvc.err.log \
-pidfile LOG_DIR/jsvc.pid \
-user jboss \
-nodetach -D"[Process Controller]" -server -Xms64m \
-Xmx512m -XX:MaxPermSize=256m \
-Djava.net.preferIPv4Stack=true \
-Djboss.modules.system.pkgs=org.jboss.byteman \
-Djava.awt.headless=true \
-Dorg.jboss.boot.log.file=LOG_DIR/process-controller.log \
-Dlogging.configuration=file:CONF_DIR/logging.properties \
-Djboss.modules.policy-permissions \
-cp "EAP_HOME/jboss-modules.jar:JSVC_JAR" \
org.apache.commons.daemon.support.DaemonWrapper \
-start org.jboss.modules.Main -start-method main \
-mp EAP_HOME/modules org.jboss.as.process-controller \
-jboss-home EAP_HOME -jvm $JAVA_HOME/bin/java \
-mp EAP_HOME/modules -- \
-Dorg.jboss.boot.log.file=LOG_DIR/host-controller.log \
-Dlogging.configuration=file:CONF_DIR/logging.properties \
-Djboss.modules.policy-permissions \
-server -Xms64m -Xmx512m -XX:MaxPermSize=256m \
-Djava.net.preferIPv4Stack=true \
-Djboss.modules.system.pkgs=org.jboss.byteman \
-Djava.awt.headless=true -- -default-jvm $JAVA_HOME/bin/java

```

### Stop JBoss EAP in Domain Mode

- ```

JSVC_BIN \
-stop \
-outfile LOG_DIR/jsvc.out.log \
-errfile LOG_DIR/jsvc.err.log \
-pidfile LOG_DIR/jsvc.pid \
-user jboss \
-nodetach -D"[Process Controller]" -server -Xms64m \

```

```
-Xmx512m -XX:MaxPermSize=256m \  
-Djava.net.preferIPv4Stack=true \  
-Djboss.modules.system.pkgs=org.jboss.byteman \  
-Djava.awt.headless=true \  
-Dorg.jboss.boot.log.file=LOG_DIR/process-controller.log \  
-Dlogging.configuration=file:CONF_DIR/logging.properties \  
-Djboss.modules.policy-permissions \  
-cp "EAP_HOME/jboss-modules.jar:JSVC_JAR" \  
org.apache.commons.daemon.support.DaemonWrapper \  
-start org.jboss.modules.Main -start-method main \  
-mp EAP_HOME/modules org.jboss.as.process-controller \  
-jboss-home EAP_HOME -jvm $JAVA_HOME/bin/java \  
-mp EAP_HOME/modules -- \  
-Dorg.jboss.boot.log.file=LOG_DIR/host-controller.log \  
-Dlogging.configuration=file:CONF_DIR/logging.properties \  
-Djboss.modules.policy-permissions \  
-server -Xms64m -Xmx512m -XX:MaxPermSize=256m \  
-Djava.net.preferIPv4Stack=true \  
-Djboss.modules.system.pkgs=org.jboss.byteman \  
-Djava.awt.headless=true -- -default-jvm $JAVA_HOME/bin/java
```



NOTE

If JBoss EAP 6 is terminated abnormally, such as a JVM crash, Jsvc will automatically restart it. If JBoss EAP 6 is terminated correctly, Jsvc will also stop.

[Report a bug](#)

CHAPTER 9. GLOBAL VALVES

9.1. ABOUT VALVES

A Valve is a Java class that gets inserted into the request processing pipeline for an application. It is inserted in the pipeline before servlet filters. Valves can make changes to the request before passing it on or perform other processing such as authentication or even canceling the request.

Valves can be configured at the server level or at the application level. The only difference is in how they are configured and packaged.

- Global Valves are configured at the server level and apply to all applications deployed to the server. Instructions to configure Global Valves are located in the *Administration and Configuration Guide* for JBoss EAP.
- Valves configured at the application level are packaged with the application deployment and only affect the specific application. Instructions to configure Valves at the application level are located in the *Development Guide* for JBoss EAP.

Version 6.1.0 and later supports global valves.

[Report a bug](#)

9.2. ABOUT GLOBAL VALVES

A Global Valve is a valve that is inserted into the request processing pipeline of all deployed applications. A valve is made global by being packaged and installed as a static module in JBoss EAP 6. Global valves are configured in the web subsystem.

Only version 6.1.0 and later supports global valves.

For instructions on how to configure Global Valves, see [Section 9.5, “Configure a Global Valve”](#).

[Report a bug](#)

9.3. ABOUT AUTHENTICATOR VALVES

An authenticator valve is a valve that authenticates the credentials of a request. Such valve is a subclass of **org.apache.catalina.authenticator.AuthenticatorBase** and overrides the **authenticate(Request request, Response response, LoginConfig config)** method.

This can be used to implement additional authentication schemes.

[Report a bug](#)

9.4. INSTALL A GLOBAL VALVE

Global valves must be packaged and installed as static modules in JBoss EAP 6. This task shows how to install the module.

Pre-requisites:

- The valve must already be created and packaged in a JAR file.

- A **module.xml** file must already be created for the module.

Refer to [Section 7.1.1, “Modules”](#) for an example of **module.xml** file.

Procedure 9.1. Install a Global Module

1. Create module installation directory

A directory for the module to be installed in must be created in the modules directory of the application server.

```
EAP_HOME/modules/system/layers/base/MODULENAME/main
```

```
$ mkdir -P EAP_HOME/modules/system/layers/base/MODULENAME/main
```

2. Copy files

Copy the JAR and **module.xml** files to the directory created in step 1.

```
$ cp MyValves.jar module.xml  
EAP_HOME/modules/system/layers/base/MODULENAME/main
```

The valve classes declared in the module are now available to be configured in the web subsystem.

[Report a bug](#)

9.5. CONFIGURE A GLOBAL VALVE

Global valves are enabled and configured in the web subsystem. This is done using the JBoss CLI tool.

Procedure 9.2. Configure a Global Valve

1. Enable the Valve

Use the **add** operation to add a new valve entry.

```
/subsystem=web/valve=VALVENAME:add(module="MODULENAME",class-  
name="CLASSNAME")
```

You need to specify the following values:

- **VALVENAME**, the name that is used to refer to this valve in application configuration.
- **MODULENAME**, the module that contains the value being configured.
- **CLASSNAME**, the classname of the specific valve in the module.

For example:

```
/subsystem=web/valve=clientlimiter:add(module="clientlimitermodule",class-  
name="org.jboss.samplevalves.RestrictedUserAgentsValve")
```

2. Optionally: Specify Parameters

If the valve has configuration parameters, specify these with the **add-param** operation.


```
/subsystem=web/valve=VALVENAME:add-param(param-name="PARAMNAME", param-  
value="PARAMVALUE")
```

You need to specify the following values:

- **VALVENAME**, the name that is used to refer to this valve in application configuration.
- **PARAMNAME**, the name of the parameter that is being configured for specific valve.
- **PARAMVALUE**, the value of the specified parameter.

For example:

Example 9.1. Valve configuration

```
/subsystem=web/valve=clientlimiter:add-param(  
  param-name="restrictedUserAgents",  
  param-value="^.*MS Web Services Client Protocol.*$" )
```

The valve is now enabled and configured for all deployed applications.

Refer to *Create a Custom Valve* section of the *Development Guide* for more information on how to create a custom valve.

[Report a bug](#)

CHAPTER 10. APPLICATION DEPLOYMENT

10.1. ABOUT APPLICATION DEPLOYMENT

JBoss EAP 6 features a range of application deployment and configuration options to cater to both administrative and development environments. For administrators, the Management Console and the Management CLI offer the ideal graphical and command line interfaces to manage application deployments in a production environment. For developers, the range of application deployment testing options include a highly configurable filesystem deployment scanner, the use of an IDE such as JBoss Developer Studio, or deployment and undeployment via Maven.

Administration

- **Management Console**
 - [Section 10.2.2, "Enable a Deployed Application Using the Management Console"](#)
 - [Section 10.2.3, "Disable a Deployed Application Using the Management Console"](#)

- **Management CLI**
 - [Section 10.3.4, "Deploy an Application in a Managed Domain Using the Management CLI"](#)
 - [Section 10.3.2, "Deploy an Application in a Standalone Server Using the Management CLI"](#)
 - [Section 10.3.5, "Undeploy an Application in a Managed Domain Using the Management CLI"](#)
 - [Section 10.3.3, "Undeploy an Application in a Standalone Server Using the Management CLI"](#)
 - [Section 10.3.1, "Manage Application Deployment in the Management CLI"](#)

Development

- **Deployment Scanner**
 - [Section 10.5.7, "Configure the Deployment Scanner"](#)
 - [Section 10.5.2, "Deploy an Application to a Standalone Server Instance with the Deployment Scanner"](#)
 - [Section 10.5.3, "Undeploy an Application from a Standalone Server Instance with the Deployment Scanner"](#)
 - [Section 10.5.4, "Redeploy an Application to a Standalone Server Instance with the Deployment Scanner"](#)
 - [Section 10.5.8, "Configure the Deployment Scanner with the Management CLI"](#)
 - [Section 10.5.6, "Reference for Deployment Scanner Attributes"](#)
 - [Section 10.5.5, "Reference for Deployment Scanner Marker Files"](#)

- **Maven**

- [Section 10.6.2, “Deploy an Application with Maven”](#)
- [Section 10.6.3, “Undeploy an Application with Maven”](#)

NOTE

Before deploying any application, you can enable validation for deployment descriptors. This can be done by setting the **org.jboss.metadata.parser.validate** system property to **true**. This can be done in one of the two ways:

- While starting the server.

Example:

- For domain mode:

```
./domain.sh -Dorg.jboss.metadata.parser.validate=true
```

- For standalone mode:

```
./standalone.sh -Dorg.jboss.metadata.parser.validate=true
```

- By defining it in the server configuration.

For more information on configuring system properties using the Management CLI, refer [Section 3.5.11, “Configure System Properties Using the Management CLI”](#)

[Report a bug](#)

10.2. DEPLOY WITH THE MANAGEMENT CONSOLE

10.2.1. Manage Application Deployment in the Management Console

Deploying applications via the Management Console gives you the benefit of a graphical interface that is easy to use. You can see at a glance what applications are deployed to your server or server groups, and you can enable, disable or delete applications from the content repository as required.

[Report a bug](#)

10.2.2. Enable a Deployed Application Using the Management Console

Prerequisites

- [Section 3.3.2, “Log in to the Management Console”](#)
- [Section 3.3.8, “Add a Deployment in the Management Console”](#)

Procedure 10.1. Enable a Deployed Application using the Management Console

- Select the **Deployments** tab from the top of the console.

The deployment method for applications will differ depending on whether you are deploying to a standalone server instance or a managed domain.

- **Enable an application on a standalone server instance**

The **Available Deployments** table shows all available application deployments and their status.

- To enable an application in a standalone server instance, select the application, then click **En/Disable**.

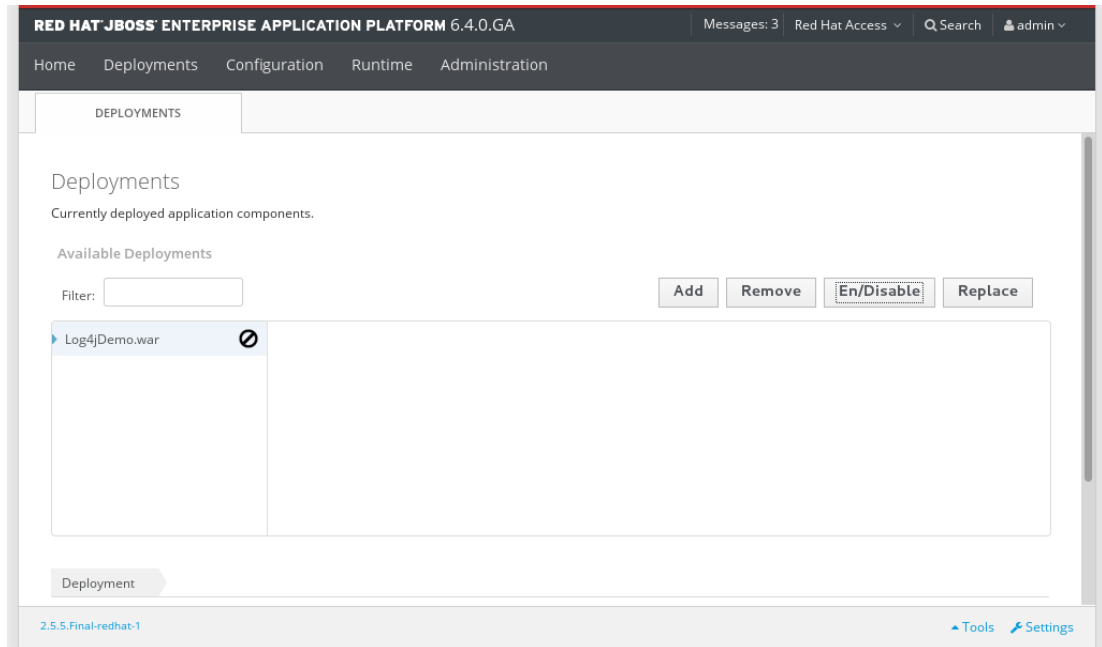


Figure 10.1. Available deployments

- Click **Confirm** to confirm that the application will be enabled on the server instance.

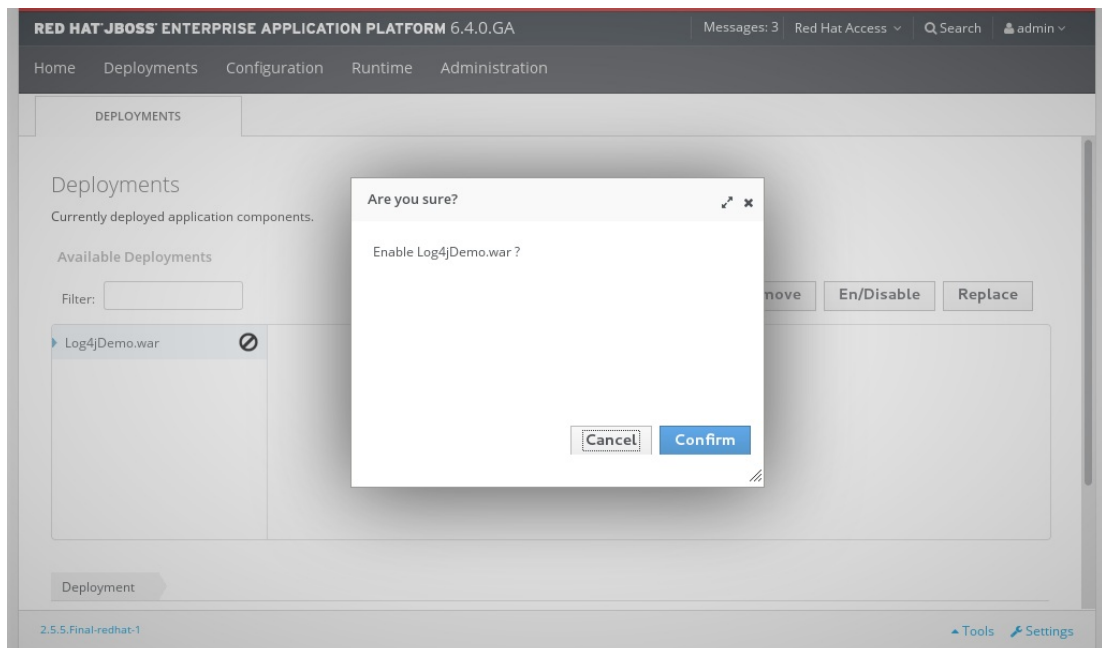


Figure 10.2. Available deployments in a standalone server

- **Enable an application in a managed domain**

The **Content Repository** tab contains an **Available Deployment Content** table showing all available application deployments and their status.

- a. To enable an application in a Managed Domain, select the application to be deployed. Click **Assign** above the **Available Deployment Content** table.

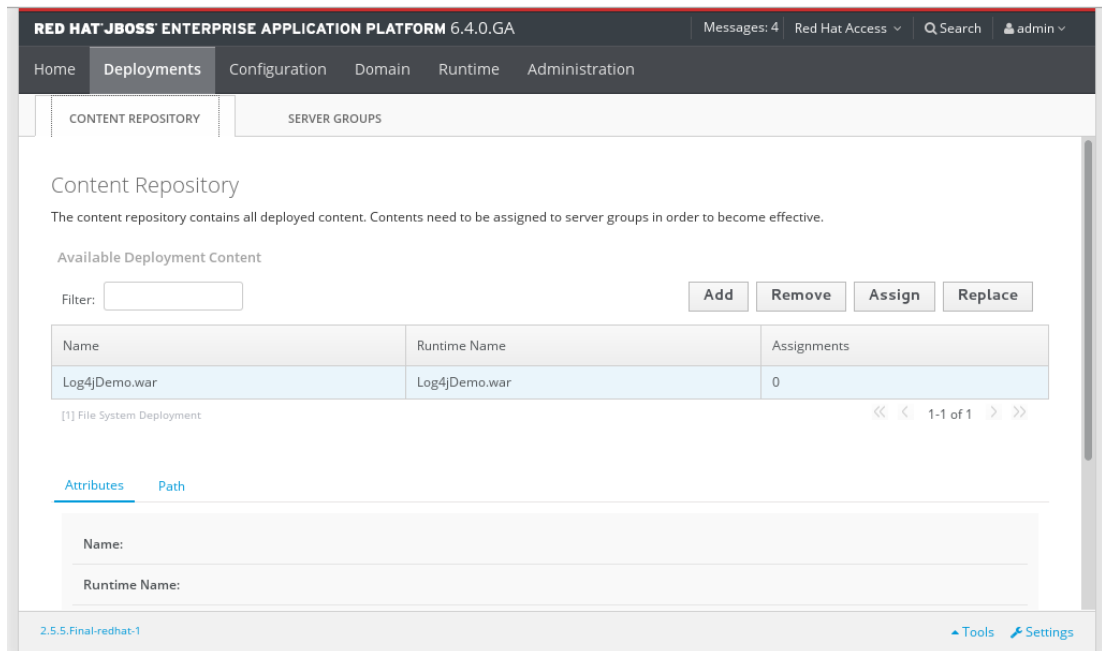


Figure 10.3. Available deployments in a managed domain

- b. Check the boxes for each of the server groups that you want the application to be added to and click **Save** to finish.
- c. Select **Server Groups** tab to view the **Server Groups** table.

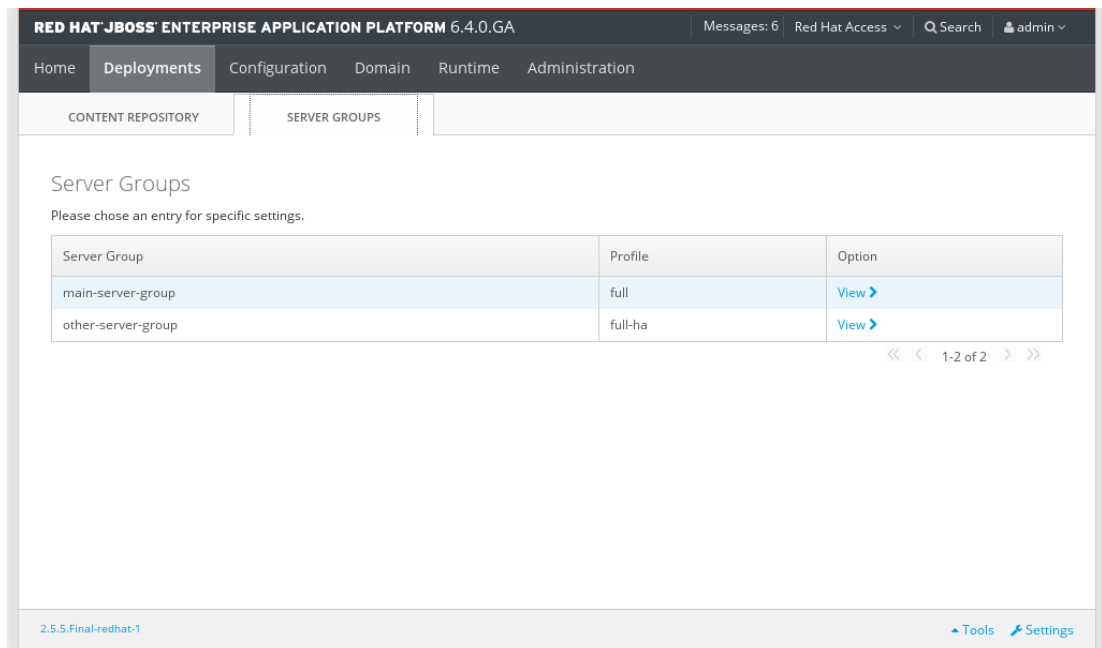


Figure 10.4. Confirmation of application deployment to server groups

- d. If your application is not already enabled, you can enable it now by clicking on **View** and then clicking on the **En/Disable** button. Click **Confirm** to confirm that the application will be enabled on the server instance.

Result

The application is enabled on the relevant server or server group.

[Report a bug](#)

10.2.3. Disable a Deployed Application Using the Management Console

Prerequisites

- [Section 3.3.2, “Log in to the Management Console”](#)
- [Section 3.3.8, “Add a Deployment in the Management Console”](#)
- [Section 10.2.2, “Enable a Deployed Application Using the Management Console”](#)

Procedure 10.2. Disable a Deployed Application using the Management Console

- Select the **Deployment** tab from the top of the console.

The method used to disable an application will differ depending on whether you are deploying to a standalone server instance or a managed domain.

- **Disable a deployed application on a Standalone server instance**

The **Available Deployments** table shows all available application deployments and their status.

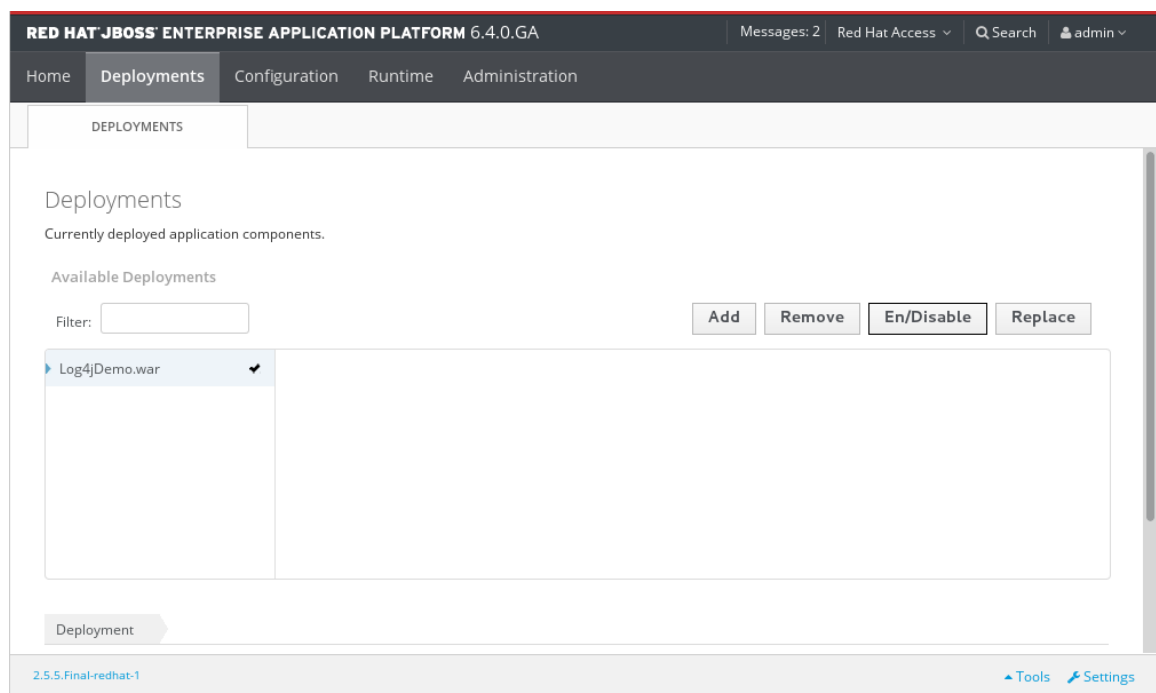


Figure 10.5. Available deployments

- Select the application to be disabled. Click **En/Disable** to disable the selected application.
 - Click **Confirm** to confirm that the application will be disabled on the server instance.
- **Disable a deployed application on a managed domain**

The **Deployments** screen contains a **Content Repository** tab. The **Available Deployment Content** table shows all available application deployments and their status.

- a. Select the **Server Groups** tab to view the server groups.

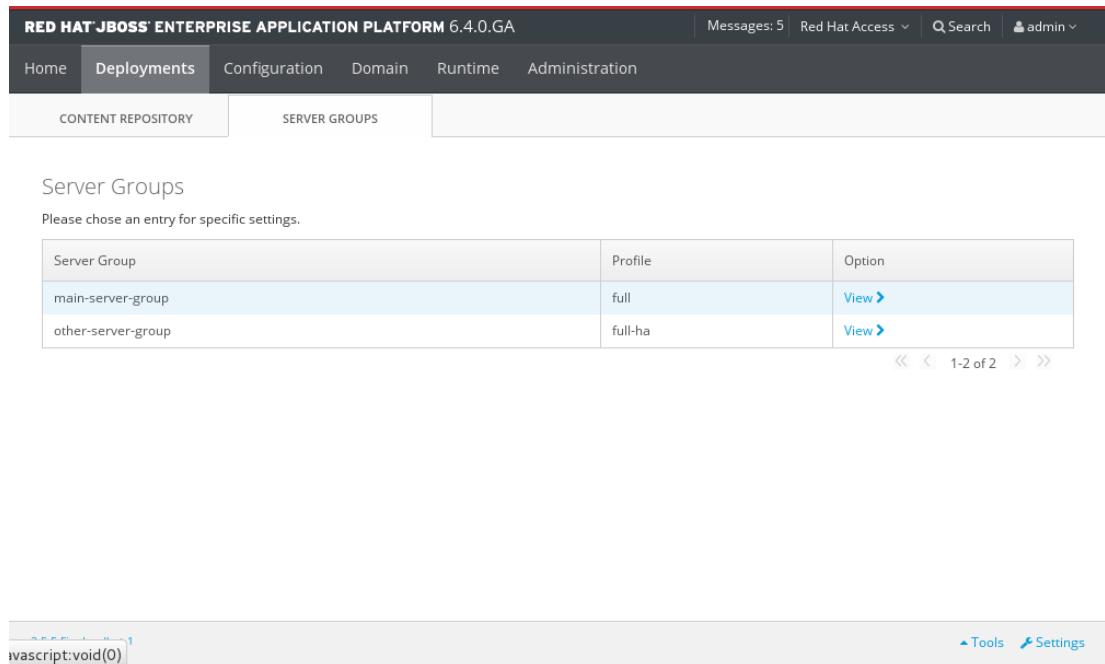


Figure 10.6. Server group deployments

- b. Select the name of the server group in the **Server Group** table to undeploy an application from. Click **View** to see the applications.
- c. Select the application and click **En/Disable** to disable the application for the selected server.
- d. Click **Confirm** to confirm that the application will be disabled on the server instance.
- e. Repeat as required for other server groups. The application status is confirmed for each server group in the **Group Deployments** table for that server group.

Result

The application is disabled from the relevant server or server group.

[Report a bug](#)

10.2.4. Undeploy an Application Using the Management Console

Prerequisites

- [Section 3.3.2, “Log in to the Management Console”](#)
- [Section 3.3.8, “Add a Deployment in the Management Console”](#)
- [Section 10.2.2, “Enable a Deployed Application Using the Management Console”](#)
- [Section 10.2.3, “Disable a Deployed Application Using the Management Console”](#)

Procedure 10.3. Undeploy an Application Using the Management Console

- Select the **Deployments** tab from the top of the console.

The method used to undeploy an application will differ depending on whether you are undeploying from a standalone server instance or a managed domain.

- **Undeploy a deployed application from a Standalone server instance**

The **Available Deployments** table shows all available application deployments and their status.

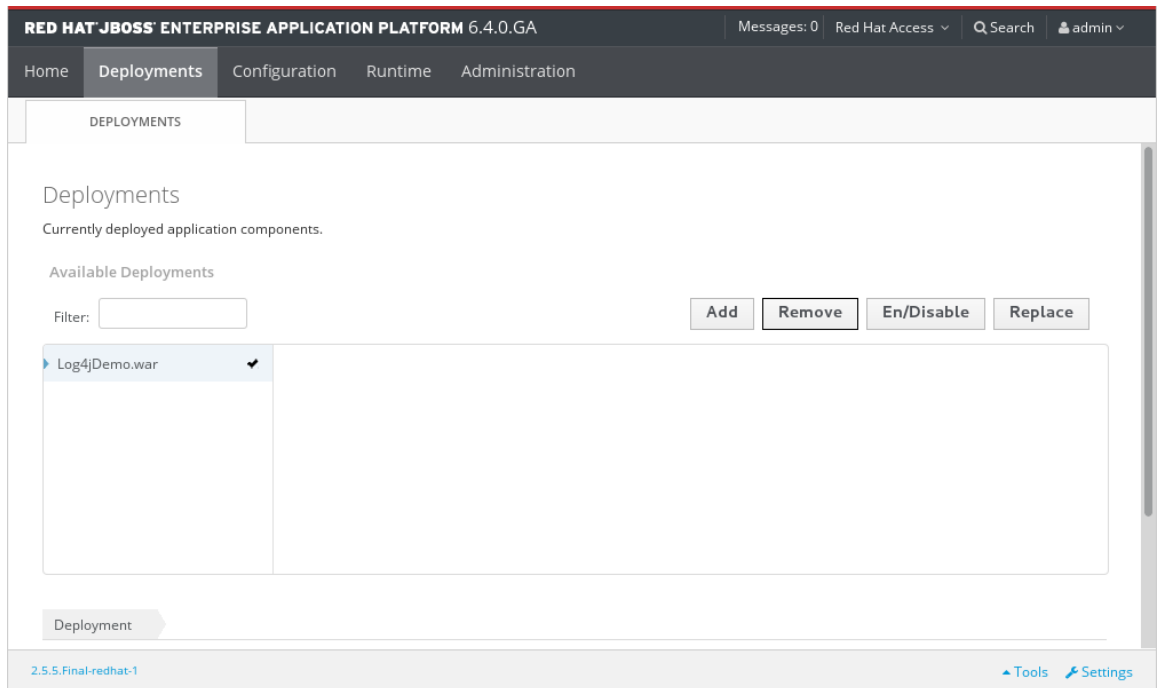


Figure 10.7. Available deployments

- Select the application to be undeployed. Click **Remove** to undeploy the selected application.
 - Click **Confirm** to confirm that the application will be undeployed on the server instance.
- **Undeploy a deployed application from a managed domain**
The **Deployments** screen contains a **Content Repository** tab. The **Available Deployment Content** table shows all available application deployments and their status.
 - Select the **Server Groups** tab to view the server groups and the status of their deployed applications.

RED HAT JBOSS ENTERPRISE APPLICATION PLATFORM 6.4.0.GA Messages: 5 Red Hat Access Search admin

Home Deployments Configuration Domain Runtime Administration

CONTENT REPOSITORY SERVER GROUPS

Server Groups

Please chose an entry for specific settings.

Server Group	Profile	Option
main-server-group	full	View
other-server-group	full-ha	View

1-2 of 2

Tools Settings

Figure 10.8. Server group deployments

- Select the name of the server group in the **Server Group** table to undeploy an application from. Click **View** to see the applications.
- Select the application and click **Remove** to undeploy the application for the selected server.
- Click **Confirm** to confirm that the application will be undeployed on the server instance.
- Repeat as required for other server groups. The application status is confirmed for each server group in the **Group Deployments** table for that server group.

Result

The application is undeployed from the relevant server or server group. On a standalone instance the deployment content is also removed. On a managed domain, the deployment content remains in the content repository and is only undeployed from the server group.

[Report a bug](#)

10.3. DEPLOY WITH THE MANAGEMENT CLI

10.3.1. Manage Application Deployment in the Management CLI

Deploying applications via the Management CLI gives you the benefit of single command line interface with the ability to create and run deployment scripts. You can use this scripting ability to configure specific application deployment and management scenarios. You can manage the deployment status of a single server in the case of a standalone instance, or an entire network of servers in the case of a managed domain.

[Report a bug](#)

10.3.2. Deploy an Application in a Standalone Server Using the Management CLI

Prerequisites

- [Section 3.4.2, “Launch the Management CLI”](#)
- [Section 3.4.4, “Connect to a Managed Server Instance Using the Management CLI”](#)

Procedure 10.4. Deploy an Application in a Standalone Server

- **Run the `deploy` command**

From the Management CLI, enter the **deploy** command with the path to the application deployment.

Example 10.1. The Deploy command

```
[standalone@localhost:9999 /] deploy /path/to/test-application.war
```

Note that a successful deploy does not produce any output to the CLI.

Result

The specified application is now deployed in the standalone server.

[Report a bug](#)

10.3.3. Undeploy an Application in a Standalone Server Using the Management CLI

Prerequisites

- [Section 3.4.2, “Launch the Management CLI”](#)
- [Section 3.4.4, “Connect to a Managed Server Instance Using the Management CLI”](#)
- [Section 10.3.2, “Deploy an Application in a Standalone Server Using the Management CLI”](#)

Procedure 10.5. Undeploy an Application in a Standalone Server

By default the **undeploy** command will undeploy *and* delete the deployment content from a standalone instance of JBoss EAP. To retain the deployment content, add the parameter **`--keep-content`**.

- **Run the `undeploy` command**

To undeploy the application and delete the deployment content, enter the Management CLI **undeploy** command with the filename of the application deployment.

```
[standalone@localhost:9999 /] undeploy test-application.war
```

To undeploy the application, but retain the deployment content, enter the Management CLI **undeploy** command with the filename of the application deployment and the parameter **`--keep-content`**.

```
[standalone@localhost:9999 /] undeploy test-application.war --keep-content
```

Result

The specified application is now undeployed. Note that the **undeploy** command does not produce any output to the Management CLI if it is successful.

[Report a bug](#)

10.3.4. Deploy an Application in a Managed Domain Using the Management CLI

Prerequisites

- [Section 3.4.2, "Launch the Management CLI"](#)
- [Section 3.4.4, "Connect to a Managed Server Instance Using the Management CLI"](#)

Procedure 10.6. Deploy an Application in a Managed Domain

- **Run the `deploy` command**

From the Management CLI, enter the **deploy** command with the path to the application deployment. Include the **`--all-server-groups`** parameter to deploy to all server groups.

```
[domain@localhost:9999 /] deploy /path/to/test-application.war --all-server-groups
```

- Alternatively, define specific server groups for the deployment with the **`--server-groups`** parameter.

```
[domain@localhost:9999 /] deploy /path/to/test-application.war --server-groups=server_group_1,server_group_2
```

Note that a successful deploy does not produce any output to the CLI.

Result

The specified application is now deployed to a server group in your managed domain.

[Report a bug](#)

10.3.5. Undeploy an Application in a Managed Domain Using the Management CLI

Prerequisites

- [Section 3.4.2, "Launch the Management CLI"](#)
- [Section 3.4.4, "Connect to a Managed Server Instance Using the Management CLI"](#)
- [Section 10.3.4, "Deploy an Application in a Managed Domain Using the Management CLI"](#)

Procedure 10.7. Undeploy an Application in a Managed Domain

- **Run the `undeploy` command**

From the Management CLI, enter the **undeploy** command with the filename of the application deployment. The application can be undeployed from any server group that it was originally deployed to with the addition of the **`--all-relevant-server-groups`** parameter.

```
[domain@localhost:9999 /] undeploy test-application.war --all-relevant-server-groups
```

Note that a successful undeploy does not produce any output to the CLI.

Result

The specified application is now undeployed.

[Report a bug](#)

10.4. DEPLOY WITH THE HTTP API

10.4.1. Deploy an application using the HTTP API

Summary

Applications can be deployed via the HTTP API using the following instructions.

Prerequisites

- Add the user for the management interfaces. For information on creating the initial administrative user for the remote management interfaces, refer [Section 4.2.1, “Add the User for the Management Interfaces”](#)

Procedure 10.8. Deploy an application using HTTP API

- Use either the **deploy** or **undeploy** command relevant to your requirements.

Example 10.2. Deploy and undeploy command

Deploy

```
curl --digest -L -D - http://<host>:<port>/management --header "Content-Type: application/json" -d '{"operation": "composite", "address": [], "steps": [{"operation": "add", "address": {"deployment": "<runtime-name>"}, "content": [{"url": "file:<path-to-archive>}],{"operation": "deploy", "address": {"deployment": "<runtime-name>"}], "json.pretty": 1}' -u <user>:<pass>
```

Example:

```
curl --digest -L -D - http://localhost:9990/management --header "Content-Type: application/json" -d '{"operation": "composite", "address": [], "steps": [{"operation": "add", "address": {"deployment": "example.war"}, "content": [{"url": "file:/home/$user/example.war"}],{"operation": "deploy", "address": {"deployment": "example.war"}], "json.pretty": 1}' -u user:password
```

Undeploy

```
curl --digest -L -D - http://<host>:<port>/management --header "Content-Type: application/json" -d '{"operation": "composite", "address": [], "steps": [{"operation": "undeploy", "address": {"deployment": "<runtime-name>"}, {"operation": "remove", "address": {"deployment": "<runtime-name>"}], "json.pretty": 1}' -u <user>:<pass>
```

Example:

```
curl --digest -L -D - http://localhost:9990/management --header "Content-Type:
```

```
application/json" -d '{"operation": "composite", "address": [], "steps": [{"operation": "undeploy", "address": {"deployment": "example.war"}}, {"operation": "remove", "address": {"deployment": "example.war"}}, {"json.pretty": 1}]' -u user:password
```



NOTE

To know more about programmatically generating the JSON requests, refer <https://access.redhat.com/solutions/82463>.

[Report a bug](#)

10.5. DEPLOY WITH THE DEPLOYMENT SCANNER

10.5.1. Manage Application Deployment in the Deployment Scanner

Deploying applications to a standalone server instance via the deployment scanner allows you to build and test applications in a manner suited for rapid development cycles. You can configure the deployment scanner to suit your needs for deployment frequency and behavior for a variety of application types.

[Report a bug](#)

10.5.2. Deploy an Application to a Standalone Server Instance with the Deployment Scanner

Prerequisites

- [Section 2.2.1, “Start JBoss EAP 6”](#)

Summary

This task shows a method for deploying applications to a standalone server instance with the deployment scanner. As indicated in the [Section 10.1, “About Application Deployment”](#) topic, this method is retained for the convenience of developers, where the Management Console and Management CLI methods are recommended for application management under production environments.

Procedure 10.9. Use the Deployment Scanner to Deploy Applications

1. **Copy content to the deployment folder**

Copy the application file to the deployment folder found at ***EAP_HOME/standalone/deployments/***.

2. **Deployment scanning modes**

There are two application deployment methods. You can choose between automatic and manual deployment scanner modes. Before starting either of the deployment methods, read [Section 10.5.8, “Configure the Deployment Scanner with the Management CLI”](#).

- **Automatic deployment**

The deployment scanner picks up a change to the state of the folder and creates a marker file as defined in [Section 10.5.8, “Configure the Deployment Scanner with the Management CLI”](#).

- **Manual deployment**

The deployment scanner requires a marker file to trigger the deployment process. The following example uses the Unix **touch** command to create a new **.dodeploy** file.

Example 10.3. Deploy with the touch command

```
[user@host bin]$ touch  
$EAP_HOME/standalone/deployments/example.war.dodeploy
```

Result

The application file is deployed to the application server. A marker file is created in the deployment folder to indicate the successful deployment, and the application is flagged as **Enabled** in the Management Console.

Example 10.4. Deployment folder contents after deployment

```
example.war  
example.war.deployed
```

[Report a bug](#)

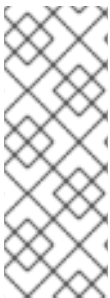
10.5.3. Undeploy an Application from a Standalone Server Instance with the Deployment Scanner

Prerequisites

- [Section 2.2.1, "Start JBoss EAP 6"](#)
- [Section 10.5.2, "Deploy an Application to a Standalone Server Instance with the Deployment Scanner"](#)

Summary

This task shows a method for undeploying applications from a standalone server instance that have been deployed with the deployment scanner. As indicated in the [Section 10.1, "About Application Deployment"](#) topic, this method is retained for the convenience of developers, where the Management Console and Management CLI methods are recommended for application management under production environments.

**NOTE**

The deployment scanner should not be used in conjunction with other deployment methods for application management. Applications removed from the application server by the management console will be removed from the runtime without affecting the marker files or application contained in the deployment directory. To minimize the risk of accidental redployment or other errors, use the Management CLI and Management Console for administration in production environments.

Procedure 10.10. Undeploy an Application using one of these Methods

- **Undeploy the application**

There are two methods to undeploy the application depending on whether you want to delete the application from the deployment folder or only alter its deployment status.

- **Undeploy by deleting the marker file**

Delete the deployed application's **example.war.deployed** marker file to trigger the deployment scanner to begin undeploying the application from the runtime.

Result

The deployment scanner undeploys the application and creates a **example.war.undeployed** marker file. The application remains in the deployment folder.

- **Undeploy by removing the application**



NOTE

Undeploying an exploded WAR file using this method is not valid. Only undeployment by removing the marker file is valid. Attempting to undeploy an exploded WAR file will result in a message like the following message being logged.

```
WARN [org.jboss.as.server.deployment.scanner] (DeploymentScanner-threads - 2) JBAS015006: The deployment scanner found that the content for exploded deployment EXAMPLE.war has been deleted, but auto-deploy/undeploy for exploded deployments is not enabled and the EXAMPLE.war.deployed marker file for this deployment has not been removed. As a result, the deployment is not being undeployed, but resources needed by the deployment may have been deleted and application errors may occur. Deleting the EXAMPLE.war.deployed marker file to trigger undeploy is recommended.
```

Remove the application from the deployment directory to trigger the deployment scanner to begin undeploying the application from the runtime.

Result

The deployment scanner undeploys the application and creates a **filename.filetype.undeployed** marker file. The application is not present in the deployment folder.

Result

The application file is undeployed from the application server and is not visible in the **Deployments** screen of the Management Console.

[Report a bug](#)

10.5.4. Redeploy an Application to a Standalone Server Instance with the Deployment Scanner

Prerequisites

- [Section 2.2.1, "Start JBoss EAP 6"](#)

- [Section 10.5.2, “Deploy an Application to a Standalone Server Instance with the Deployment Scanner”](#)

Summary

This task shows a method for redeploying applications to a standalone server instance that have been deployed with the deployment scanner. As indicated in the [Section 10.1, “About Application Deployment”](#) topic, this method is retained for the convenience of developers, where the Management Console and Management CLI methods are recommended for application management under production environments.

Procedure 10.11. Redeploy an Application to a Standalone Server

- **Redeploy the application**

There are three possible methods to redeploy an application deployed with the deployment scanner. These methods trigger the deployment scanner to initiate a deployment cycle, and can be chosen to suit personal preference.

- **Redeploy by altering the marker file**

Trigger the deployment scanner redeployment by altering the marker file's access and modification timestamp. In the following Linux example, a Unix **touch** command is used.

Example 10.5. Redeploy with the Unix touch command

```
[user@host bin]$ touch EAP_HOME/standalone/deployments/example.war.dodeploy
```

Result

The deployment scanner detects a change in the marker file and redeploys the application. A new **.deployed** file marker replaces the previous.

- **Redeploy by creating a new .dodeploy marker file**

Trigger the deployment scanner redeployment by creating a new **.dodeploy** marker file. Refer to the manual deployment instructions in [Section 10.5.2, “Deploy an Application to a Standalone Server Instance with the Deployment Scanner”](#).

- **Redeploy by deleting the marker file**

As described in [Section 10.5.5, “Reference for Deployment Scanner Marker Files”](#), deleting a deployed application's **.deployed** marker file will trigger an undeployment and create an **.undeployed** marker. Deleting the undeployment marker will trigger the deployment cycle again. Refer to [Section 10.5.3, “Undeploy an Application from a Standalone Server Instance with the Deployment Scanner”](#) for further information.

Result

The application file is redeployed.

[Report a bug](#)

10.5.5. Reference for Deployment Scanner Marker Files

Marker files

Marker files are a part of the deployment scanner subsystem. These files mark the status of an application within the deployment directory of the standalone server instance. A marker file has the same name as the application, with the file suffix indicating the state of the application's deployment.

The following table defines the types and responses for each marker file.

Example 10.6. Marker file example

The following example shows the marker file for a successfully deployed instance of an application called **testapplication.war**.

```
testapplication.war.deployed
```

Table 10.1. Marker filetype definitions

Filename Suffix	Origin	Description
.dodeploy	User generated	Indicates that the content should be deployed or redeployed into the runtime.
.skipdeploy	User generated	Disables auto-deploy of an application while present. Useful as a method of temporarily blocking the auto-deployment of exploded content, preventing the risk of incomplete content edits pushing live. Can be used with zipped content, although the scanner detects in-progress changes to zipped content and waits until completion.
.isdeploying	System generated	Indicates the initiation of deployment. The marker file will be deleted when the deployment process completes.
.deployed	System generated	Indicates that the content has been deployed. The content will be undeployed if this file is deleted.
.failed	System generated	Indicates deployment failure. The marker file contains information about the cause of failure. If the marker file is deleted, the content will be visible to the auto-deployment again.
.isundeploying	System generated	Indicates a response to a .deployed file deletion. The content will be undeployed and the marker will be automatically deleted upon completion.
.undeployed	System generated	Indicates that the content has been undeployed. Deletion of the marker file has no impact to content redeployment.
.pending	System generated	Indicates that deployment instructions will be sent to the server pending resolution of a detected issue. This marker serves as a global deployment road-block. The scanner will not instruct the server to deploy or undeploy any other content while this condition exists.

[Report a bug](#)

10.5.6. Reference for Deployment Scanner Attributes

The deployment scanner contains the following attributes that are exposed to the Management CLI and able to be configured using the **write-attribute** operation. For more information on configuration options, refer to the topic [Section 10.5.8, “Configure the Deployment Scanner with the Management CLI”](#).

Table 10.2. Deployment Scanner Attributes

Name	Description	Type	Default Value
auto-deploy-exploded	Allows the automatic deployment of exploded content without requiring a .dodeploy marker file. Recommended for only basic development scenarios to prevent exploded application deployment from occurring during changes by the developer or operating system.	Boolean	False
auto-deploy-xml	Allows the automatic deployment of XML content without requiring a .dodeploy marker file.	Boolean	True
auto-deploy-zipped	Allows the automatic deployment of zipped content without requiring a .dodeploy marker file.	Boolean	True
deployment-timeout	The time value in seconds for the deployment scanner to allow a deployment attempt before being cancelled.	Long	600
path	Defines the actual filesystem path to be scanned. If the relative-to attribute is specified, the path value acts as a relative addition to that directory or path.	String	deployment s
relative-to	Reference to a filesystem path defined in the paths section of the server configuration XML file.	String	jboss.server .base.dir
scan-enabled	Allows the automatic scanning for applications by scan-interval and at startup.	Boolean	True
scan-interval	The time interval in milliseconds between scans of the repository. A value of less than 1 restricts the scanner to operate only at startup.	Int	5000

[Report a bug](#)

10.5.7. Configure the Deployment Scanner

The deployment scanner can be configured using the Management Console or the Management CLI. You can create a new deployment scanner or manage the existing scanner attributes. These include the scanning interval, the location of the deployment folder, and the application file types that will trigger a deployment.

[Report a bug](#)

10.5.8. Configure the Deployment Scanner with the Management CLI

Prerequisites

- [Section 3.4.2, “Launch the Management CLI”](#)

Summary

While there are multiple methods of configuring the deployment scanner, the Management CLI can be used to expose and modify the attributes by use of batch scripts or in real time. You can modify the behavior of the deployment scanner by use of the **read-attribute** and **write-attribute** global command line operations. Further information about the deployment scanner attributes are defined in the topic [Section 10.5.6, “Reference for Deployment Scanner Attributes”](#).

The deployment scanner is a subsystem of JBoss EAP 6, and can be viewed in the **standalone.xml**.

Example 10.7. Excerpt from **standalone.xml**

```
<subsystem xmlns="urn:jboss:domain:deployment-scanner:1.1">
  <deployment-scanner path="deployments" relative-to="jboss.server.base.dir" scan-
interval="5000"/>
</subsystem>
```

Procedure 10.12. Configure the Deployment Scanner

1. Determine the deployment scanner attributes to configure

Configuring the deployment scanner via the Management CLI requires that you first expose the correct attribute names. You can do this with the **read-resources** operation at either the root node, or by using the **cd** command to change into the subsystem child node. You can also display the attributes with the **ls** command at this level.

- **Expose the deployment scanner attributes with the **read-resource** operation**
Use the **read-resource** operation to expose the attributes defined by the default deployment scanner resource.

Example 10.8. Sample **read-resource** output

```
[standalone@localhost:9999 /]/subsystem=deployment-
scanner/scanner=default:read-resource
{
  "outcome" => "success",
  "result" => {
    "auto-deploy-exploded" => false,
    "auto-deploy-xml" => true,
    "auto-deploy-zipped" => true,
    "deployment-timeout" => 600,
```

```

    "path" => "deployments",
    "relative-to" => "jboss.server.base.dir",
    "scan-enabled" => true,
    "scan-interval" => 5000
  }
}

```

- o **Expose the deployment scanner attributes with the `ls` command**

Use the `ls` command with the `-l` optional argument to display a table of results that include the subsystem node attributes, values, and type. You can learn more about the `ls` command and its arguments by exposing the CLI help entry by typing `ls --help`. For more information about the help menu in the Management CLI, refer to the topic [Section 3.4.5, "Obtain Help with the Management CLI"](#).

Example 10.9. Sample `ls -l` output

```

[standalone@localhost:9999 /] ls -l /subsystem=deployment-scanner/scanner=default
ATTRIBUTE      VALUE      TYPE
auto-deploy-exploded false      BOOLEAN
auto-deploy-xml true       BOOLEAN
auto-deploy-zipped true       BOOLEAN
deployment-timeout 600      LONG
path            deployments STRING
relative-to    jboss.server.base.dir STRING
scan-enabled   true      BOOLEAN
scan-interval  5000     INT

```

2. Configure the deployment scanner with the `write-attribute` operation

Once you have determined the name of the attribute to modify, use the **write-attribute** to specify the attribute name and the new value to write to it. The following examples are all run at the child node level, which can be accessed by using the `cd` command and tab completion to expose and change into the default scanner node.

```

[standalone@localhost:9999 /] cd subsystem=deployment-scanner/scanner=default

```

a. Enable automatic deployment of exploded content

Use the **write-attribute** operation to enable the automatic deployment of exploded application content.

```

[standalone@localhost:9999 scanner=default] :write-attribute(name=auto-deploy-
exploded,value=true)
{"outcome" => "success"}

```

b. Disable the automatic deployment of XML content

Use the **write-attribute** operation to disable the automatic deployment of XML application content.

```

[standalone@localhost:9999 scanner=default] :write-attribute(name=auto-deploy-
xml,value=false)
{"outcome" => "success"}

```

c. **Disable the automatic deployment of zipped content**

Use the **write-attribute** command to disable the automatic deployment of zipped application content.

```
[standalone@localhost:9999 scanner=default] :write-attribute(name=auto-deploy-zipped,value=false)
{"outcome" => "success"}
```

d. **Configure the path attribute**

Use the **write-attribute** operation to modify the path attribute, substituting the example **newpathname** value for the new path name for the deployment scanner to monitor. Note that the server will require a reload to take effect.

```
[standalone@localhost:9999 scanner=default] :write-attribute(name=path,value=newpathname)
{
  "outcome" => "success",
  "response-headers" => {
    "operation-requires-reload" => true,
    "process-state" => "reload-required"
  }
}
```

e. **Configure the relative path attribute**

Use the **write-attribute** operation to modify the relative reference to the filesystem path defined in the paths section of the configuration XML file. Note that the server will require a reload to take effect.

```
[standalone@localhost:9999 scanner=default] :write-attribute(name=relative-to,value=new.relative.dir)
{
  "outcome" => "success",
  "response-headers" => {
    "operation-requires-reload" => true,
    "process-state" => "reload-required"
  }
}
```

f. **Disable the deployment scanner**

Use the **write-attribute** operation to disable the deployment scanner by setting the **scan-enabled** value to false.

```
[standalone@localhost:9999 scanner=default] :write-attribute(name=scan-enabled,value=false)
{"outcome" => "success"}
```

g. **Change the scan interval**

Use the **write-attribute** operation to modify the scan interval time from 5000 milliseconds to 10000 milliseconds.

```
[standalone@localhost:9999 scanner=default] :write-attribute(name=scan-interval,value=10000)
{"outcome" => "success"}
```

Result

Your configuration changes are saved to the deployment scanner.

[Report a bug](#)

10.5.9. Define a Custom Deployment Scanner**Prerequisites**

- [Section 3.4.2, "Launch the Management CLI"](#)

Summary

A new deployment scanner can be added using the Management Console or the Management CLI. This will define a new directory to scan for deployments. The default deployment scanner monitors ***EAP_HOME/standalone/deployments/***. See [Section 10.5.8, "Configure the Deployment Scanner with the Management CLI"](#) for details on configuring an existing deployment scanner.

Procedure 10.13. Define a Custom Deployment Scanner with the Management CLI

1. Add a deployment scanner using the Management CLI **add** operation:

```
[standalone@localhost:9999 /] /subsystem=deployment-scanner/scanner=new-scanner:add(path=new_deployment_dir,relative-to=jboss.server.base.dir,scan-interval=5000) {"outcome" => "success"}
```

**NOTE**

The specified directory must already exist or this command will fail with an error.

2. The new deployment scanner is now visible in the **standalone.xml** file and management interfaces.

Example 10.10. Excerpt from standalone.xml

```
<subsystem xmlns="urn:jboss:domain:deployment-scanner:1.1">
  <deployment-scanner path="deployments" relative-to="jboss.server.base.dir" scan-interval="5000"/>
  <deployment-scanner name="new-scanner" path="new_deployment_dir" relative-to="jboss.server.base.dir" scan-interval="5000"/>
</subsystem>
```

3. The specified directory will now be scanned for deployments. See [Section 10.5.2, "Deploy an Application to a Standalone Server Instance with the Deployment Scanner"](#) for details on deploying an application with the deployment scanner.

Result

A new deployment scanner has been defined and is monitoring for deployments.

[Report a bug](#)

10.6. DEPLOY WITH MAVEN

10.6.1. Manage Application Deployment with Maven

Deploying applications via Maven allows you to incorporate a deployment cycle as part of your existing development workflow.

[Report a bug](#)

10.6.2. Deploy an Application with Maven

Prerequisites

- [Section 2.2.1, "Start JBoss EAP 6"](#)

Summary

This task shows a method for deploying applications with Maven. The example provided uses the **jboss-helloworld.war** application found in the JBoss EAP 6 Quickstarts collection. The **helloworld** project contains a POM file which initializes the **jboss-as-maven-plugin**. This plug-in provides simple operations to deploy and undeploy applications to and from the application server.

Procedure 10.14. Deploy an application with Maven

1. Open a terminal session and navigate to the directory containing the quickstart examples.

Example 10.11. Change into the helloworld application directory

```
[localhost]$ cd /QUICKSTART_HOME/helloworld
```

2. Run the Maven deploy command to deploy the application. If the application is already running, it will be redeployed.

```
[localhost]$ mvn package jboss-as:deploy
```

3. View the results.
 - The deployment can be confirmed by viewing the operation logs in the terminal window.

Example 10.12. Maven confirmation for helloworld application

```
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 32.629s
[INFO] Finished at: Fri Mar 14 09:09:50 EDT 2014
[INFO] Final Memory: 23M/204M
[INFO] -----
```

- The deployment can also be confirmed in the status stream of the active application server instance.

Example 10.13. Application server confirmation for helloworld application

```

09:09:49,167 INFO [org.jboss.as.repository] (management-handler-thread - 1)
JBAS014900: Content added at location
/home/username/EAP_HOME/standalone/data/content/32/4b4ef9a4bbe7206d3674a89
807203a2092fc70/content
09:09:49,175 INFO [org.jboss.as.server.deployment] (MSC service thread 1-7)
JBAS015876: Starting deployment of "jboss-helloworld.war" (runtime-name: "jboss-
helloworld.war")
09:09:49,563 INFO [org.jboss.weld.deployer] (MSC service thread 1-8) JBAS016002:
Processing weld deployment jboss-helloworld.war
09:09:49,611 INFO [org.jboss.weld.deployer] (MSC service thread 1-1) JBAS016005:
Starting Services for CDI deployment: jboss-helloworld.war
09:09:49,680 INFO [org.jboss.weld.Version] (MSC service thread 1-1) WELD-000900
1.1.17 (redhat)
09:09:49,705 INFO [org.jboss.weld.deployer] (MSC service thread 1-2) JBAS016008:
Starting weld service for deployment jboss-helloworld.war
09:09:50,080 INFO [org.jboss.web] (ServerService Thread Pool -- 55) JBAS018210:
Register web context: /jboss-helloworld
09:09:50,425 INFO [org.jboss.as.server] (management-handler-thread - 1)
JBAS018559: Deployed "jboss-helloworld.war" (runtime-name : "jboss-
helloworld.war")

```

Result

The application is deployed to the application server.

[Report a bug](#)

10.6.3. Undeploy an Application with Maven**Prerequisites**

- [Section 2.2.1, "Start JBoss EAP 6"](#)

Summary

This task shows a method for undeploying applications with Maven. The example provided uses the **jboss-helloworld.war** application found in the JBoss EAP 6 Quickstarts collection. The **helloworld** project contains a POM file which initializes the **jboss-as-maven-plugin**. This plug-in provides simple operations to deploy and undeploy applications to and from the application server.

Procedure 10.15. Undeploy an Application with Maven

1. Open a terminal session and navigate to the directory containing the quickstart examples.

Example 10.14. Change into the helloworld application directory

```
[localhost]$ cd /QUICKSTART_HOME/helloworld
```

2. Run the Maven undeploy command to undeploy the application.


```
[localhost]$ mvn jboss-as:undeploy
```

3. View the results.

- The undeployment can be confirmed by viewing the operation logs in the terminal window.

Example 10.15. Maven confirmation for undeploy of helloworld application

```
[INFO] -----
[INFO] BUILD SUCCESSFUL
[INFO] -----
[INFO] Total time: 1 second
[INFO] Finished at: Mon Oct 10 17:33:02 EST 2011
[INFO] Final Memory: 11M/212M
[INFO] -----
```

- The undeployment can also be confirmed in the status stream of the active application server instance.

Example 10.16. Application server confirmation for undeploy of helloworld application

```
09:51:40,512 INFO [org.jboss.web] (ServerService Thread Pool -- 69) JBAS018224:
Unregister web context: /jboss-helloworld
09:51:40,522 INFO [org.jboss.weld.deployer] (MSC service thread 1-3) JBAS016009:
Stopping weld service for deployment jboss-helloworld.war
09:51:40,536 INFO [org.jboss.as.server.deployment] (MSC service thread 1-1)
JBAS015877: Stopped deployment jboss-helloworld.war (runtime-name: jboss-
helloworld.war) in 27ms
09:51:40,621 INFO [org.jboss.as.repository] (management-handler-thread - 10)
JBAS014901: Content removed from location /home/username/EAP_HOME/jboss-
eap-
6.4/standalone/data/content/44/e1f3c55c84b777b0fc201d69451223c09c9da5/content
09:51:40,621 INFO [org.jboss.as.server] (management-handler-thread - 10)
JBAS018558: Undeployed "jboss-helloworld.war" (runtime-name: "jboss-
helloworld.war")
```

Result

The application is undeployed from the application server.

[Report a bug](#)

10.7. CONTROL THE ORDER OF DEPLOYED APPLICATIONS ON JBOSS EAP 6

JBoss EAP 6 offers fine grained control over the order of deployment of applications when the server is started. Strict order of deployment of applications present in multiple ear files can be enabled along with persistence of the order after a restart.

Procedure 10.16. Control the order of deployment in EAP 6.0

1. Create CLI scripts that will deploy and undeploy the applications in sequential order when the server is started/stopped.
2. CLI also supports the concept of batch mode which allows you to group commands and operations and execute them together as an atomic unit. If at least one of the commands or operations fails, all the other successfully executed commands and operations in the batch are rolled back.

Procedure 10.17. Control the order of deployment in EAP 6.1 and later

From EAP 6.1 onward, Inter Deployment Dependencies allows you to declare dependencies between top level deployments.

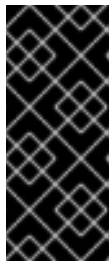
1. Create (if it doesn't exist) a **jboss-all.xml** file in the **app.ear/META-INF** folder, where **app.ear** is the application archive that depends on another application archive to be deployed before it is.
2. Make a **jboss-deployment-dependencies** entry in this file as shown below. Note that in the listing below, **framework.ear** is the dependency application archive that should be deployed before **app.ear** application archive is.

```
<jboss xmlns="urn:jboss:1.0">
  <jboss-deployment-dependencies xmlns="urn:jboss:deployment-dependencies:1.0">
    <dependency name="framework.ear" />
  </jboss-deployment-dependencies>
</jboss>
```



NOTE

You can use the deployment's runtime name as the dependency name in the **jboss-all.xml** file.



IMPORTANT

Although the **jboss-all.xml** file allows you to declare dependencies that the server does not otherwise detect, it is not a strict ordering feature. JBoss EAP assumes that all dependencies specified in the **jboss-all.xml** file have already been deployed or are available. If there are missing dependencies, JBoss EAP does not automatically deploy them, and the deployment fails.

[Report a bug](#)

10.8. DEFINE A CUSTOM DIRECTORY FOR DEPLOYED CONTENT

JBoss EAP provides the option to define the location that the server will use for storing deployed content.

Define a Custom Directory for Deployed Content in Standalone Mode

By default, deployed content in Standalone Mode is stored in the **EAP_HOME/standalone/data/content** directory.

- This location can be changed by passing in the **-Djboss.server.deploy.dir** argument when starting the server:

```
./standalone.sh -Djboss.server.deploy.dir=/path/to/new_deployed_content
```

- The chosen location should be unique among JBoss EAP instances.



NOTE

jboss.server.deploy.dir specifies the directory used for storing content that has been deployed via the Management Console or Management CLI. For defining a custom deployments directory to be monitored by the deployment scanner, see [Section 10.5.9, “Define a Custom Deployment Scanner”](#).

Define a Custom Directory for Deployed Content in Domain Mode

By default, deployed content in Domain Mode is stored in the **EAP_HOME/domain/data/content** directory.

- This location can be changed by passing in the **-Djboss.domain.deployment.dir** argument when starting the domain:

```
./domain.sh -Djboss.domain.deployment.dir=/path/to/new_deployed_content
```

- The chosen location should be unique among JBoss EAP instances.

[Report a bug](#)

10.9. DEPLOYMENT DESCRIPTOR OVERRIDES

From JBoss EAP 6.1 onward you can override deployment descriptors, JARs, classes, JSP pages, and other files at runtime. A *deployment overlay* represents a ruleset of files that must be overridden in the archive. It also provides links to the new files that must be used instead of the overridden ones. If the file being overridden is not present in the deployment archive, it will be added back to the deployment.

Procedure 10.18. Override the deployment descriptor using the Management CLI

The following steps assume that you already have a deployed application called **app.war** and you wish to override its **WEB-INF/web.xml** file with another **web.xml** file located in **/home/user/web.xml**.

1. Add a deployment overlay and add content to it. You can achieve this in the following two ways:

- Using DMR tree

- a.

```
/deployment-overlay=myoverlay:add
```

- b.

```
/deployment-overlay=myoverlay/content=WEB-INF/web.xml:add(content={url=file:///home/user/web.xml})
```

You can also add more content rules using the second statement.

- Using convenience methods

```
deployment-overlay add --name=myoverlay --content=WEB-INF/web.xml=/home/user/web.xml
```

2. Link the overlay to a deployment archive. You can achieve this in the following two ways:

- **Using DMR tree**

```
/deployment-overlay=myoverlay/deployment=app.war:add
```

- **Using convenience methods**

```
deployment-overlay link --name=myoverlay --deployments=app.war
```

To specify multiple archive names, separate them by commas.

Note that the deployment archive name need not exist on the server. You are specifying the name, but not yet linking it to an actual deployment.

3. **Redeploy the application**

```
/deployment=app.war:redeploy
```

[Report a bug](#)

10.10. ROLLOUT PLAN

10.10.1. Rollout Plans

Operations targeted at domain or host level resources can potentially impact multiple servers. Such operations can include a *roll out plan* detailing the sequence in which the operation would be applied to the servers, as well as the policies for detailing whether the operation could be reverted if it fails to execute successfully on some servers.

Example 10.17. CLI format of a rollout plan

```
rollout (id=plan_id | server_group_list) [rollback-across-groups]
server_group_list := server_group [ (sequence_separator | concurrent_separator) server_group ]
sequence_separator := ','
concurrent_separator := '^'
server_group := server_group_name [group_policy_list]
group_policy_list := '(' policy_property_name=policy_property_value (,
policy_property_name=policy_property_value)* ')'
policy_property_name := 'rolling-to-servers' | 'max-failed-servers' | 'max-failure-percentage'
```

The value of `policy_property_value` depends on the property. It can be a boolean, an integer, etc.

Rollout plans can potentially be long and complex. There is a possibility, though, to store them as a part of the domain management model and then later be referenced from commands and operations using their name (or ID in the definition above). Stored rollout plans are managed using the **rollout-plan** command.

Example 10.18. Rollout plan managed with the `rollout-plan` command

```
rollout-plan add --name=my-plan --content={rollout main-server-group^other-server-group}
:write-attribute(name=my-attr,value=my-value){rollout id=my-plan}
```

Example 10.19. Using a stored rollout plan

```
rollout-plan add --name=my-plan --content={rollout main-server-group^other-server-group}
:write-attribute(name=my-attr,value=my-value){rollout id=my-plan}
```

[Report a bug](#)

10.10.2. Operations with a Rollout Plan

The structure is of **rollout-plan** within an **operation** is as follows:

```
{
  "operation" => "write-core-threads",
  "address" => [
    ("profile" => "production"),
    ("subsystem" => "threads"),
    ("bounded-queue-thread-pool" => "pool1")
  ],
  "count" => 0,
  "per-cpu" => 20,
  "operation-headers" => {
    "rollout-plan" => {
      "in-series" => [
        {
          "concurrent-groups" => {
            "groupA" => {
              "rolling-to-servers" => true,
              "max-failure-percentage" => 20
            },
            "groupB" => undefined
          }
        },
        {
          "server-group" => {
            "groupC" => {
              "rolling-to-servers" => false,
              "max-failed-servers" => 1
            }
          }
        }
      ],
      {
        "concurrent-groups" => {
          "groupD" => {
            "rolling-to-servers" => true,
            "max-failure-percentage" => 20
          },
          "groupE" => undefined
        }
      }
    }
  }
}
```

```

    ],
    "rollback-across-groups" => true
  }
}
}

```

The **rollout-plan** is nested within the **operation-headers** structure. The root node of the structure allows two children:

- **in-series** - A list of steps that are to be performed in series, with each step reaching completion before the next step is executed. Each step involves the application of the operation to the servers in one or more server groups. See below for details on each element in the list.
- **rollback-across-groups** - A boolean that indicates whether the need to rollback the operation on all the servers in one server group triggers a rollback across all the server groups. This is an optional setting, and defaults to false.

Each element in the list under the **in-series** node must have one or the other of the following structures:

- **concurrent-groups** - A map of server group names to policies controlling how the operation should be applied to that server group. For each server group in the map, the operation may be applied concurrently. See below for details on the per-server-group policy configuration.
- **server-group** - A single key/value mapping of a server group name to a policy controlling how the operation should be applied to that server group. See below for details on the policy configuration. (Note: there is no difference in plan execution between this and a "concurrent-groups" map with a single entry.)

The policy controlling how the operation is applied to the servers within a server group has the following elements, each of which is optional:

- **rolling-to-servers** - A boolean which if set to **true**, the operation will be applied to each server in the group in series. If false or not specified, the operation will be applied to the servers in the group concurrently.
- **max-failed-servers** - An integer which takes the maximum number of servers in the group that can fail to apply the operation before it should be reverted on all servers in the group. The default value if not specified is zero; i.e. failure on any server triggers rollback across the group.
- **max-failure-percentage** - An integer between 0 and 100 which takes the maximum percentage of the total number of servers in the group that can fail to apply the operation before it should be reverted on all servers in the group. The default value if not specified is zero; i.e. failure on any server triggers rollback across the group.

If both **max-failed-servers** and **max-failure-percentage** are set to non-zero values, **max-failure-percentage** takes precedence.

Looking at the (contrived) example above, application of the operation to the servers in the domain would be done in 3 phases. If the policy for any server group triggers a rollback of the operation across the server group, all other server groups will be rolled back as well. The 3 phases are:

1. Server groups groupA and groupB will have the operation applied concurrently. The operation will be applied to the servers in groupA in series, while all servers in groupB will handle the operation concurrently. If more than 20% of the servers in groupA fail to apply the operation, it will be rolled back across that group. If any servers in groupB fail to apply the operation it will be rolled back across that group.

2. Once all servers in groupA and groupB are complete, the operation will be applied to the servers in groupC. Those servers will handle the operation concurrently. If more than one server in groupC fails to apply the operation it will be rolled back across that group.
3. Once all servers in groupC are complete, server groups groupD and groupE will have the operation applied concurrently. The operation will be applied to the servers in groupD in series, while all servers in groupE will handle the operation concurrently. If more than 20% of the servers in groupD fail to apply the operation, it will be rolled back across that group. If any servers in groupE fail to apply the operation it will be rolled back across that group.

Default Rollout Plan

All operations that impact multiple servers will be executed with a rollout plan. However, actually specifying the rollout plan in the operation request is not required. If no rollout-plan is specified, a default plan will be generated. The plan will have the following characteristics:

- There will only be a single high level phase. All server groups affected by the operation will have the operation applied concurrently.
- Within each server group, the operation will be applied to all servers concurrently.
- Failure on any server in a server group will cause rollback across the group.
- Failure of any server group will result in rollback of all other server groups.

[Report a bug](#)

10.10.3. Creating a Rollout Deployment Plan

How to create a roll out deployment plan to deploy applications in a clustered domain in JBoss EAP 6

1. Create a rollout deployment plan using CLI with `rolling-to-servers=true`. The package will be deployed to each server in the server group in a serial manner.

An example CLI deployment plan for serial deployment is provided below:

```
deploy ClusterWebApp.war --name=ClusterWebApp.war --runtime-
name=ClusterWebApp.war --server-groups=ha-server-group --headers={rollout ha-server-
group(rolling-to-servers=true)}
```

2. To apply a rollout plan to all the server-groups, you need to mention the names of each server-group in master host:

```
deploy /NotBackedUp/PREVIOUS/ALLWAR/ClusterWebApp.war --
name=ClusterWebApp.war --runtime-name=ClusterWebApp.war --server-groups=main-
server-group,other-server-group --headers={rollout main-server-group(rolling-to-
servers=true),other-server-group(rolling-to-servers=true)}
```

[Report a bug](#)

CHAPTER 11. SUBSYSTEM CONFIGURATION

11.1. SUBSYSTEM CONFIGURATION OVERVIEW

Introduction

JBoss EAP 6 uses a simplified configuration, with one configuration file per domain or per standalone server. In domain mode, a separate file exists for each host controller as well. Changes to the configuration persist automatically, so XML configuration file should not be edited manually. The configuration is scanned and overwritten automatically by the Management API. The command-line based Management CLI and web-based Management Console allow you to configure each aspect of JBoss EAP 6.

JBoss EAP 6 is built on the concept of modular class loading. Each API or service provided by the Platform is implemented as a module, which is loaded and unloaded on demand. Most modules include a configurable element called a subsystem. Subsystem configuration information is stored in the unified configuration file ***EAP_HOME/domain/configuration/domain.xml*** for a managed domain or ***EAP_HOME/standalone/configuration/standalone.xml*** for a standalone server. Many of the subsystems include configuration details that were configured via deployment descriptors in previous versions of JBoss EAP.

Subsystem Configuration Schemas

Each subsystem's configuration is defined in an XML schema. The configuration schemas are located in the ***EAP_HOME/docs/schema/*** directory of your installation.

[Report a bug](#)

CHAPTER 12. THE LOGGING SUBSYSTEM

12.1. INTRODUCTION

12.1.1. Overview of Logging

JBoss EAP 6 provides highly configurable logging facilities for both its own internal use and for use by deployed applications. The logging subsystem is based on JBoss LogManager and it supports several third party application logging frameworks in addition to JBoss Logging.

The logging subsystem is configured using a system of log categories and log handlers. Log categories define what messages to capture, and log handlers define how to deal with those messages (write to disk, send to console etc).

Logging Profiles allow uniquely named sets of logging configuration to be created and assigned to applications independent of any other logging configuration. The configuration of logging profiles is almost identical to the main logging subsystem.

[Report a bug](#)

12.1.2. Application Logging Frameworks Supported By JBoss LogManager

JBoss LogManager supports the following logging frameworks:

- JBoss Logging - included with JBoss EAP 6
- Apache Commons Logging - <http://commons.apache.org/logging/>
- Simple Logging Facade for Java (SLF4J) - <http://www.slf4j.org/>
- Apache log4j - <http://logging.apache.org/log4j/1.2/>
- Java SE Logging (java.util.logging) - <http://download.oracle.com/javase/6/docs/api/java/util/logging/package-summary.html>

JBoss LogManager supports the following APIs:

- java.util.logging
- JBoss Logging
- Log4j
- SLF4J
- commons-logging

JBoss LogManager also supports the following SPIs:

- java.util.logging Handler
- Log4j Appender

**NOTE**

If you are using the **Log4j API** and a **Log4J Appender**, then Objects will be converted to **string** before being passed.

[Report a bug](#)

12.1.3. Bootup Logging

During bootup JBoss EAP outputs log entries about the Java environment and the startup of each service. The log can be useful when troubleshooting. By default all log entries are written to the file **server.log**, the location of which depends on the runtime mode.

Standalone mode

EAP_HOME/standalone/log/server.log

Domain mode

EAP_HOME/domain/servers/SERVER_NAME/log/server.log

The configuration of bootup logging is specified in the configuration file **logging.properties**, the location of which depends on the runtime mode.

Standalone mode

EAP_HOME/standalone/configuration/logging.properties

Domain mode

In domain mode there is a **logging.properties** file for the domain controller and each server.

Domain controller: ***EAP_HOME/domain/configuration/logging.properties***

Server: ***EAP_HOME/domain/servers/SERVER_NAME/data/logging.properties***

The configuration file **logging.properties** is active until the logging subsystem is started and takes over.

**WARNING**

It is recommended that you do not directly edit the **logging.properties** file unless you know of a specific use case that requires you to do so. Before doing so, it is recommended that you raise a Support Case.

Changes made manually to the **logging.properties** file are overwritten on startup.

[Report a bug](#)

12.1.4. View Bootup Errors

When troubleshooting JBoss EAP, checking for errors which occurred during bootup should be one of the first steps taken. There are two methods of viewing bootup errors, each with its advantages. Each method results in a list of any errors which occurred during bootup. Use the information provided to diagnose and resolve their causes. Contact Red Hat Customer Support for assistance in troubleshooting.

- Examine the **server.log** log file.

This method allows you to see each error message together with possibly related messages, allowing you to get more information about *why* an error might have occurred. It also allows you to see error messages in plain text format.

- From JBoss EAP 6.4, use the Management CLI command **read-boot-errors**.

This method does not require access to the server's file system, which is useful for anyone responsible for monitoring for errors who does not have file system access. Since it is a Management CLI command, it can be used in a script. For example, you could write a script which starts multiple JBoss EAP instances, then checks for errors which occurred on bootup.

Procedure 12.1. Examine server.log for Errors

1. Open the file **server.log** in a file viewer.
2. Navigate to the end of the file.
3. Search backward for the message identifier **JBAS015899**, which marks the start of the latest bootup sequence.
4. Search the log from that point onward for instances of **ERROR**. Each instance will include a description of the error and list the modules involved.

Example 12.1. Error Description from server.log

The following is an example error description from the **server.log** log file.

```
13:23:14,281 ERROR [org.apache.coyote.http11.Http11Protocol] (MSC service thread 1-4)
JBWEB003043: Error initializing endpoint: java.net.BindException: Address already in use
/127.0.0.1:8080
```

Procedure 12.2. List Bootup Errors via the Management CLI

- Run the following Management CLI command.

```
/core-service=management:read-boot-errors
```

Any errors which occurred during bootup will be listed.

The timestamp of each error uses the Java method **currentTimeMillis()**, which is the difference, measured in milliseconds, between the current time and midnight, January 1, 1970 UTC (coordinated universal time).

Example 12.2. Output from read-boot-errors Command

```

{
  "outcome" => "success",
  "result" => [{
    "failed-operation" => {
      "operation" => "add",
      "address" => [
        ("subsystem" => "web"),
        ("connector" => "http")
      ]
    },
    "failure-timestamp" => 1417560953245L,
    "failure-description" => "{\"JBAS014671: Failed services\" => {\"jboss.web.connector.http\" =>
\"org.jboss.msc.service.StartException in service jboss.web.connector.http: JBAS018007: Error
starting web connector
Caused by: LifecycleException: JBWEB000023: Protocol handler initialization failed\"}},
    "failed-services" => {\"jboss.web.connector.http\" => \"org.jboss.msc.service.StartException in
service jboss.web.connector.http: JBAS018007: Error starting web connector
Caused by: LifecycleException: JBWEB000023: Protocol handler initialization failed\"}
  ]
}

```

[Report a bug](#)

12.1.5. About Garbage Collection Logging

Garbage collection logging logs all garbage collection activity to plain text log files. These log files can be useful for diagnostic purposes. From JBoss EAP 6 garbage collection logging is enabled by default for **standalone** mode on all supported configurations *except* IBM Java development kit.

Logging is output to the file ***EAP_HOME/standalone/log/gc.log.digit***. Log rotation has been enabled, with the number of log files limited to five and each file limited to a maximum size of three MiB.

[Report a bug](#)

12.1.6. Implicit Logging API Dependencies

The JBoss EAP 6 logging subsystem has the **add-logging-api-dependencies** attribute that controls whether the container adds implicit logging API dependencies to deployments. By default this attribute is set to **true**, which means that all implicit logging API dependencies are added to deployments. If set to **false**, implicit logging API dependencies will not be added.

The **add-logging-api-dependencies** attribute can be configured using the Management CLI. For example:

```
/subsystem=logging:write-attribute(name=add-logging-api-dependencies, value=false)
```

[Report a bug](#)

12.1.7. Default Log File Locations

These are the log files that get created for the default logging configurations. The default configuration writes the server log files using periodic log handlers

Table 12.1. Default Log File for a standalone server

Log File	Description
<code>EAP_HOME/standalone/log/server.log</code>	Server Log. Contains all server log messages, including server startup messages.
<code>EAP_HOME/standalone/log/gc.log</code>	Garbage collection log. Contains details of all garbage collection.

Table 12.2. Default Log Files for a managed domain

Log File	Description
<code>EAP_HOME/domain/log/host-controller.log</code>	Host Controller boot log. Contains log messages related to the startup of the host controller.
<code>EAP_HOME/domain/log/process-controller.log</code>	Process controller boot log. Contains log messages related to the startup of the process controller.
<code>EAP_HOME/domain/servers/SERVERNAME/log/server.log</code>	The server log for the named server. Contains all log messages for that server, including server startup messages.

[Report a bug](#)

12.1.8. Filter Expressions for Logging

Filter expressions are used to record log messages based on various criterion. Filter checking is always done on a raw unformatted message. You can include a filter for a logger or handler, the logger filter takes precedence over the filter put on a handler.



NOTE

A **filter-spec** specified for the root logger is *not* inherited by other loggers. Instead a **filter-spec** must be specified per handler.

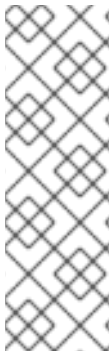
Table 12.3. Filter Expressions for Logging

Filter Type expression	Description	Parameters
Accept accept	Accept all log messages	accept
Deny deny	Deny all log messages	deny

Filter Type	Description	Parameters
expression Not not[filter expression]	Returns the inverted value of the filter expression	Takes single filter expression as a parameter not(match("JBAS"))
All all[filter expression]	Returns concatenated value from multiple filter expressions.	Takes multiple filter expressions delimited by commas all(match("JBAS"),match("WELD"))
Any any[filter expression]	Returns one value from multiple filter expressions.	Takes multiple filter expressions delimited by commas any(match("JBAS"),match("WELD"))
Level Change levelChange[level]	Modifies the log record with the specified level	Takes single string-based level as an argument levelChange("WARN")
Levels levels[levels]	Filters log messages with a level listed in the list of levels	Takes multiple string-based levels delimited by commas as argument levels("DEBUG","INFO","WARN","ERROR")

Filter Type expression	Description	Parameters
Level Range levelRange[minLevel,maxLevel]	Filters log messages within the specified level range.	<p>The filter expression uses [to indicate a minimum inclusive level and a] to indicate a maximum inclusive level. Alternatively, one can use (or) respectively to indicate exclusive. The first argument for the expression is the minimum level allowed, the second argument is the maximum level allowed.</p> <p>Examples are shown below.</p> <ul style="list-style-type: none"> levelRange("DEBUG","ERROR") Minimum level must be greater than DEBUG and the maximum level must be less than ERROR. levelRange["DEBUG","ERROR"] Minimum level must be greater than or equal to DEBUG and the maximum level must be less than ERROR. levelRange["INFO","ERROR"] Minimum level must be greater than or equal to INFO and the maximum level must be less than or equal to ERROR.
Match (match["pattern"])	A regular-expression based filter. The unformatted message is used against the pattern specified in the expression.	Takes a regular expression as argument match("JBAS\d+")
Substitute (substitute["pattern","replacement value"])	A filter which replaces the first match to the pattern with the replacement value	The first argument for the expression is the pattern the second argument is the replacement text substitute("JBAS","EAP")

Filter Type	Description	Parameters
expression		
Substitute All (substituteAll [" pattern ", " replacement value "])	A filter which replaces all matches of the pattern with the replacement value	The first argument for the expression is the pattern the second argument is the replacement text substituteAll ("JBAS", "EAP")



NOTE

Escape the comma and quotation marks in the value (by preceding them with the '\ ' operator) and wrap the entire expression in quotation marks, so that the value is correctly interpreted to be a **string**. Otherwise, it would be parsed as a **list**. An example of the correct format would be:

```
[standalone@localhost:9999 /] /subsystem=logging/console-handler=CONSOLE:write-attribute(name=filter-spec, value="substituteAll(\"JBAS\", \"SABJ\")")
```

[Report a bug](#)

12.1.9. About Log Levels

Log levels are an ordered set of enumerated values that indicate the nature and severity of a log message. The level of a given log message is specified by the developer using the appropriate methods of their chosen logging framework to send the message.

JBoss EAP 6 supports all the log levels used by the supported application logging frameworks. The most commonly used six log levels are (in order of lowest to highest): **TRACE**, **DEBUG**, **INFO**, **WARN**, **ERROR** and **FATAL**.

Log levels are used by log categories and handlers to limit the messages they are responsible for. Each log level has an assigned numeric value which indicates its order relative to other log levels. Log categories and handlers are assigned a log level and they only process log messages of that level or higher. For example a log handler with the level of **WARN** will only record messages of the levels **WARN**, **ERROR** and **FATAL**.

[Report a bug](#)

12.1.10. Supported Log Levels

Table 12.4. Supported Log Levels

Log Level	Value	Description
FINEST	300	-
FINER	400	-

Log Level	Value	Description
TRACE	400	Use for messages that provide detailed information about the running state of an application. Log messages of TRACE are usually only captured when debugging an application.
DEBUG	500	Use for messages that indicate the progress individual requests or activities of an application. Log messages of DEBUG are usually only captured when debugging an application.
FINE	500	-
CONFIG	700	-
INFO	800	Use for messages that indicate the overall progress of the application. Often used for application startup, shutdown and other major lifecycle events.
WARN	900	Use to indicate a situation that is not in error but is not considered ideal. May indicate circumstances that may lead to errors in the future.
WARNING	900	-
ERROR	1000	Use to indicate an error that has occurred that could prevent the current activity or request from completing but will not prevent the application from running.
SEVERE	1000	-
FATAL	1100	Use to indicate events that could cause critical service failure and application shutdown and possibly cause JBoss EAP 6 to shutdown.

[Report a bug](#)

12.1.11. About Log Categories

Log categories define a set of log messages to capture and one or more log handlers which will process the messages.

The log messages to capture are defined by their Java package of origin and log level. Messages from classes in that package and of that log level or lower are captured by the log category and sent to the specified log handlers.

Log categories can optionally use the log handlers of the root logger instead of their own handlers.

[Report a bug](#)

12.1.12. About the Root Logger

The root logger captures all log messages sent to the server (of a specified level) that are not captured by a log category. These messages are then sent to one or more log handlers.

By default the root logger is configured to use a console and a periodic log handler. The periodic log handler is configured to write to the file **server.log**. This file is sometimes referred to as the server log.

[Report a bug](#)

12.1.13. About Log Handlers

Log handlers define how captured log messages are recorded. The available log handlers are: **Console**, **File**, **Periodic**, **Size**, **Async**, **syslog**, **Periodic Size** and **Custom**.



NOTE

A log handler must be added to at least one logger to be active.

[Report a bug](#)

12.1.14. Types of Log Handlers

Console

Console log handlers write log messages to either the host operating system's standard out (**stdout**) or standard error (**stderr**) stream. These messages are displayed when JBoss EAP 6 is run from a command line prompt. The messages from a Console log handler are not saved unless the operating system is configured to capture the standard out or standard error stream.

File

File log handlers write log messages to a specified file.

Periodic

Periodic log handlers write log messages to a named file until a specified period of time has elapsed. Once the time period has passed then the file is renamed by appending the specified timestamp and the handler continues to write into a newly created log file with the original name.

Size

Size log handlers write log messages to a named file until the file reaches a specified size. When the file reaches a specified size, it is renamed with a numeric suffix and the handler continues to write into a newly created log file with the original name. Each size log handler must specify the maximum number of files to be kept in this fashion.

Periodic Size

Available from JBoss EAP 6.4. This is a combination of the Periodic and Size handlers and supports their combined attributes.

Once the current log file reaches the specified size, *or* the specified time period has passed, the file is renamed and the handler continues to write to a newly created log file with the original name.

Async

Async log handlers are wrapper log handlers that provide asynchronous behavior for one or more other log handlers. These are useful for log handlers that may have high latency or other performance problems such as writing a log file to a network file system.

Custom

Custom log handlers enable you to configure new types of log handlers that have been implemented. A custom handler must be implemented as a Java class that extends **java.util.logging.Handler** and be contained in a module.

syslog

Syslog handlers can be used to send messages to a remote logging server. This allows multiple applications to send their log messages to the same server, where they can all be parsed together.

[Report a bug](#)

12.1.15. About Log Formatters

A log formatter is the configuration property of a log handler that defines the appearance of log messages from that handler. It is a string that uses a syntax based on **java.util.Formatter** class.

For example the log formatter string from the default configuration, **%d{HH:mm:ss,SSS} %-5p [%c] (%t) %s%E%n**, creates log messages that look like:

```
15:53:26,546 INFO [org.jboss.as] (Controller Boot Thread) JBAS015951: Admin console listening on
http://127.0.0.1:9990
```

[Report a bug](#)

12.1.16. Log Formatter Syntax

Table 12.5. Log Formatter Syntax

Symbol	Description
%c	The category of the logging event
%p	The level of the log entry (info/debug/etc)
%P	The localized level of the log entry
%d	The current date/time (yyyy-MM-dd HH:mm:ss,SSS form)
%r	The relative time (milliseconds since the log was initialized)
%z	The time zone
%k	A log resource key (used for localization of log messages)
%m	The log message (including exception trace)
%s	The simple log message (no exception trace)
%e	The exception stack trace (no extended module information)

Symbol	Description
%E	The exception stack trace (with extended module information)
%t	The name of the current thread
%n	A newline character
%C	The class of the code calling the log method (slow)
%F	The filename of the class calling the log method (slow)
%I	The source location of the code calling the log method (slow)
%L	The line number of the code calling the log method (slow)
%M	The method of the code calling the log method (slow)
%x	The Nested Diagnostic Context
%X	The Message Diagnostic Context
%%	A literal percent character (escaping)

[Report a bug](#)

12.2. CONFIGURE LOGGING IN THE MANAGEMENT CONSOLE

The management console provides a graphical user interface for the configuration of the root logger, log handlers, and log categories. See [Section 3.3.1, “Management Console”](#) for more information about the Management Console.

To access logging configuration, follow the steps below.

Procedure 12.3. Access Logging configuration

1. Log in to the Management Console
2. Navigate to the logging subsystem configuration. This step varies between servers running as standalone servers and servers running in a managed domain.
 - o **Standalone Server**
Click on **Configuration**, expand **Core** in the **Subsystems** menu, and then click **Logging**.
 - o **Managed Domain**
Click on **Configuration**, select the profile to edit from the drop-down menu. Expand **Core** in the **Subsystems** menu, and then click **Logging**.

The tasks you can perform to configure the root logger are:

- Edit the log level.
- Add and remove log handlers.

The tasks you can perform to configure log categories are:

- Add and remove log categories.
- Edit log category properties.
- Add and remove log handlers from a category.

The main tasks you can perform to configure log handlers are:

- Adding new handlers.
- Configuring handlers.

All supported log handlers (including custom) can be configured in the management console.

[Report a bug](#)

12.3. LOGGING CONFIGURATION IN THE CLI

Prerequisite

The Management CLI must be running and connected to the relevant JBoss EAP instance. For further information see [Section 3.4.2, “Launch the Management CLI”](#)

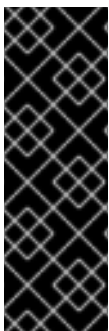
[Report a bug](#)

12.3.1. Configure the Root Logger with the CLI

The root logger configuration can be viewed and edited using the Management CLI.

The main tasks you will perform to configure the root logger are:

- Add log handlers to the root logger.
- Display the root logger configuration.
- Change the log level.
- Remove log handlers from the root logger.



IMPORTANT

When configuring a root logger in a logging profile for a standalone system, the root of the configuration path is **/subsystem=logging/logging-profile=NAME/** instead of **/subsystem=logging/**.

For a managed domain, you must specify which profile to use. You must add the profile name to the beginning of the configuration path for a managed domain, replacing **/subsystem=logging/** with **/profile=NAME/subsystem=logging/**.

Add a Log Handler to the Root Logger

Use the **add-handler** operation with the following syntax where *HANDLER* is the name of the log handler to be added.

```
/subsystem=logging/root-logger=ROOT:add-handler(name="HANDLER")
```

The log handler must already have been created before it can be added to the root logger.

Example 12.3. Root Logger add-handler operation

```
[standalone@localhost:9999 /] /subsystem=logging/root-logger=ROOT:add-  
handler(name="FILE")  
{  
  "outcome" => "success"  
}
```

Display the Contents of the Root Logger Configuration

Use the **read-resource** operation with the following syntax.

```
/subsystem=logging/root-logger=ROOT:read-resource
```

Example 12.4. Root Logger read-resource operation

```
[standalone@localhost:9999 /] /subsystem=logging/root-logger=ROOT:read-resource  
{  
  "outcome" => "success",  
  "result" => {  
    "filter" => undefined,  
    "filter-spec" => undefined,  
    "handlers" => [  
      "CONSOLE",  
      "FILE"  
    ],  
    "level" => "INFO"  
  }  
}
```

Set the Log Level of the Root Logger

Use the **write-attribute** operation with the following syntax where *LEVEL* is one of the supported log levels.

```
/subsystem=logging/root-logger=ROOT:write-attribute(name="level", value="LEVEL")
```

Example 12.5. Root Logger write-attribute operation to set the log level

```
[standalone@localhost:9999 /] /subsystem=logging/root-logger=ROOT:write-  
attribute(name="level", value="DEBUG")  
{  
  "outcome" => "success"  
}
```

Remove a Log Handler from the Root Logger

Use the **remove-handler** with the following syntax, where *HANDLER* is the name of the log handler to be removed.

```
/subsystem=logging/root-logger=ROOT:remove-handler(name="HANDLER")
```

Example 12.6. Remove a Log Handler

```
[standalone@localhost:9999 /] /subsystem=logging/root-logger=ROOT:remove-  
handler(name="FILE")  
{"outcome" => "success"}
```

[Report a bug](#)

12.3.2. Configure a Log Category in the CLI

Log categories can be added, removed and edited in the CLI.

The main tasks you will perform to configure a log category are:

- Add a new log category.
- Display the configuration of a log category.
- Set the log level.
- Add log handlers to a log category.
- Remove log handlers from a log category.
- Remove a log category.

IMPORTANT

When configuring a log category in a logging profile for a standalone system, the root of the configuration path is **/subsystem=logging/logging-profile=NAME/** instead of **/subsystem=logging/**.

For a managed domain, you must specify which profile to use. You must add the profile name to the beginning of the configuration path for a managed domain, replacing **/subsystem=logging/** with **/profile=NAME/subsystem=logging/**.

Add a log category

Use the **add** operation with the following syntax. Replace *CATEGORY* with the category to be added.

```
/subsystem=logging/logger=CATEGORY:add
```

Example 12.7. Adding a new log category

```
[standalone@localhost:9999 /] /subsystem=logging/logger=com.company.accounts.rec:add  
{"outcome" => "success"}
```

Display a log category configuration

Use the **read-resource** operation with the following syntax. Replace *CATEGORY* with the name of the category.

```
/subsystem=logging/logger=CATEGORY:read-resource
```

Example 12.8. Log Category read-resource operation

```
[standalone@localhost:9999 /]
/subsystem=logging/logger=org.apache.tomcat.util.modeler:read-resource
{
  "outcome" => "success",
  "result" => {
    "category" => "org.apache.tomcat.util.modeler",
    "filter" => undefined,
    "filter-spec" => undefined,
    "handlers" => undefined,
    "level" => "WARN",
    "use-parent-handlers" => true
  }
}
```

Set the log level

Use the **write-attribute** operation with the following syntax. Replace *CATEGORY* with the name of the log category and *LEVEL* with the log level that is to be set.

```
/subsystem=logging/logger=CATEGORY:write-attribute(name="level", value="LEVEL")
```

Example 12.9. Setting a log level

```
[standalone@localhost:9999 /] /subsystem=logging/logger=com.company.accounts.rec:write-attribute(name="level", value="DEBUG")
{"outcome" => "success"}
```

Set the log category to use the log handlers of the root logger.

Use the **write-attribute** operation with the following syntax. Replace *CATEGORY* with the name of the log category. Replace *BOOLEAN* with true for this log category to use the handlers of the root logger. Replace it with false if it is to use only its own assigned handlers.

```
/subsystem=logging/logger=CATEGORY:write-attribute(name="use-parent-handlers", value="BOOLEAN")
```

Example 12.10. Setting use-parent-handlers


```
[standalone@localhost:9999 /] /subsystem=logging/logger=com.company.accounts.rec:write-attribute(name="use-parent-handlers", value="true")
{"outcome" => "success"}
```

Add a log handlers to a log category

Use the **add-handler** operation with the following syntax. Replace *CATEGORY* with the name of the category and *HANDLER* with the name of the handler to be added.

```
/subsystem=logging/logger=CATEGORY:add-handler(name="HANDLER")
```

The log handler must already have been created before it can be added to the root logger.

Example 12.11. Adding a log handler

```
[standalone@localhost:9999 /] /subsystem=logging/logger=com.company.accounts.rec:add-handler(name="AccountsNFSAsync")
{"outcome" => "success"}
```

Remove a log handler from a log category

Use the **remove-handler** operation with the following syntax. Replace *CATEGORY* with the name of the category and *HANDLER* with the name of the log handler to be removed.

```
/subsystem=logging/logger=CATEGORY:remove-handler(name="HANDLER")
```

Example 12.12. Removing a log handler

```
[standalone@localhost:9999 /] /subsystem=logging/logger=jacorb:remove-handler(name="AccountsNFSAsync")
{"outcome" => "success"}
```

Remove a category

Use the **remove** operation with the following syntax. Replace *CATEGORY* with the name of the category to be removed.

```
/subsystem=logging/logger=CATEGORY:remove
```

Example 12.13. Removing a log category

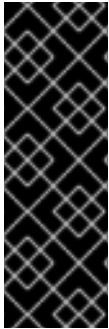
```
[standalone@localhost:9999 /]
/subsystem=logging/logger=com.company.accounts.rec:remove
{"outcome" => "success"}
```

12.3.3. Configure a Console Log Handler in the CLI

Console log handlers can be added, removed and edited in the CLI.

The main tasks you will perform to configure a console log handler are:

- Add a new console log handler.
- Display the configuration of a console log handler.
- Set the handler's log level.
- Set the target for the handler's output.
- Set the encoding used for the handler's output.
- Set the formatter used for the handler's output.
- Set whether the handler uses autoflush or not.
- Remove a console log handler.



IMPORTANT

When configuring a log handler in a logging profile for a standalone system, the root of the configuration path is **/subsystem=logging/logging-profile=NAME/** instead of **/subsystem=logging/**.

For a managed domain, you must specify which profile to use. You must add the profile name to the beginning of the configuration path for a managed domain, replacing **/subsystem=logging/** with **/profile=NAME/subsystem=logging/**.

Add a Console Log Handler

Use the **add** operation with the following syntax. Replace *HANDLER* with the console log handler to be added.

```
/subsystem=logging/console-handler=HANDLER:add
```

Example 12.14. Add a Console Log Handler

```
[standalone@localhost:9999 /] /subsystem=logging/console-handler=ERRORCONSOLE:add  
{"outcome" => "success"}
```

Display a console log handler configuration

Use the **read-resource** operation with the following syntax. Replace *HANDLER* with the name of the console log handler.

```
/subsystem=logging/console-handler=HANDLER:read-resource
```

Example 12.15. Display a console log handler configuration

```
[standalone@localhost:9999 /] /subsystem=logging/console-handler=CONSOLE:read-resource
{
  "outcome" => "success",
  "result" => {
    "autoflush" => true,
    "enabled" => true,
    "encoding" => undefined,
    "filter" => undefined,
    "filter-spec" => undefined,
    "formatter" => "%d{HH:mm:ss,SSS} %-5p [%c] (%t) %s%E%n",
    "level" => "INFO",
    "name" => "CONSOLE",
    "named-formatter" => "COLOR-PATTERN",
    "target" => "System.out"
  }
}
```

Set the Log Level

Use the **write-attribute** operation with the following syntax. Replace *HANDLER* with the name of the console log handler and *LEVEL* with the log level that is to be set.

```
/subsystem=logging/console-handler=HANDLER:write-attribute(name="level", value="INFO")
```

Example 12.16. Set the Log Level

```
[standalone@localhost:9999 /] /subsystem=logging/console-handler=ERRORCONSOLE:write-
attribute(name="level", value="TRACE")
{"outcome" => "success"}
```

Set the Target

Use the **write-attribute** operation with the following syntax. Replace *HANDLER* with the name of the console log handler. Replace *TARGET* with either **System.err** or **System.out** for the system error stream or standard out stream respectively.

```
/subsystem=logging/console-handler=HANDLER:write-attribute(name="target", value="TARGET")
```

Example 12.17. Set the Target

```
[standalone@localhost:9999 /] /subsystem=logging/console-handler=ERRORCONSOLE:write-
attribute(name="target", value="System.err")
{"outcome" => "success"}
```

Set the Encoding

Use the **write-attribute** operation with the following syntax. Replace *HANDLER* with the name of the console log handler. Replace *ENCODING* with the name of the required character encoding system.

```
/subsystem=logging/console-handler=HANDLER:write-attribute(name="encoding",
value="ENCODING")
```

Example 12.18. Set the Encoding

```
[standalone@localhost:9999 /] /subsystem=logging/console-handler=ERRORCONSOLE:write-
attribute(name="encoding", value="utf-8")
{"outcome" => "success"}
```

Set the Formatter

Use the **write-attribute** operation with the following syntax. Replace *HANDLER* with the name of the console log handler. Replace *FORMAT* with the required formatter string.

```
/subsystem=logging/console-handler=HANDLER:write-attribute(name="formatter",
value="FORMAT")
```

Example 12.19. Set the Formatter

```
[standalone@localhost:9999 /] /subsystem=logging/console-handler=ERRORCONSOLE:write-
attribute(name="formatter", value="%d{HH:mm:ss,SSS} %-5p [%c] (%t) %s%E%n")
{"outcome" => "success"}
```

Set the Auto Flush

Use the **write-attribute** operation with the following syntax. Replace *HANDLER* with the name of the console log handler. Replace *BOOLEAN* with **true** if this handler is to immediately write its output.

```
/subsystem=logging/console-handler=HANDLER:write-attribute(name="autoflush",
value="BOOLEAN")
```

Example 12.20. Set the Auto Flush

```
[standalone@localhost:9999 /] /subsystem=logging/console-handler=ERRORCONSOLE:write-
attribute(name="autoflush", value="true")
{"outcome" => "success"}
```

Remove a Console Log Handler

Use the **remove** operation with the following syntax. Replace *HANDLER* with the name of the console log handler to be removed.

```
/subsystem=logging/console-handler=HANDLER:remove
```

Example 12.21. Remove a Console Log Handler

```
[standalone@localhost:9999 /] /subsystem=logging/console-
handler=ERRORCONSOLE:remove
{"outcome" => "success"}
```

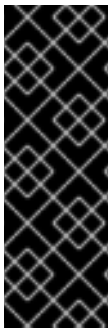
[Report a bug](#)

12.3.4. Configure a File Log Handler in the CLI

File log handlers can be added, removed and edited in the CLI.

The main tasks you will perform to configure a file log handler are:

- Add a new file log handler.
- Display the configuration of a file log handler
- Set the handler's log level.
- Set the handler's appending behavior.
- Set whether the handler uses autoflush or not.
- Set the encoding used for the handler's output.
- Specify the file to which the log handler will write.
- Set the formatter used for the handler's output.
- Remove a file log handler.



IMPORTANT

When configuring a log handler in a logging profile for a standalone system, the root of the configuration path is **/subsystem=logging/logging-profile=NAME/** instead of **/subsystem=logging/**.

For a managed domain, you must specify which profile to use. You must add the profile name to the beginning of the configuration path for a managed domain, replacing **/subsystem=logging/** with **/profile=NAME/subsystem=logging/**.

Add a file log handler

Use the **add** operation with the following syntax. Replace *PATH* with the filename for the file that the log is being written to. Replace *DIR* with the name of the directory where the file is to be located. The value of *DIR* can be a path variable.

```
/subsystem=logging/file-handler=HANDLER:add(file={"path"=>"PATH", "relative-to"=>"DIR"})
```

Example 12.22. Add a file log handler

```
[standalone@localhost:9999 /] /subsystem=logging/file-handler=accounts_log:add(file=
{"path"=>"accounts.log", "relative-to"=>"jboss.server.log.dir"})
{"outcome" => "success"}
```

Display a file log handler configuration

Use the **read-resource** operation with the following syntax. Replace *HANDLER* with the name of the file log handler.

```
/subsystem=logging/file-handler=HANDLER:read-resource
```

Example 12.23. Using the read-resource operation

```
[standalone@localhost:9999 /] /subsystem=logging/file-handler=accounts_log:read-resource
{
  "outcome" => "success",
  "result" => {
    "append" => true,
    "autoflush" => true,
    "enabled" => true,
    "encoding" => undefined,
    "file" => {
      "path" => "accounts.log",
      "relative-to" => "jboss.server.log.dir"
    },
    "filter" => undefined,
    "filter-spec" => undefined,
    "formatter" => "%d{HH:mm:ss,SSS} %-5p [%c] (%t) %s%E%n",
    "level" => "ALL",
    "name" => "accounts_log",
    "named-formatter" => undefined
  }
}
```

Set the Log level

Use the **write-attribute** operation with the following syntax. Replace *HANDLER* with the name of the file log handler. Replace *LOG_LEVEL_VALUE* with the log level that is to be set.

```
/subsystem=logging/file-handler=HANDLER:write-attribute(name="level",
value="LOG_LEVEL_VALUE")
```

Example 12.24. Changing the log level

```
/subsystem=logging/file-handler=accounts_log:write-attribute(name="level", value="DEBUG")
{"outcome" => "success"}
```

Set the append behaviour

Use the **write-attribute** operation with the following syntax. Replace *HANDLER* with the name of the file log handler. Replace *BOOLEAN* with **false** if you required that a new log file be created each time the application server is launched. Replace *BOOLEAN* with **true** if the application server should continue to use the same file.

```
/subsystem=logging/file-handler=HANDLER:write-attribute(name="append", value="BOOLEAN")
```

Example 12.25. Changing the append property

```
[standalone@localhost:9999 /] /subsystem=logging/file-handler=accounts_log:write-attribute(name="append", value="true")
{
  "outcome" => "success",
  "response-headers" => {
    "operation-requires-reload" => true,
    "process-state" => "reload-required"
  }
}
```

JBoss EAP 6 must be restarted for this change to take effect.

Set the Auto Flush

Use the **write-attribute** operation with the following syntax. Replace *HANDLER* with the name of the file log handler. Replace *BOOLEAN* with **true** if this handler is to immediately write its output.

```
/subsystem=logging/file-handler=HANDLER:write-attribute(name="autoflush", value="BOOLEAN")
```

Example 12.26. Changing the autoflush property

```
[standalone@localhost:9999 /] /subsystem=logging/file-handler=accounts_log:write-attribute(name="autoflush", value="false")
{"outcome" => "success"}
```

Set the Encoding

Use the **write-attribute** operation with the following syntax. Replace *HANDLER* with the name of the file log handler. Replace *ENCODING* with the name of the required character encoding system.

```
/subsystem=logging/file-handler=HANDLER:write-attribute(name="encoding", value="ENCODING")
```

Example 12.27. Set the Encoding

```
[standalone@localhost:9999 /] /subsystem=logging/file-handler=accounts_log:write-attribute(name="encoding", value="utf-8")
{"outcome" => "success"}
```

Change the file to which the log handler writes

Use the **write-attribute** operation with the following syntax. Replace *PATH* with the filename for the file that the log is being written to. Replace *DIR* with the name of the directory where the file is to be located. The value of *DIR* can be a path variable.

```
/subsystem=logging/file-handler=HANDLER:write-attribute(name="file", value={"path"=>"PATH", "relative-to"=>"DIR"})
```

Example 12.28. Change the file to which the log handler writes

```
[standalone@localhost:9999 /] /subsystem=logging/file-handler=accounts_log:write-attribute(name="file", value={"path"=>"accounts-debug.log", "relative-to"=>"jboss.server.log.dir"}) {"outcome" => "success"}
```

Set the Formatter

Use the **write-attribute** operation with the following syntax. Replace *HANDLER* with the name of the file log handler. Replace *FORMAT* with the required formatter string.

```
/subsystem=logging/file-handler=HANDLER:write-attribute(name="formatter", value="FORMAT")
```

Example 12.29. Set the Formatter

```
[standalone@localhost:9999 /] /subsystem=logging/file-handler=accounts_log:write-attribute(name="formatter", value="%d{HH:mm:ss,SSS} %-5p [%c] (%t) %s%E%n") {"outcome" => "success"}
```

Remove a File Log Handler

Use the **remove** operation with the following syntax. Replace *HANDLER* with the name of the file log handler to be removed.

```
/subsystem=logging/file-handler=HANDLER:remove
```

Example 12.30. Remove a File Log Handler

```
[standalone@localhost:9999 /] /subsystem=logging/file-handler=accounts_log:remove {"outcome" => "success"}
```

A log handler can only be removed if it is not being referenced by a log category or an async log handler.

[Report a bug](#)

12.3.5. Configure a Periodic Log Handler in the CLI

Periodic log handlers can be added, removed and edited in the CLI.

The main tasks you will perform to configure a periodic log handler are:

- Add a new periodic log handler.
- Display the configuration of a periodic log handler
- Set the handler's log level.

- Set the handler's appending behavior.
- Set whether or not the handler uses **autoflush**.
- Set the encoding used for the handler's output.
- Specify the file to which the log handler will write.
- Set the formatter used for the handler's output.
- Set the suffix for rotated logs.
- Remove a periodic log handler.

Each of those tasks are described below.



IMPORTANT

When configuring a log handler in a logging profile for a standalone system, the root of the configuration path is **/subsystem=logging/logging-profile=NAME/** instead of **/subsystem=logging/**.

For a managed domain, you must specify which profile to use. You must add the profile name to the beginning of the configuration path for a managed domain, replacing **/subsystem=logging/** with **/profile=NAME/subsystem=logging/**.

Add a new Periodic Rotating File log handler

Use the **add** operation with the following syntax.

```
/subsystem=logging/periodic-rotating-file-handler=HANDLER:add(file={"path"=>"PATH", "relative-to"=>"DIR"}, suffix="SUFFIX")
```

Replace *HANDLER* with the name of the log handler. Replace *PATH* with the filename for the file that the log is being written to. Replace *DIR* with the name of the directory where the file is to be located. The value of *DIR* can be a path variable. Replace *SUFFIX* with the file rotation suffix to be used.

Example 12.31. Add a new handler

```
[standalone@localhost:9999 /] /subsystem=logging/periodic-rotating-file-
handler=HOURLY_DEBUG:add(file={"path"=>"daily-debug.log", "relative-
to"=>"jboss.server.log.dir"}, suffix=".yyyy.MM.dd")
{"outcome" => "success"}
```

Display a Periodic Rotating File log handler configuration

Use the **read-resource** operation with the following syntax.

```
/subsystem=logging/periodic-rotating-file-handler=HANDLER:read-resource
```

Replace *HANDLER* with the name of the log handler.

Example 12.32. Using the read-resource operation

```
[standalone@localhost:9999 /] /subsystem=logging/periodic-rotating-file-
handler=HOURLY_DEBUG:read-resource
{
  "outcome" => "success",
  "result" => {
    "append" => true,
    "autoflush" => true,
    "enabled" => true,
    "encoding" => undefined,
    "file" => {
      "path" => "daily-debug.log",
      "relative-to" => "jboss.server.log.dir"
    },
    "filter" => undefined,
    "filter-spec" => undefined,
    "formatter" => "%d{HH:mm:ss,SSS} %-5p [%c] (%t) %s%E%n",
    "level" => "ALL",
    "name" => "HOURLY_DEBUG",
    "named-formatter" => undefined,
    "suffix" => ".yyyy.MM.dd"
  },
  "response-headers" => {"process-state" => "reload-required"}
}
```

Set the Log level

Use the **write-attribute** operation with the following syntax.

```
/subsystem=logging/periodic-rotating-file-handler=HANDLER:write-attribute(name="level".
value="LOG_LEVEL_VALUE")
```

Replace *HANDLER* with the name of the periodic log handler. Replace *LOG_LEVEL_VALUE* with the log level that is to be set.

Example 12.33. Set the log level

```
[standalone@localhost:9999 /] /subsystem=logging/periodic-rotating-file-
handler=HOURLY_DEBUG:write-attribute(name="level", value="DEBUG")
{"outcome" => "success"}
```

Set the append behavior

Use the **write-attribute** operation with the following syntax.

```
/subsystem=logging/periodic-rotating-handler=HANDLER:write-attribute(name="append",
value="BOOLEAN")
```

Replace *HANDLER* with the name of the periodic log handler. Replace *BOOLEAN* with **false** if you required that a new log file be created each time the application server is launched. Replace *BOOLEAN* with **true** if the application server should continue to use the same file.

JBoss EAP 6 must be restarted for this change to take effect.

Example 12.34. Set the append behavior

```
[standalone@localhost:9999 /] /subsystem=logging/periodic-rotating-file-
handler=HOURLY_DEBUG:write-attribute(name="append", value="true")
{
  "outcome" => "success",
  "response-headers" => {
    "operation-requires-reload" => true,
    "process-state" => "reload-required"
  }
}
```

Set the Auto Flush

Use the **write-attribute** operation with the following syntax.

```
/subsystem=logging/periodic-rotating-file-handler=HANDLER:write-attribute(name="autoflush",
value="BOOLEAN")
```

Replace *HANDLER* with the name of the periodic log handler. Replace *BOOLEAN* with **true** if this handler is to immediately write its output.

Example 12.35. Set the Auto Flush behavior

```
[standalone@localhost:9999 /] /subsystem=logging/periodic-rotating-file-
handler=HOURLY_DEBUG:write-attribute(name="autoflush", value="false")
{
  "outcome" => "success",
  "response-headers" => {"process-state" => "reload-required"}
}
```

Set the Encoding

Use the **write-attribute** operation with the following syntax.

```
/subsystem=logging/periodic-rotating-file-handler=HANDLER:write-attribute(name="encoding",
value="ENCODING")
```

Replace *HANDLER* with the name of the periodic log handler. Replace *ENCODING* with the name of the required character encoding system.

Example 12.36. Set the Encoding

```
[standalone@localhost:9999 /] /subsystem=logging/periodic-rotating-file-
handler=HOURLY_DEBUG:write-attribute(name="encoding", value="utf-8")
{"outcome" => "success"}
```

Change the file to which the log handler writes

Use the **write-attribute** operation with the following syntax.

```
/subsystem=logging/periodic-rotating-file-handler=HANDLER:write-attribute(name="file", value={
"path"=>"PATH", "relative-to"=>"DIR"})
```

Replace *HANDLER* with the name of the periodic log handler. Replace *PATH* with the filename for the file that the log is being written to. Replace *DIR* with the name of the directory where the file is to be located. The value of *DIR* can be a path variable.

Example 12.37. Change the file to which the log handler writes

```
[standalone@localhost:9999 /] /subsystem=logging/periodic-rotating-file-
handler=HOURLY_DEBUG:write-attribute(name="file", value={"path"=>"daily-debug.log",
"relative-to"=>"jboss.server.log.dir"})
{"outcome" => "success"}
```

Set the Formatter

Use the **write-attribute** operation with the following syntax.

```
/subsystem=logging/periodic-rotating-file-handler=HANDLER:write-attribute(name="formatter",
value="FORMAT")
```

Replace *HANDLER* with the name of the periodic log handler. Replace *FORMAT* with the required formatter string.

Example 12.38. Set the Formatter

```
[standalone@localhost:9999 /] /subsystem=logging/periodic-rotating-file-
handler=HOURLY_DEBUG:write-attribute(name="formatter", value="%d{HH:mm:ss,SSS} %-
5p [%c] (%t) %s%E%n")
{"outcome" => "success"}
```

Set the suffix for rotated logs

Use the **write-attribute** operation with the following syntax.

```
/subsystem=logging/periodic-rotating-file-handler=HANDLER:write-attribute(name="suffix",
value="SUFFIX")
```

Replace *HANDLER* with the name of the log handler. Replace *SUFFIX* with the required suffix string.

Example 12.39. Set the suffix for rotated log

```
[standalone@localhost:9999 /] /subsystem=logging/periodic-rotating-file-
handler=HOURLY_DEBUG:write-attribute(name="suffix", value=".yyyy-MM-dd-HH")
{"outcome" => "success"}
```

Remove a periodic log handler

Use the **remove** operation with the following syntax.

```
/subsystem=logging/periodic-rotating-file-handler=HANDLER:remove
```

Replace *HANDLER* with the name of the periodic log handler.

Example 12.40. Remove a periodic log handler

```
[standalone@localhost:9999 /] /subsystem=logging/periodic-rotating-file-  
handler=HOURLY_DEBUG:remove  
{"outcome" => "success"}
```

[Report a bug](#)

12.3.6. Configure a Size Log Handler in the CLI

Size rotated file log handlers can be added, removed and edited in the CLI.

The tasks you will perform to configure a size rotated file log handler are:

- Add a new log handler.
- Display the configuration of the log handler.
- Set the handler's log level.
- Set the handler's appending behavior.
- Set whether the handler uses autoflush or not.
- Set the encoding used for the handler's output.
- Specify the file to which the log handler will write.
- Set the formatter used for the handler's output.
- Set the maximum size of each log file.
- Set the maximum number of backup logs to keep.
- Set the rotate on boot option for the size rotation file handler.
- Set the suffix for rotated logs.
- Remove a log handler.

Each of these tasks are described below.



IMPORTANT

When configuring a log handler in a logging profile for a standalone system, the root of the configuration path is **/subsystem=logging/logging-profile=NAME/** instead of **/subsystem=logging/**.

For a managed domain, you must specify which profile to use. You must add the profile name to the beginning of the configuration path for a managed domain, replacing **/subsystem=logging/** with **/profile=NAME/subsystem=logging/**.

Add a new log handler

Use the **add** operation with the following syntax.

```
/subsystem=logging/size-rotating-file-handler=HANDLER:add(file={"path"=>"PATH", "relative-to"=>"DIR"})
```

Replace *HANDLER* with the name of the log handler. Replace *PATH* with the filename for the file that the log is being written to. Replace *DIR* with the name of the directory where the file is to be located. The value of *DIR* can be a path variable.

Example 12.41. Add a new log handler

```
[standalone@localhost:9999 /] /subsystem=logging/size-rotating-file-
handler=ACCOUNTS_TRACE:add(file={"path"=>"accounts_trace.log", "relative-
to"=>"jboss.server.log.dir"})
{"outcome" => "success"}
```

Display the configuration of the log handler

Use the **read-resource** operation with the following syntax.

```
/subsystem=logging/size-rotating-file-handler=HANDLER:read-resource
```

Replace *HANDLER* with the name of the log handler.

Example 12.42. Display the configuration of the log handler

```
[standalone@localhost:9999 /] /subsystem=logging/size-rotating-file-
handler=ACCOUNTS_TRACE:read-resource
{
  "outcome" => "success",
  "result" => {
    "append" => true,
    "autoflush" => true,
    "enabled" => true,
    "encoding" => undefined,
    "file" => {
      "path" => "accounts_trace.log",
      "relative-to" => "jboss.server.log.dir"
    },
    "filter" => undefined,
    "filter-spec" => undefined,
  }
}
```

```

"formatter" => "%d{HH:mm:ss,SSS} %-5p [%c] (%t) %s%E%n",
"level" => "ALL",
"max-backup-index" => 1,
"name" => "ACCOUNTS_TRACE",
"named-formatter" => undefined,
"rotate-on-boot" => false,
"rotate-size" => "2m",
"suffix" => undefined
}
}

```

Set the handler's log level

Use the **write-attribute** operation with the following syntax.

```

/subsystem=logging/size-rotating-file-handler=HANDLER:write-attribute(name="level",
value="LOG_LEVEL_VALUE")

```

Replace *HANDLER* with the name of the log handler. Replace *LOG_LEVEL_VALUE* with the log level that is to be set.

Example 12.43. Set the handler's log level

```

[standalone@localhost:9999 /] /subsystem=logging/size-rotating-file-
handler=ACCOUNTS_TRACE:write-attribute(name="level", value="TRACE")
{"outcome" => "success"}

```

Set the handler's appending behavior

Use the **write-attribute** operation with the following syntax.

```

/subsystem=logging/size-rotating-file-handler=HANDLER:write-attribute(name="append",
value="BOOLEAN")

```

Replace *HANDLER* with the name of the log handler. Replace *BOOLEAN* with **false** if you required that a new log file be created each time the application server is launched. Replace *BOOLEAN* with **true** if the application server should continue to use the same file.

JBoss EAP 6 must be restarted for this change to take effect.

Example 12.44. Set the handler's appending behavior

```

[standalone@localhost:9999 /] /subsystem=logging/size-rotating-file-
handler=ACCOUNTS_TRACE:write-attribute(name="append", value="true")
{
  "outcome" => "success",
  "response-headers" => {
    "operation-requires-reload" => true,
    "process-state" => "reload-required"
  }
}

```

-

Set whether the handler uses autoflush or not

Use the **write-attribute** operation with the following syntax.

```
/subsystem=logging/size-rotating-file-handler=HANDLER:write-attribute(name="autoflush",
value="BOOLEAN")
```

Replace *HANDLER* with the name of the log handler. Replace *BOOLEAN* with **true** if this handler is to immediately write its output.

Example 12.45. Set whether the handler uses autoflush or not

```
[standalone@localhost:9999 /] /subsystem=logging/size-rotating-file-
handler=ACCOUNTS_TRACE:write-attribute(name="autoflush", value="true")
{"outcome" => "success"}
```

Set the encoding used for the handler's output

Use the **write-attribute** operation with the following syntax.

```
/subsystem=logging/size-rotating-file-handler=HANDLER:write-attribute(name="encoding",
value="ENCODING")
```

Replace *HANDLER* with the name of the log handler. Replace *ENCODING* with the name of the required character encoding system.

Example 12.46. Set the encoding used for the handler's output

```
[standalone@localhost:9999 /] /subsystem=logging/size-rotating-file-
handler=ACCOUNTS_TRACE:write-attribute(name="encoding", value="utf-8")
{"outcome" => "success"}
```

Specify the file to which the log handler will write

Use the **write-attribute** operation with the following syntax.

```
/subsystem=logging/size-rotating-file-handler=HANDLER:write-attribute(name="file", value=
{"path"=>"PATH", "relative-to"=>"DIR"})
```

Replace *HANDLER* with the name of the log handler. Replace *PATH* with the filename for the file that the log is being written to. Replace *DIR* with the name of the directory where the file is to be located. The value of *DIR* can be a path variable.

Example 12.47. Specify the file to which the log handler will write

```
[standalone@localhost:9999 /] /subsystem=logging/size-rotating-file-
handler=ACCOUNTS_TRACE:write-attribute(name="file", value=
{"path"=>"accounts_trace.log", "relative-to"=>"jboss.server.log.dir"})
{"outcome" => "success"}
```


Set the formatter used for the handler's output

Use the **write-attribute** operation with the following syntax.

```
/subsystem=logging/size-rotating-file-handler=HANDLER:write-attribute(name="formatter",
value="FORMATTER")
```

Replace *HANDLER* with the name of the log handler. Replace *FORMAT* with the required formatter string.

Example 12.48. Set the formatter used for the handler's output

```
[standalone@localhost:9999 /] /subsystem=logging/size-rotating-file-
handler=ACCOUNTS_TRACE:write-attribute(name="formatter", value="%d{HH:mm:ss,SSS}
%-5p (%c) [%t] %s%E%n")
{"outcome" => "success"}
```

Set the maximum size of each log file

Use the **write-attribute** operation with the following syntax.

```
/subsystem=logging/size-rotating-file-handler=HANDLER:write-attribute(name="rotate-size",
value="SIZE")
```

Replace *HANDLER* with the name of the log handler. Replace *SIZE* with maximum file size.

Example 12.49. Set the maximum size of each log file

```
[standalone@localhost:9999 /] /subsystem=logging/size-rotating-file-
handler=ACCOUNTS_TRACE:write-attribute(name="rotate-size", value="50m")
{"outcome" => "success"}
```

Set the maximum number of backup logs to keep

Use the **write-attribute** operation with the following syntax.

```
/subsystem=logging/size-rotating-file-handler=HANDLER:write-attribute(name="max-backup-
index", value="NUMBER")
```

Replace *HANDLER* with the name of the log handler. Replace *NUMBER* with the required number of log files to keep.

Example 12.50. Set the maximum number of backup logs to keep

```
[standalone@localhost:9999 /] /subsystem=logging/size-rotating-file-
handler=ACCOUNTS_TRACE:write-attribute(name="max-backup-index", value="5")
{"outcome" => "success"}
```

Set the rotate-on-boot option on the size-rotating-file-handler

This option is only available for the **size-rotating-file-handler** file handler. It defaults to **false**, meaning a new log file is not created on server restart.

To change it, use the **write-attribute** operation with the following syntax.

```
/subsystem=logging/size-rotating-file-handler=HANDLER:write-attribute(name="rotate-on-boot", value="BOOLEAN")
```

Replace *HANDLER* with the name of the **size-rotating-file-handler** log handler. Replace *BOOLEAN* with **true** if a new **size-rotating-file-handler** log file should be created on restart.

Example 12.51. Specify to create a new **size-rotating-file-handler** log file on server restart

```
[standalone@localhost:9999 /] /subsystem=logging/size-rotating-file-  
handler=ACCOUNTS_TRACE:write-attribute(name="rotate-on-boot", value="true")  
{"outcome" => "success"}
```

Set the suffix for rotated logs

Use the **write-attribute** operation with the following syntax.

```
/subsystem=logging/size-rotating-file-handler=HANDLER:write-attribute(name="suffix",  
value="SUFFIX")
```

Replace *HANDLER* with the name of the log handler. Replace *SUFFIX* with the required suffix string.

Example 12.52. Set the suffix for rotated logs

```
[standalone@localhost:9999 /] /subsystem=logging/size-rotating-file-  
handler=ACCOUNTS_TRACE:write-attribute(name="suffix", value=".yyyy-MM-dd-HH")  
{"outcome" => "success"}
```

Remove a log handler

Use the **remove** operation with the following syntax.

```
/subsystem=logging/size-rotating-file-handler=HANDLER:remove
```

Replace *HANDLER* with the name of the log handler.

Example 12.53. Remove a log handler

```
[standalone@localhost:9999 /] /subsystem=logging/size-rotating-file-  
handler=ACCOUNTS_TRACE:remove  
{"outcome" => "success"}
```

[Report a bug](#)

12.3.7. Configure a Periodic Size Rotating Log Handler in the CLI

Periodic Size Rotating log handlers can be added, removed and edited in the CLI.

The main tasks you will perform to configure a periodic size rotating handler are:

- Add a new periodic size rotating handler.
- Display the configuration of a periodic size rotating handler.
- Set the handler's log level.
- Set the handler's appending behavior.
- Set whether or not the handler uses **autoflush**.
- Set the encoding used for the handler's output.
- Specify the file to which the log handler will write.
- Set the formatter used for the handler's output.
- Set the maximum size of each log file.
- Set the maximum number of backup logs to keep.
- Set the rotate on boot option for the periodic size rotating handler.
- Set the suffix for rotated logs.
- Remove a periodic size rotating handler.

Each of those tasks are described below.



IMPORTANT

When configuring a periodic size rotating handler in a logging profile for a standalone system, the root of the configuration path is **/subsystem=logging/logging-profile=NAME/** instead of **/subsystem=logging/**.

For a managed domain, you must specify which profile to use. You must add the profile name to the beginning of the configuration path for a managed domain, replacing **/subsystem=logging/** with **/profile=NAME/subsystem=logging/**.

Add a new Periodic Size Rotating handler

Use the **add** operation with the following syntax.

```
/subsystem=logging/periodic-size-rotating-file-handler=HANDLER:add(file={"path"=>"PATH",
"relative-to"=>"DIR"})
```

Replace *HANDLER* with the name of the log handler. Replace *PATH* with the filename for the file that the log is being written to. Replace *DIR* with the name of the directory where the file is to be located. The value of *DIR* can be a path variable.

Example 12.54. Add a new log handler

```
[standalone@localhost:9999 /] /subsystem=logging/periodic-size-rotating-file-
handler=PERIODIC_SIZE:add(file={"path"=>"periodic_size.log","relative-
to"=>"jboss.server.log.dir"},suffix=".yyyy.MM.dd")
{"outcome" => "success"}
```

Display the configuration of the log handler

Use the **read-resource** operation with the following syntax.

```
/subsystem=logging/periodic-size-rotating-file-handler=HANDLER:read-resource
```

Replace *HANDLER* with the name of the log handler.

Example 12.55. Display the configuration of the log handler

```
[standalone@localhost:9999 /] /subsystem=logging/periodic-size-rotating-file-
handler=PERIODIC_SIZE:read-resource
{
  "outcome" => "success",
  "result" => {
    "append" => true,
    "autoflush" => true,
    "enabled" => true,
    "encoding" => undefined,
    "file" => {
      "relative-to" => "jboss.server.log.dir",
      "path" => "periodic_size.log"
    },
    "filter-spec" => undefined,
    "formatter" => "%d{HH:mm:ss,SSS} %-5p [%c] (%t) %s%E%n",
    "level" => "ALL",
    "max-backup-index" => 1,
    "name" => "PERIODIC_SIZE",
    "named-formatter" => undefined,
    "rotate-on-boot" => false,
    "rotate-size" => "2m",
    "suffix" => ".yyyy.MM.dd"
  }
}
```

Set the handler's log level

Use the **write-attribute** operation with the following syntax.

```
/subsystem=logging/periodic-size-rotating-file-handler=HANDLER:write-attribute(name="level",
value="LOG_LEVEL_VALUE")
```

Replace *HANDLER* with the name of the log handler. Replace *LOG_LEVEL_VALUE* with the log level that is to be set.

Example 12.56. Set the handler's log level

```
[standalone@localhost:9999 /] /subsystem=logging/periodic-size-rotating-file-
handler=PERIODIC_SIZE:write-attribute(name="level", value="TRACE")
{"outcome" => "success"}
```

Set the handler's appending behavior

Use the **write-attribute** operation with the following syntax.

```
/subsystem=logging/periodic-size-rotating-file-handler=HANDLER:write-
attribute(name="append", value="BOOLEAN")
```

Replace *HANDLER* with the name of the log handler. Replace *BOOLEAN* with **false** if you required that a new log file be created each time the application server is launched. Replace *BOOLEAN* with **true** if the application server should continue to use the same file.

JBoss EAP 6 must be restarted for this change to take effect.

Example 12.57. Set the handler's appending behavior

```
[standalone@localhost:9999 /] /subsystem=logging/periodic-size-rotating-file-
handler=PERIODIC_SIZE:write-attribute(name="append", value="true")
{
  "outcome" => "success",
  "response-headers" => {
    "operation-requires-reload" => true,
    "process-state" => "reload-required"
  }
}
```

Set whether the handler uses autoflush or not

Use the **write-attribute** operation with the following syntax.

```
/subsystem=logging/periodic-size-rotating-file-handler=HANDLER:write-
attribute(name="autoflush", value="BOOLEAN")
```

Replace *HANDLER* with the name of the log handler. Replace *BOOLEAN* with **true** if this handler is to immediately write its output.

Example 12.58. Set whether the handler uses autoflush or not

```
[standalone@localhost:9999 /] /subsystem=logging/periodic-size-rotating-file-
handler=PERIODIC_SIZE:write-attribute(name="autoflush", value="true")
{"outcome" => "success"}
```

Set the encoding used for the handler's output

Use the **write-attribute** operation with the following syntax.

```
/subsystem=logging/periodic-size-rotating-file-handler=HANDLER:write-attribute(name="encoding", value="ENCODING")
```

Replace *HANDLER* with the name of the log handler. Replace *ENCODING* with the name of the required character encoding system.

Example 12.59. Set the encoding used for the handler's output

```
[standalone@localhost:9999 /] /subsystem=logging/periodic-size-rotating-file-handler=PERIODIC_SIZE:write-attribute(name="encoding", value="utf-8") {"outcome" => "success"}
```

Specify the file to which the log handler will write

Use the **write-attribute** operation with the following syntax.

```
/subsystem=logging/periodic-size-rotating-file-handler=HANDLER:write-attribute(name="file", value={"path"=>"PATH", "relative-to"=>"DIR"})
```

Replace *HANDLER* with the name of the log handler. Replace *PATH* with the filename for the file that the log is being written to. Replace *DIR* with the name of the directory where the file is to be located. The value of *DIR* can be a path variable.

Example 12.60. Specify the file to which the log handler will write

```
[standalone@localhost:9999 /] /subsystem=logging/periodic-size-rotating-file-handler=PERIODIC_SIZE:write-attribute(name="file", value={"path"=>"accounts_trace.log", "relative-to"=>"jboss.server.log.dir"}) {"outcome" => "success"}
```

Set the formatter used for the handler's output

Use the **write-attribute** operation with the following syntax.

```
/subsystem=logging/periodic-size-rotating-file-handler=HANDLER:write-attribute(name="formatter", value="FORMAT")
```

Replace *HANDLER* with the name of the log handler. Replace *FORMAT* with the required formatter string or name of the formatter as specified in the configuration file.

Example 12.61. Set the formatter used for the handler's output

```
[standalone@localhost:9999 /] /subsystem=logging/periodic-size-rotating-file-handler=PERIODIC_SIZE:write-attribute(name="formatter", value="%d{HH:mm:ss,SSS} %-5p (%c) [%t] %s%E%n") {"outcome" => "success"}
```

Set the maximum size of each log file

Use the **write-attribute** operation with the following syntax.

```
/subsystem=logging/periodic-size-rotating-file-handler=HANDLER:write-attribute(name="rotate-size", value="SIZE")
```

Replace *HANDLER* with the name of the log handler. Replace *SIZE* with maximum file size.

Example 12.62. Set the maximum size of each log file

```
[standalone@localhost:9999 /] /subsystem=logging/periodic-size-rotating-file-
handler=PERIODIC_SIZE:write-attribute(name="rotate-size", value="50m")
{"outcome" => "success"}
```

Set the maximum number of backup logs to keep

Use the **write-attribute** operation with the following syntax.

```
/subsystem=logging/periodic-size-rotating-file-handler=HANDLER:write-attribute(name="max-
backup-index", value="NUMBER")
```

Replace *HANDLER* with the name of the log handler. Replace *NUMBER* with the required number of log files to keep.

Example 12.63. Set the maximum number of backup logs to keep

```
[standalone@localhost:9999 /] /subsystem=logging/periodic-size-rotating-file-
handler=PERIODIC_SIZE:write-attribute(name="max-backup-index", value="5")
{"outcome" => "success"}
```

Set the rotate-on-boot option on the periodic-size-rotating-file-handler

It defaults to **false**, meaning a new log file is not created on server restart.

To change it, use the **write-attribute** operation with the following syntax.

```
/subsystem=logging/periodic-size-rotating-file-handler=HANDLER:write-attribute(name="rotate-
on-boot", value="BOOLEAN")
```

Replace *HANDLER* with the name of the **periodic-size-rotating-file-handler** log handler. Replace *BOOLEAN* with **true** if a new **periodic-size-rotating-file-handler** log file should be created on restart.

Example 12.64. Specify to create a new periodic-size-rotating-file-handler log file on server restart

```
[standalone@localhost:9999 /] /subsystem=logging/periodic-size-rotating-file-
handler=PERIODIC_SIZE:write-attribute(name="rotate-on-boot", value="true")
{"outcome" => "success"}
```

Set the suffix for rotated logs

Use the **write-attribute** operation with the following syntax.

```
/subsystem=logging/periodic-size-rotating-file-handler=HANDLER:write-attribute(name="suffix", value="SUFFIX")
```

Replace *HANDLER* with the name of the log handler. Replace *SUFFIX* with the required suffix string.

Example 12.65. Set a suffix for rotated logs

```
[standalone@localhost:9999 /] /subsystem=logging/periodic-size-rotating-file-  
handler=PERIODIC_SIZE:write-attribute(name="suffix", value=".yyyy-MM-dd-HH")  
{"outcome" => "success"}
```

Remove a log handler

Use the **remove** operation with the following syntax.

```
/subsystem=logging/periodic-size-rotating-file-handler=HANDLER:remove
```

Replace *HANDLER* with the name of the log handler.

Example 12.66. Remove a log handler

```
[standalone@localhost:9999 /] /subsystem=logging/periodic-size-rotating-file-  
handler=PERIODIC_SIZE:remove  
{"outcome" => "success"}
```

[Report a bug](#)

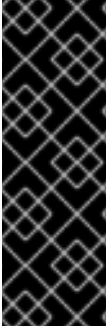
12.3.8. Configure a Async Log Handler in the CLI

Async log handlers can be added, removed and edited in the CLI.

The tasks you will perform to configure an async log handler are:

- Add a new async log handler
- Display the configuration of an async log handler
- Change the log level
- Set the queue length
- Set the overflow action
- Add sub-handlers
- Remove sub-handlers
- Remove an async log handler

Each of these tasks are described below.



IMPORTANT

When configuring a log handler in a logging profile for a standalone system, the root of the configuration path is **/subsystem=logging/logging-profile=NAME/** instead of **/subsystem=logging/**.

For a managed domain, you must specify which profile to use. You must add the profile name to the beginning of the configuration path for a managed domain, replacing **/subsystem=logging/** with **/profile=NAME/subsystem=logging/**.

Add a new async log handler

Use the **add** operation with the following syntax.

```
/subsystem=logging/async-handler=HANDLER:add(queue-length="LENGTH")
```

Replace *HANDLER* with the name of the log handler. Replace *LENGTH* with value of the maximum number of log requests that can be held in queue.

Example 12.67. Add a new async log handler

```
[standalone@localhost:9999 /] /subsystem=logging/async-handler=NFS_LOGS:add(queue-length="10")
{"outcome" => "success"}
```

Display the configuration of an async log handler

Use the **read-resource** operation with the following syntax.

```
/subsystem=logging/async-handler=HANDLER:read-resource
```

Replace *HANDLER* with the name of the log handler.

Example 12.68. Display the configuration of an async log handler

```
[standalone@localhost:9999 /] /subsystem=logging/async-handler=NFS_LOGS:read-resource
{
  "outcome" => "success",
  "result" => {
    "enabled" => true,
    "encoding" => undefined,
    "filter" => undefined,
    "filter-spec" => undefined,
    "formatter" => "%d{HH:mm:ss,SSS} %-5p [%c] (%t) %s%E%n",
    "level" => "ALL",
    "name" => "NFS_LOGS",
    "overflow-action" => "BLOCK",
    "queue-length" => "10",
    "subhandlers" => undefined
  },
  "response-headers" => {"process-state" => "reload-required"}
}
```

Change the log level

Use the **write-attribute** operation with the following syntax.

```
/subsystem=logging/async-handler=HANDLER:write-attribute(name="level",
value="LOG_LEVEL_VALUE")
```

Replace *HANDLER* with the name of the log handler. Replace *LOG_LEVEL_VALUE* with the log level that is to be set.

Example 12.69. Change the log level

```
[standalone@localhost:9999 /] /subsystem=logging/async-handler=NFS_LOGS:write-
attribute(name="level", value="INFO")
{"outcome" => "success"}
```

Set the queue length

Use the **write-attribute** operation with the following syntax.

```
/subsystem=logging/async-handler=HANDLER:write-attribute(name="queue-length",
value="LENGTH")
```

Replace *HANDLER* with the name of the log handler. Replace *LENGTH* with value of the maximum number of log requests that can be held in queue.

JBoss EAP 6 must be restarted for this change to take effect.

Example 12.70. Set the queue length

```
[standalone@localhost:9999 /] /subsystem=logging/async-handler=NFS_LOGS:write-
attribute(name="queue-length", value="150")
{
  "outcome" => "success",
  "response-headers" => {
    "operation-requires-reload" => true,
    "process-state" => "reload-required"
  }
}
```

Set the overflow action

Use the **write-attribute** operation with the following syntax.

```
/subsystem=logging/async-handler=HANDLER:write-attribute(name="overflow-action",
value="ACTION")
```

Replace *HANDLER* with the name of the log handler. Replace *ACTION* with either *DISCARD* or *BLOCK*.

Example 12.71. Set the overflow action

```
[standalone@localhost:9999 /] /subsystem=logging/async-handler=NFS_LOGS:write-attribute(name="overflow-action", value="DISCARD")
{"outcome" => "success"}
```

Add sub-handlers

Use the **add-handler** operation with the following syntax.

```
/subsystem=logging/async-handler=HANDLER:add-handler(name="SUBHANDLER")
```

Replace *HANDLER* with the name of the log handler. Replace *SUBHANDLER* with the name of the log handler that is to be added as a sub-handler of this async handler.

Example 12.72. Add sub-handlers

```
[standalone@localhost:9999 /] /subsystem=logging/async-handler=NFS_LOGS:add-handler(name="NFS_FILE")
{"outcome" => "success"}
```

Remove sub-handlers

Use the **remove-handler** operation with the following syntax.

```
/subsystem=logging/async-handler=HANDLER:remove-handler(name="SUBHANDLER")
```

Replace *HANDLER* with the name of the log handler. Replace *SUBHANDLER* with the name of the sub-handler to remove.

Example 12.73. Remove sub-handlers

```
[standalone@localhost:9999 /] /subsystem=logging/async-handler=NFS_LOGS:remove-handler(name="NFS_FILE")
{"outcome" => "success"}
```

Remove an async log handler

Use the **remove** operation with the following syntax.

```
/subsystem=logging/async-handler=HANDLER:remove
```

Replace *HANDLER* with the name of the log handler.

Example 12.74. Remove an async log handler

```
[standalone@localhost:9999 /] /subsystem=logging/async-handler=NFS_LOGS:remove
{"outcome" => "success"}
```

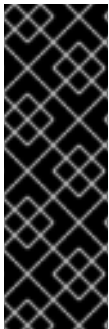
[Report a bug](#)

12.3.9. Configure a Custom Handler in the CLI

Custom handler can be added, removed, and edited in the CLI.

The main tasks you will perform to configure a custom handler are:

- Add a new custom handler.
- Display the configuration of a custom handler.
- Set the log level.
- Remove a custom handler.



IMPORTANT

When configuring a custom handler in a logging profile for a standalone system, the root of the configuration path is **/subsystem=logging/logging-profile=NAME/** instead of **/subsystem=logging/**.

For a managed domain, you must specify which profile to use. You must add the profile name to the beginning of the configuration path for a managed domain, replacing **/subsystem=logging/** with **/profile=NAME/subsystem=logging/**.

Add a New Custom Handler

Use the **add** operation with the following syntax.

```
/subsystem=logging/custom-handler="MyCustomHandler":add
```

Example 12.75. Adding a new custom handler

```
[standalone@localhost:9999 /] /subsystem=logging/custom-  
handler="MyCustomHandler":add(class="JdbcLogger",module="com.MyModule",formatter="%d{  
HH:mm:ss,SSS} %-5p [%c] (%t) %s%E%n",properties=  
{ "maxNumberOfDays"=>"90","fileName"=>"custom.log","compressBackups"=>"true"})
```

Display Custom Handler

Use the **read-resource** operation with the following syntax. Replace *CUSTOMHANDLER* with the name of the custom handler.

```
/subsystem=logging/custom-handler=CUSTOMHANDLER:read-resource
```

Example 12.76. Display a custom handler configuration

```
[standalone@localhost:9999 /] /subsystem=logging/custom-handler="MyCustomHandler":read-  
resource  
{  
  "outcome" => "success",  
  "result" => {
```

```

    "autoflush" => true,
    "enabled" => true,
    "encoding" => undefined,
    "filter" => undefined,
    "filter-spec" => undefined,
    "formatter" => "%d{HH:mm:ss,SSS} %-5p [%c] (%t) %s%E%n",
    "level" => "INFO",
    "name" => "CONSOLE",
    "named-formatter" => "COLOR-PATTERN",
    "target" => "System.out"
  }
}

```

Set the Log Level

Use the **write-attribute** operation with the following syntax. Replace *CUSTOMHANDLER* with the name of the log category and *LEVEL* with the log level that is to be set.

```

/subsystem=logging/custom-handler=CUSTOMHANDLER:write-attribute(name="level",
value="INFO")

```

Example 12.77. Set the Log Level

```

[standalone@localhost:9999 /] /subsystem=logging/custom-
handler="MyCustomHandler":write-attribute(name="level", value="TRACE")
{"outcome" => "success"}

```

Remove Custom Handler

Use the **remove** operation with the following syntax. Replace *CUSTOMHANDLER* with the name of the custom handler to be removed.

```

/subsystem=logging/custom-handler=CUSTOMHANDLER:remove

```

Example 12.78. Remove a custom handler

```

[standalone@localhost:9999 /] /subsystem=logging/custom-
handler="MyCustomHandler":remove
{"outcome" => "success"}

```

[Report a bug](#)

12.3.10. Configure a Syslog Handler in the CLI

The log manager for JBoss EAP 6 now contains a syslog handler. Syslog handlers can be used to send messages to a remote logging server that supports the **Syslog** protocol (RFC-3164 or RFC-5424). This allows the log messages of multiple applications to be sent to the same server, where they can be parsed together. This topic covers how to create and configure a syslog handler using the Management CLI, and the available configuration options.

For details of syslog handler attributes see [Section A.3, "Management Interface Audit Logging Reference"](#).

Prerequisites

- Access and the correct permissions for the Management CLI.

Procedure 12.4. Add a Syslog Handler

- Run the following command to add a syslog handler:

```
/subsystem=logging/syslog-handler=HANDLER_NAME:add
```

Procedure 12.5. Configure a Syslog Handler

- Run the following command to configure a syslog handler attribute:

```
/subsystem=logging/syslog-handler=HANDLER_NAME:write-  
attribute(name=ATTRIBUTE_NAME,value=ATTRIBUTE_VALUE)
```

Procedure 12.6. Remove a Syslog Handler

- Run the following command to remove an existing syslog handler:

```
/subsystem=logging/syslog-handler=HANDLER_NAME:remove
```

[Report a bug](#)

12.3.11. Configure a Custom Log Formatter in the CLI

Summary

In addition to the log formatter syntax specified in [Section 12.1.16, "Log Formatter Syntax"](#), a custom log formatter can be created for use with any log handler. This example procedure will demonstrate this by creating a XML formatter for a console log handler.

Prerequisites

- Access to the Management CLI for the JBoss EAP 6 server.
- A previously configured log handler. This example procedure uses a console log handler.

Procedure 12.7. Configure a Custom XML Formatter for a Log Handler

1. Create custom formatter.

In this example, the following command creates a custom formatter named **XML_FORMATTER** that uses the **java.util.logging.XMLFormatter** class.

```
[standalone@localhost:9999 /] /subsystem=logging/custom-  
formatter=XML_FORMATTER:add(class=java.util.logging.XMLFormatter,  
module=org.jboss.logmanager)
```

2. Register a custom formatter for the log handler you want to use it with.

In this example, the formatter from the previous step is added to a console log handler.

```
[standalone@localhost:9999 /] /subsystem=logging/console-handler=HANDLER:write-attribute(name=named-formatter, value=XML_FORMATTER)
```

3. Restart the JBoss EAP 6 server for the change to take effect.

```
[standalone@localhost:9999 /] shutdown --restart=true
```

Result

The custom XML formatter is added to the console log handler. Output to the console log will be formatted in XML, for example:

```
<record>
  <date>2014-03-11T13:02:53</date>
  <millis>1394539373833</millis>
  <sequence>116</sequence>
  <logger>org.jboss.as</logger>
  <level>INFO</level>
  <class>org.jboss.as.server.BootstrapListener</class>
  <method>logAdminConsole</method>
  <thread>282</thread>
  <message>JBAS015951: Admin console listening on http://%s:%d</message>
  <param>127.0.0.1</param>
  <param>9990</param>
</record>
```

[Report a bug](#)

12.4. PER-DEPLOYMENT LOGGING

12.4.1. About Per-deployment Logging

Per-deployment logging allows a developer to configure in advance the logging configuration for their application. When the application is deployed, logging begins according to the defined configuration. The log files created through this configuration contain information only about the behavior of the application.

This approach has advantages and disadvantages over using system-wide logging. An advantage is that the administrator of the JBoss EAP instance does not need to configure logging. A disadvantage is that the per-deployment logging configuration is read only on startup and so cannot be changed at runtime.

[Report a bug](#)

12.4.2. Disable Per-deployment Logging

Procedure 12.8. Disable Per-deployment Logging

- Two methods of disabling per-deployment logging are available. One works on all versions of JBoss EAP 6, while the other works only on JBoss EAP 6.3 and higher.

- **JBoss EAP 6 (all versions)**

Add the system property:

```
org.jboss.as.logging.per-deployment=false
```

- **JBoss EAP 6.3 (and higher)**

Exclude the logging subsystem using a **jboss-deployment-structure.xml** file. For details on how to do this, see *Exclude a Subsystem from a Deployment* in the *Development Guide*.

[Report a bug](#)

12.5. LOGGING PROFILES

12.5.1. About Logging Profiles



IMPORTANT

Logging profiles are only available in version 6.1.0 and later. They cannot be configured using the management console.

Logging profiles are independent sets of logging configuration that can be assigned to deployed applications. As with the regular logging subsystem, a logging profile can define handlers, categories and a root logger but cannot refer to configuration in other profiles or the main logging subsystem. The design of logging profiles mimics the logging subsystem for ease of configuration.

The use of logging profiles allows administrators to create logging configuration that are specific to one or more applications without affecting any other logging configuration. Because each profile is defined in the server configuration, the logging configuration can be changed without requiring that the affected applications be redeployed.

Each logging profile can have the following configuration:

- A unique name. This is required.
- Any number of log handlers.
- Any number of log categories.
- Up to one root logger.

An application can specify a logging profile to use in its **MANIFEST.MF** file, using the ***logging-profile*** attribute.

[Report a bug](#)

12.5.2. Create a new Logging Profile using the CLI

A new logging profile can be created using the CLI command below, replacing *NAME* with your required profile name:

```
/subsystem=logging/logging-profile=NAME:add
```

This will create a new empty profile to which handlers, categories and a root logger can be added.

[Report a bug](#)

12.5.3. Configuring a Logging Profile using the CLI

A logging profile can be configured with log handlers, categories and a root logger using almost exactly the same syntax as when using the main logging subsystem.

There are only two differences between configuring the main logging subsystem and the logging profile:

1. The root configuration path is **/subsystem=logging/logging-profile=NAME**
2. A logging profile cannot contain other logging profiles.

Refer to the appropriate logging management task:

- [Section 12.3.1, "Configure the Root Logger with the CLI"](#)
- [Section 12.3.2, "Configure a Log Category in the CLI"](#)
- [Section 12.3.3, "Configure a Console Log Handler in the CLI"](#)
- [Section 12.3.4, "Configure a File Log Handler in the CLI"](#)
- [Section 12.3.5, "Configure a Periodic Log Handler in the CLI"](#)
- [Section 12.3.6, "Configure a Size Log Handler in the CLI"](#)
- [Section 12.3.8, "Configure a Async Log Handler in the CLI"](#)

Example 12.79. Creating and Configuring a Logging Profile

Creating a logging profile and adding a category and file log handler.

1. Create the profile:

```
/subsystem=logging/logging-profile=accounts-app-profile:add
```

2. Create file handler

```
/subsystem=logging/logging-profile=accounts-app-profile/file-handler=ejb-trace-file:add(file={path=>"ejb-trace.log", "relative-to"=>"jboss.server.log.dir"})
```

```
/subsystem=logging/logging-profile=accounts-app-profile/file-handler=ejb-trace-file:write-attribute(name="level", value="DEBUG")
```

3. Create logger category

```
/subsystem=logging/logging-profile=accounts-app-profile/logger=com.company.accounts.ejbs:add(level=TRACE)
```

4. Assign file handler to category

```
/subsystem=logging/logging-profile=accounts-app-profile/logger=com.company.accounts.ejbs:add-handler(name="ejb-trace-file")
```

[Report a bug](#)

12.5.4. Specify a Logging Profile in an Application

An application specifies the logging profile to use in its **MANIFEST.MF** file.

Prerequisites:

1. You must know the name of the logging profile that has been setup on the server for this application to use. Ask your server administrator for the name of the profile to use.

Procedure 12.9. Add Logging Profile configuration to an Application

- **Edit MANIFEST.MF**

If your application does not have a **MANIFEST.MF** file: create one with the following content, replacing *NAME* with the required profile name.

```
Manifest-Version: 1.0
Logging-Profile: NAME
```

If your application already has a **MANIFEST.MF** file: add the following line to it, replacing *NAME* with the required profile name.

```
Logging-Profile: NAME
```

NOTE

If you are using Maven and the **maven-war-plugin**, you can put your MANIFEST.MF file in **src/main/resources/META-INF/** and add the following configuration to your **pom.xml** file.

```
<plugin>
  <artifactId>maven-war-plugin</artifactId>
  <configuration>
    <archive>
      <manifestFile>src/main/resources/META-INF/MANIFEST.MF</manifestFile>
    </archive>
  </configuration>
</plugin>
```

When the application is deployed it will use the configuration in the specified logging profile for its log messages.

[Report a bug](#)

12.5.5. Example Logging Profile Configuration

This example shows the configuration of a logging profile and the application that makes use of it. The CLI session is shown, the XML configuration that is generated, and the **MANIFEST.MF** file of the application.

The logging profile example has the following characteristics:

- The Name is **accounts-app-profile**.
- The Log Category is **com.company.accounts.ejbs**.
- The Log level **TRACE**.
- The Log handler is a file handler using the file **ejb-trace.log**.

Example 12.80. CLI session

```
localhost:bin user$ ./jboss-cli.sh -c
[standalone@localhost:9999 /] /subsystem=logging/logging-profile=accounts-app-profile:add
{"outcome" => "success"}

[standalone@localhost:9999 /] /subsystem=logging/logging-profile=accounts-app-profile/file-
handler=ejb-trace-file:add(file={path=>"ejb-trace.log", "relative-to"=>"jboss.server.log.dir"})
{"outcome" => "success"}

[standalone@localhost:9999 /] /subsystem=logging/logging-profile=accounts-app-profile/file-
handler=ejb-trace-file:write-attribute(name="level", value="DEBUG")
{"outcome" => "success"}

[standalone@localhost:9999 /] /subsystem=logging/logging-profile=accounts-app-
profile/logger=com.company.accounts.ejbs:add(level=TRACE)
{"outcome" => "success"}

[standalone@localhost:9999 /] /subsystem=logging/logging-profile=accounts-app-
profile/logger=com.company.accounts.ejbs:add-handler(name="ejb-trace-file")
{"outcome" => "success"}
```

Example 12.81. XML Configuration

```
<logging-profiles>
  <logging-profile name="accounts-app-profile">
    <file-handler name="ejb-trace-file">
      <level name="DEBUG"/>
      <file relative-to="jboss.server.log.dir" path="ejb-trace.log"/>
    </file-handler>
    <logger category="com.company.accounts.ejbs">
      <level name="TRACE"/>
      <handlers>
        <handler name="ejb-trace-file"/>
      </handlers>
    </logger>
  </logging-profile>
</logging-profiles>
```

Example 12.82. Application MANIFEST.MF file

Manifest-Version: 1.0
 Logging-Profile: accounts-app-profile

[Report a bug](#)

12.6. LOGGING CONFIGURATION PROPERTIES

12.6.1. Root Logger Properties

Table 12.6. Root Logger Properties

Property	Datatype	Description
level	String	The maximum level of log message that the root logger records.
handlers	String[]	A list of log handlers that are used by the root logger.
filter-spec	String	An expression value that defines a filter. The following expression defines a filter that excludes log entries that do <i>not</i> match a pattern: not(match("JBAS.*"))



NOTE

A **filter-spec** specified for the root logger is *not* inherited by other handlers. Instead a **filter-spec** must be specified per handler.

[Report a bug](#)

12.6.2. Log Category Properties

Table 12.7. Log Category Properties

Property	Datatype	Description
level	String	The maximum level of log message that the log category records.
handlers	String[]	A list of log handlers associated with the logger.
use-parent-handlers	Boolean	If set to true, this category will use the log handlers of the root logger in addition to any other assigned handlers.
category	String	The log category from which log messages will be captured.
filter-spec	String	An expression value that defines a filter. The following expression defines a filter that does not match a pattern: not(match("JBAS.*"))

[Report a bug](#)

12.6.3. Console Log Handler Properties

Table 12.8. Console Log Handler Properties

Property	Datatype	Description
level	String	The maximum level of log message the log handler records.
encoding	String	The character encoding scheme to be used for the output.
formatter	String	The log formatter used by this log handler.
named-formatter	String	The name of the defined formatter to be used on the handler.
target	String	The system output stream where the output of the log handler goes. This can be System.err or System.out for the system error stream or standard out stream respectively.
autoflush	Boolean	If set to true the log messages will be sent to the handlers target immediately upon receipt.
name	String	The unique identifier for this log handler.
enabled	Boolean	If set to true , the handler is enabled and functioning as normal. If set to false , the handler is ignored when processing log messages.
filter-spec	String	An expression value that defines a filter. The following expression defines a filter that does not match a pattern: not(match("JBAS.*"))

[Report a bug](#)

12.6.4. File Log Handler Properties

Table 12.9. File Log Handler Properties

Property	Datatype	Description
level	String	The maximum level of log message the log handler records.
encoding	String	The character encoding scheme to be used for the output.
formatter	String	The log formatter used by this log handler.
named-formatter	String	The name of the defined formatter to be used on the handler.

Property	Datatype	Description
append	Boolean	If set to true then all messages written by this handler will be appended to the file if it already exists. If set to false a new file will be created each time the application server launches. Changes to append require a server reboot to take effect.
autoflush	Boolean	If set to true the log messages will be sent to the handlers assigned file immediately upon receipt.
name	String	The unique identifier for this log handler.
file	Object	The object that represents the file where the output of this log handler is written to. It has two configuration properties, relative-to and path .
relative-to	String	This is a property of the file object and is the directory where the log file is written to. JBoss EAP 6 file path variables can be specified here. The jboss.server.log.dir variable points to the log/ directory of the server.
path	String	This is a property of the file object and is the name of the file where the log messages will be written. It is a relative path name that is appended to the value of the relative-to property to determine the complete path.
enabled	Boolean	If set to true , the handler is enabled and functioning as normal. If set to false , the handler is ignored when processing log messages.
filter-spec	String	An expression value that defines a filter. The following expression defines a filter that does not match a pattern: not(match("JBAS.*"))

[Report a bug](#)

12.6.5. Periodic Log Handler Properties

Table 12.10. Periodic Log Handler Properties

Property	Datatype	Description
append	Boolean	If set to true then all messages written by this handler will be appended to the file if it already exists. If set to false a new file will be created each time the application server launches. Changes to append require a server reboot to take effect.
autoflush	Boolean	If set to true the log messages will be sent to the handlers assigned file immediately upon receipt.
encoding	String	The character encoding scheme to be used for the output.

Property	Datatype	Description
formatter	String	The log formatter used by this log handler.
named-formatter	String	The name of the defined formatter to be used on the handler.
level	String	The maximum level of log message the log handler records.
name	String	The unique identifier for this log handler.
file	Object	Object that represents the file to which the output of this log handler is written. It has two configuration properties, relative-to and path .
relative-to	String	This is a property of the file object and is the directory containing the log file. File path variables can be specified here. The jboss.server.log.dir variable points to the log/ directory of the server.
path	String	This is a property of the file object and is the name of the file where the log messages will be written. It is a relative path name that is appended to the value of the relative-to property to determine the complete path.
suffix	String	<p>This String is appended to the filename of the rotated logs and is used to determine the frequency of rotation. The format of the suffix is a dot (.) followed by a date String which is able to be parsed by the SimpleDateFormat class. The log is rotated on the basis of the smallest time unit defined by the suffix. Note that the smallest time unit allowed in the suffix attribute is minutes.</p> <p>For example, a suffix value of .YYYY-MM-dd will result in daily log rotation. Assuming a log file named server.log and a suffix value of .YYYY-MM-dd, the log file rotated on 20 October 2014 would be named server.log.2014-10-19. For a periodic log handler, the suffix includes the previous value for the smallest time unit. In this example the value for dd is 19, the previous day.</p>
enabled	Boolean	If set to true , the handler is enabled and functioning as normal. If set to false , the handler is ignored when processing log messages.
filter-spec	String	An expression value that defines a filter. The following expression defines a filter that does not match a pattern: not(match("JBAS.*")) .

[Report a bug](#)

12.6.6. Size Log Handler Properties

Table 12.11. Size Log Handler Properties

Property	Datatype	Description
append	Boolean	If set to true then all messages written by this handler will be appended to the file if it already exists. If set to false a new file will be created each time the application server launches. Changes to append require a server reboot to take effect.
autoflush	Boolean	If set to true the log messages will be sent to the handlers assigned file immediately upon receipt.
encoding	String	The character encoding scheme to be used for the output.
formatter	String	The log formatter used by this log handler.
named-formatter	String	The name of the defined formatter to be used on the handler.
level	String	The maximum level of log message the log handler records.
name	String	The unique identifier for this log handler.
file	Object	Object that represents the file where the output of this log handler is written to. It has two configuration properties, relative-to and path .
relative-to	String	This is a property of the file object and is the directory containing the log file. File path variables can be specified here. The jboss.server.log.dir variable points to the log/ directory of the server.
path	String	This is a property of the file object and is the name of the file where the log messages will be written. It is a relative path name that is appended to the value of the relative-to property to determine the complete path.
rotate-size	Integer	The maximum size that the log file can reach before it is rotated. A single character appended to the number indicates the size units: b for bytes, k for kilobytes, m for megabytes, g for gigabytes. Eg. 50m for 50 megabytes.
max-backup-index	Integer	<p>The maximum number of rotated logs that are kept. When this number is reached, the oldest log is reused. Default: 1.</p> <p>Available from JBoss EAP 6.4. If the suffix attribute is used, the suffix of rotated log files is included in the rotation algorithm. When the log file is rotated, the oldest file whose name starts with name+suffix is deleted, the remaining rotated log files have their numeric suffix incremented and the newly rotated log file is given the numeric suffix 1.</p> <p>Assume a log file name server.log, a suffix value of .YYYY-mm, and max-backup-index value of 3. When the log file has been rotated twice, log file names could be server.log, server.log.2014-10.1 and server.log.2014-10.2. The next time the log file is rotated, server.log.2014-10.2 is deleted, server.log.2014-10.1 is renamed server.log.2014-10.2 and the newly rotated log file is named server.log.2014-10.1.</p>

Property	Datatype	Description
enabled	Boolean	If set to true , the handler is enabled and functioning as normal. If set to false , the handler is ignored when processing log messages.
filter-spec	String	An expression value that defines a filter. The following expression defines a filter that does not match a pattern: not(match("JBAS.*"))
rotate-on-boot	Boolean	If set to true , a new log file will be created on server restart. Default is false .
suffix	String	Available from JBoss EAP 6.4. This String is included in the suffix appended to rotated logs. The format of the suffix is a dot (.) followed by a date String which is able to be parsed by the SimpleDateFormat class. For example, assuming a log file named server.log and a suffix value of .YYYY-mm , the first log file rotated on 1 October 2014 would be named server.log.2014-10.1 . In this example, the 2014-10 portion of the suffix is derived from the value of suffix .

[Report a bug](#)

12.6.7. Periodic Size Rotating Log Handler Properties

Table 12.12. Periodic Size Log Handler Properties

Property	Datatype	Description
append	Boolean	If set to true then all messages written by this handler will be appended to the file if it already exists. If set to false a new file will be created each time the application server launches. Changes to append require a server reboot to take effect.
rotate-size	String	The maximum size that the log file can reach before it is rotated.
max-backup-index	Integer	The maximum number of size-rotated logs that are kept. When this number is reached, the oldest log is reused. Default: 1. This setting only applies to logs that are rotated based on file size. Note that if a file is rotated with a date format suffix, it will not be purged using the max-backup-index option.

Property	Datatype	Description
suffix	String	<p>This String is appended to the filename of the rotated logs and is used to determine the frequency of rotation. The format of the suffix is a dot (.) followed by a date String which is able to be parsed by the SimpleDateFormat class. The log is rotated on the basis of the smallest time unit defined by the suffix. Note that the smallest time unit allowed in the suffix attribute is minutes.</p> <p>For example, a suffix value of .YYYY-MM-dd will result in daily log rotation. Assuming a log file named server.log and a suffix value of .YYYY-MM-dd, the log file rotated on 20 October 2014 would be named server.log.2014-10-19. For a periodic log handler, the suffix includes the previous value for the smallest time unit. In this example the value for dd is 19, the previous day.</p>
autoflush	Boolean	If set to true the log messages will be sent to the handlers assigned file immediately upon receipt.
file	Object	Object that represents the file where the output of this log handler is written to. It has two configuration properties, relative-to and path .
relative-to	String	This is a property of the file object and is the directory containing the log file. File path variables can be specified here. The jboss.server.log.dir variable points to the log/ directory of the server.
path	String	This is a property of the file object and is the name of the file where the log messages will be written. It is a relative path name that is appended to the value of the relative-to property to determine the complete path.
enabled	Boolean	If set to true , the handler is enabled and functioning as normal. If set to false , the handler is ignored when processing log messages.
filter-spec	String	An expression value that defines a filter. The following expression defines a filter that does not match a pattern: not(match("JBAS.*"))
rotate-on-boot	Boolean	If set to true , a new log file will be created on server restart. Default is false .
formatter	String	The log formatter used by this log handler.
level	String	The minimum level of log message the log handler records.
name	String	The unique identifier for this log handler.
named-formatter	String	The name of the defined formatter to be used on the handler.

Property	Datatype	Description
encoding	String	The character encoding scheme to be used for the output.

[Report a bug](#)

12.6.8. Async Log Handler Properties

Table 12.13. Async Log Handler Properties

Property	Datatype	Description
level	String	The maximum level of log message the log handler records.
name	String	The unique identifier for this log handler.
encoding	String	The character encoding scheme to be used for the output.
formatter	String	The log formatter used by this log handler.
queue-length	Integer	Maximum number of log messages that will be held by this handler while waiting for sub-handlers to respond.
overflow-action	String	How this handler responds when its queue length is exceeded. This can be set to BLOCK or DISCARD . BLOCK makes the logging application wait until there is available space in the queue. This is the same behavior as a non-async log handler. DISCARD allows the logging application to continue but the log message is deleted.
subhandlers	String[]	This is the list of log handlers to which this async handler passes its log messages.
enabled	Boolean	If set to true , the handler is enabled and functioning as normal. If set to false , the handler is ignored when processing log messages.
filter-spec	String	An expression value that defines a filter. The following expression defines a filter that does not match a pattern: not(match("JBAS.*"))

[Report a bug](#)

12.7. SAMPLE XML CONFIGURATION FOR LOGGING

12.7.1. Sample XML Configuration for the Root Logger

```
<root-logger>
  <level name="INFO"/>
  <handlers>
```

```
<handler name="CONSOLE"/>
<handler name="FILE"/>
</handlers>
</root-logger>
```

[Report a bug](#)

12.7.2. Sample XML Configuration for a Log Category

```
<logger category="com.company.accounts.rec">
<handlers>
<handler name="accounts-rec"/>
</handlers>
</logger>
```

[Report a bug](#)

12.7.3. Sample XML Configuration for a Console Log Handler

```
<console-handler name="CONSOLE">
<level name="INFO"/>
<formatter>
<pattern-formatter pattern="%d{HH:mm:ss,SSS} %-5p [%c] (%t) %s%E%n"/>
</formatter>
</console-handler>
```

[Report a bug](#)

12.7.4. Sample XML Configuration for a File Log Handler

```
<file-handler name="accounts-rec-trail" autoflush="true">
<level name="INFO"/>
<file relative-to="jboss.server.log.dir" path="accounts-rec-trail.log"/>
<append value="true"/>
</file-handler>
```

[Report a bug](#)

12.7.5. Sample XML Configuration for a Periodic Log Handler

```
<periodic-rotating-file-handler name="FILE">
<formatter>
<pattern-formatter pattern="%d{HH:mm:ss,SSS} %-5p [%c] (%t) %s%E%n"/>
</formatter>
<file relative-to="jboss.server.log.dir" path="server.log"/>
<suffix value=".yyyy-MM-dd"/>
<append value="true"/>
</periodic-rotating-file-handler>
```

[Report a bug](#)

12.7.6. Sample XML Configuration for a Size Log Handler

```
<size-rotating-file-handler name="accounts_debug" autoflush="false">
  <level name="DEBUG"/>
  <file relative-to="jboss.server.log.dir" path="accounts-debug.log"/>
  <rotate-size value="500k"/>
  <max-backup-index value="5"/>
  <append value="true"/>
</size-rotating-file-handler>
```

[Report a bug](#)

12.7.7. Sample XML Configuration for a Periodic Size Rotating Log Handler

```
<periodic-size-rotating-file-handler name="Periodic_size_rotating_Handler" autoflush="false">
  <level name="INFO"/>
  <file relative-to="jboss.server.log.dir" path="Sample.log"/>
  <max-backup-index value="1"/>
  <suffix value=".yyyy.MM.dd"/>
  <append value="false"/>
</periodic-size-rotating-file-handler>
```

[Report a bug](#)

12.7.8. Sample XML Configuration for a Async Log Handler

```
<async-handler name="Async_NFS_handlers">
  <level name="INFO"/>
  <queue-length value="512"/>
  <overflow-action value="block"/>
  <subhandlers>
    <handler name="FILE"/>
    <handler name="accounts-record"/>
  </subhandlers>
</async-handler>
```

[Report a bug](#)

CHAPTER 13. INFINISPAN

13.1. ABOUT INFINISPAN

Infinispan is a Java data grid platform. It provides a [JSR-107](#) compatible cache interface for managing cached data.

The following Infinispan cache containers are used in JBoss Enterprise Application Platform 6:

- **web** for Web Session Clustering
- **ejb** for Stateful Session Bean Clustering
- **hibernate** for entity caching
- **singleton** for singleton caching

Each cache container defines a "repl" and a "dist" cache. These caches should not be used directly by user applications.



IMPORTANT

Users can add more cache containers and caches, and reference them via JNDI. However, this is unsupported in JBoss Enterprise Application Platform 6.

For more information about Infinispan functionality and configuration options see the [Infinispan Documentation](#).

[Report a bug](#)

13.2. CLUSTERING MODES

Clustering can be configured in two different ways in JBoss EAP 6 using Infinispan. The correct method for your application will depend on your requirements. There is a trade off between availability, consistency, reliability and scalability with each mode. Before choosing a clustering mode, you must identify what are the most important features of your network for you, and balance those requirements.

Replicated Mode

Replicated Mode automatically detects and adds new instances on the cluster. Changes made to these instances will be replicated to all nodes on the cluster. Replicated mode typically works best in small clusters because of the amount of information that has to be replicated over the network. Infinispan can be configured to use UDP multicast, which alleviates network traffic congestion to a degree.

Distribution Mode

Distribution mode allows Infinispan to scale the cluster linearly. Distribution mode uses a consistent hash algorithm to determine where in a cluster a new node should be placed. The number of copies of information to be kept is configurable. There is a trade off between the number of copies kept, durability of the data and performance: the more copies that are kept, the more impact on performance, but the less likely you are to lose data in a server failure. The hash algorithm also works to reduce network traffic by locating entries without multicasting or storing metadata.

One should consider using Distribution (dist) mode as a caching strategy when the cluster size exceeds 6-8 nodes. With Distribution mode, data is distributed to only a subset of nodes within the cluster, as opposed to all nodes (default Replicated mode).

Synchronous and Asynchronous Replication

Replication can be performed either in synchronous or asynchronous mode, and the mode chosen depends on your requirements and your application. With synchronous replication, the thread that handles the user request is blocked until replication has been successful. Only when the replication is successful, a response is sent back to the client and the thread is released. Synchronous replication has an impact on network traffic because it requires a response from each node in the cluster. It has the advantage, however, of ensuring that all modifications have been made to all nodes in the cluster.

Asynchronous replication is carried out in the background. Infinispan implements a replication queue, which is used by a background thread to carry out replication. Replication is triggered either on a time basis, or on the queue size. A replication queue allows increased performance because there is no conversation being carried out between the cluster nodes. The trade off with asynchronous replication is that it is not quite so accurate. Failed replication attempts are written to a log, not notified in real time.

[Report a bug](#)

13.3. CACHE CONTAINERS

Cache Containers

A cache container is repository for the caches used by a subsystem. For Infinispan default cache containers are defined in the configuration xml files (standalone-ha.xml, standalone-full-ha.xml, domain.xml). One cache is defined as the default cache, which is the cache that will be used for clustering.

Example 13.1. Cache container definitions from standalone-ha.xml configuration file

```
<subsystem xmlns="urn:jboss:domain:infinispan:1.5">
  <cache-container name="singleton" aliases="cluster ha-partition" default-cache="default">
    <transport lock-timeout="60000"/>
    <replicated-cache name="default" mode="SYNC" batching="true">
      <locking isolation="REPEATABLE_READ"/>
    </replicated-cache>
  </cache-container>
  <cache-container name="web" aliases="standard-session-cache" default-cache="repl"
module="org.jboss.as.clustering.web.infinispan">
    <transport lock-timeout="60000"/>
    <replicated-cache name="repl" mode="ASYNC" batching="true">
      <file-store/>
    </replicated-cache>
    <replicated-cache name="sso" mode="SYNC" batching="true"/>
    <distributed-cache name="dist" l1-lifespan="0" mode="ASYNC" batching="true">
      <file-store/>
    </distributed-cache>
  </cache-container>
```

Note the default cache defined in each cache container. In this example, in the **web** cache container, the **repl** cache is defined as the default. The **repl** cache will therefore be used when clustering web sessions.

The cache containers and cache attributes can be configured using the Management Console or CLI commands, but it is not advisable to change the names of either cache containers or caches.

Configure Cache Containers

Cache containers for Infinispan can be configured using the CLI or the Management Console.

Procedure 13.1. Configure the Infinispan Cache Containers in the Management Console

1. Select the **Configuration** tab from the top of the screen.
2. For Domain mode only, select either **ha** or **full-ha** from the drop down menu at top left.
3. Expand the **Subsystems** menu, then expand the **Infinispan** menu. Select **Cache Containers**.
4. Select a cache container from the **Cache Containers** table.
5. **Add, Remove or Set Default Cache Container**
 - a. To create a new cache container, click **Add** from the **Cache Containers** table.
 - b. To remove a cache container, select the cache container in the **Cache Containers** table. Click **Remove** and click **OK** to confirm.
 - c. To set a cache container as default, click **Set Default**, enter a cache container name from the drop down list, click **Save** to confirm.
6. To add or update the attributes of a cache container, select the cache container in the **Cache Containers** table. Select one from the **Attributes**, **Transport** and **Aliases** tabs in the **Details** area of the screen, and click **Edit**. For help about the content of the **Attributes**, **Transport** and **Aliases** tabs, click **Need Help?**

Procedure 13.2. Configure the Infinispan Cache Containers in the Management CLI

1. To get a list of configurable attributes, enter the following CLI command:

```
| /profile=profile name/subsystem=infinispan/cache-container=container name:read-resource
```

2. You can use the Management CLI to add, remove and update cache containers. Before issuing any commands to do with cache containers, ensure that you use the correct profile in the Management CLI command.

- a. **Add a Cache Container**

To add a cache container base your command on the following example:

```
| /profile=profile-name/subsystem=infinispan/cache-container="cache container name":add
```

- b. **Remove a Cache Container**

To remove a cache container base your command on the following example:

```
| /profile=profile-name/subsystem=infinispan/cache-container="cache container name":remove
```


c. Update Cache Container attributes

Use the write-attribute operation to write a new value to an attribute. You can use tab completion to help complete the command string as you type, as well as to expose the available attributes. The following example updates statistics-enabled to true.

```
/profile=profile name/subsystem=infinispan/cache-container=cache container name:write-attribute(name=statistics-enabled,value=true)
```

[Report a bug](#)

13.4. ABOUT INFINISPAN STATISTICS

Runtime statistics about Infinispan cache and cache-container objects can be enabled for monitoring purposes. Statistics collection is not enabled by default for performance reasons.



WARNING

Enabling Infinispan statistics can have a negative impact on the performance of the Infinispan subsystem. Statistics should be enabled only when required.

Statistics collection can be enabled for each cache-container, cache or both. The statistics option for each cache overrides the option for the cache-container. Enabling or disabling statistics collection for a cache container will cause all caches in that container to inherit the setting, unless they explicitly specify their own. If only a cache-container is enabled for statistics, useful statistics are available.

[Report a bug](#)

13.5. ENABLE INFINISPAN STATISTICS COLLECTION

Statistics collection can be enabled from either the startup configuration file (for example **standalone.xml**, **standalone-ha.xml**, **domain.xml**) or the Management CLI.

[Report a bug](#)

13.5.1. Enable Infinispan Statistics Collection in the Startup Configuration File

Procedure 13.3. Enable Infinispan Statistics in the Startup Configuration File

- Add the attribute **statistics-enabled=VALUE** to the required **cache-container** or **cache** XML tag under the Infinispan subsystem.

Example 13.2. Enable Statistics Collection for a cache

```
<replicated-cache name="sso" mode="SYNC" batching="true" statistics-enabled="true"/>
```

Example 13.3. Enable Statistics Collection for `acache-container`

```
<cache-container name="singleton" aliases="cluster ha-partition" default-cache="default"
statistics-enabled="true">
```

[Report a bug](#)

13.5.2. Enable Infinispan Statistics Collection from the Management CLI**Procedure 13.4. Enable Infinispan Statistics Collection from the Management CLI**

In this procedure:

- **CACHE_CONTAINER** is the preferred **cache-container** (for example, **web**)
- **CACHE_TYPE** is the preferred cache type (for example, **distributed-cache**)
- **CACHE** is the cache name (for example, **dist**)

1. Enter the following command:

Example 13.4. Enable Statistics Collection for `acache`

```
/subsystem=infinispan/cache-
container=CACHE_CONTAINER/CACHE_TYPE=CACHE:write-attribute(name=statistics-
enabled,value=true)
```

Example 13.5. Enable Statistics Collection for `acache-container`

```
/subsystem=infinispan/cache-container=CACHE_CONTAINER:write-
attribute(name=statistics-enabled,value=true)
```

2. Enter the following command to reload the server:

```
reload
```

NOTE

To undefine an attribute, enter the following command:

```
/subsystem=infinispan/cache-
container=CACHE_CONTAINER/CACHE_TYPE=CACHE:undefine-
attribute(name=statistics-enabled)
```

```
/subsystem=infinispan/cache-container=CACHE_CONTAINER:undefine-
attribute(name=statistics-enabled)
```

[Report a bug](#)

13.5.3. Verify Infinispan Statistics Collection is Enabled

Procedure 13.5. Verify Infinispan Statistics Collection is Enabled

Depending on whether you are confirming that statistics collection is enabled on a **cache** or **cache-container**, use one of the following Management CLI commands.

- ○ For a **cache**

```
/subsystem=infinispan/cache-
container=CACHE_CONTAINER/CACHE_TYPE=CACHE:read-attribute(name=statistics-
enabled)
```

- For a **cache-container**

```
/subsystem=infinispan/cache-container=CACHE_CONTAINER:read-
attribute(name=statistics-enabled)
```

[Report a bug](#)

13.6. SWITCHING TO DISTRIBUTED CACHE MODE FOR WEB SESSION REPLICATION

Change the web cache-container for Infinispan subsystem to **dist** from **repl**. The **owners** attribute defines the number of cluster nodes that will hold the session data. One of them is called primary owner, and the others are backup owners.

Following is the CLI command to update the cache mode to **dist** with three owners assuming an **ha** profile is being used:

```
/subsystem=infinispan/cache-container=web/:write-attribute(name=default-cache,value=dist)
/subsystem=infinispan/cache-container=web/distributed-cache=dist/:write-
attribute(name=owners,value=3)
```

Running the above CLI will yield the following output:

```
<cache-container name="web" aliases="standard-session-cache" default-cache="dist"
module="org.jboss.as.clustering.web.infinispan">
  <transport lock-timeout="60000"/>
  ...
  <distributed-cache name="dist" owners="3" l1-lifespan="0" mode="ASYNC" batching="true">
    <file-store/>
  </distributed-cache>
</cache-container>
```

[Report a bug](#)

13.7. JGROUPS

13.7.1. About JGroups

JGroups is a messaging toolkit which allows developers to create reliable messaging applications where system reliability is an issue. JGroups can be used to create clusters whose nodes can send messages to each other.

The JGroups subsystem provides all the communication mechanisms for how the servers in a cluster talk to each other. EAP is preconfigured with two JGroups stacks.

- `udp` - the nodes in the cluster use UDP (User Datagram Protocol) multicasting to communicate with each other. UDP is generally faster but less reliable than TCP.
- `tcp` - the nodes in the cluster use TCP (Transmission Control Protocol) to communicate with each other. TCP tends to be slower than UDP, but will more reliably deliver data to its destination.

The preconfigured stacks can be used, or you can define your own to suit your system's specific requirements.

[Report a bug](#)

13.8. JGROUPS TROUBLESHOOTING

13.8.1. Nodes do not form a cluster

Make sure your machine is set up correctly for IP multicast. There are 2 test programs that can be used to detect this: `McastReceiverTest` and `McastSenderTest`. For example:

```
java -cp EAP_HOME/bin/client/jboss-client.jar org.jgroups.tests.McastReceiverTest -mcast_addr 230.11.11.11 -port 5555 -bind_addr YOUR_BIND_ADDRESS
```

Then in another window start **`McastSenderTest`**:

```
java -cp EAP_HOME/bin/client/jboss-client.jar org.jgroups.tests.McastSenderTest -mcast_addr 230.11.11.11 -port 5555 -bind_addr YOUR_BIND_ADDRESS
```

If you want to bind to a specific network interface card (NIC), use **`-bind_addr 192.168.0.2`**, where `192.168.0.2` is the IP address of the NIC to which you want to bind. Use this parameter in both the sender and the receiver.

You should be able to type in the **`McastSenderTest`** window and see the output in the **`McastReceiverTest`** window. If not, try to use **`-ttl 32`** in the sender. If this still fails, consult a system administrator to help you setup IP multicast correctly, and ask the admin to make sure that multicast will work on the interface you have chosen or, if the machines have multiple interfaces, ask to be told the correct interface. Once you know multicast is working properly on each machine in your cluster, you can repeat the above test to test the network, putting the sender on one machine and the receiver on another.

[Report a bug](#)

13.8.2. Causes of missing heartbeats in FD

Sometimes a member is suspected by FD because a heartbeat ack has not been received for some time T (defined by `timeout` and `max_tries`). This can have multiple reasons, e.g. in a cluster of A,B,C,D; C can be suspected if (note that A pings B, B pings C, C pings D and D pings A):

- B or C are running at 100% CPU for more than T seconds. So even if C sends a heartbeat ack to B, B may not be able to process it because it is at 100%
- B or C are garbage collecting, same as above.
- A combination of the 2 cases above
- The network loses packets. This usually happens when there is a lot of traffic on the network, and the switch starts dropping packets (usually broadcasts first, then IP multicasts, TCP packets last).
- B or C are processing a callback. Let us say C received a remote method call over its channel and takes $T+1$ seconds to process it. During this time, C will not process any other messages, including heartbeats, and therefore B will not receive the heartbeat ack and will suspect C.

[Report a bug](#)

CHAPTER 14. JVM

14.1. ABOUT JVM

14.1.1. About JVM Settings

Configuration of Java Virtual Machine (JVM) settings varies between the managed domain and standalone server instances. In a managed domain, the JVM settings are declared in **host.xml** and **domain.xml** configuration files, and determined by the domain controller components responsible for starting and stopping server processes. In a standalone server instance, the server startup processes can pass command line settings at startup. These can be declared from the command line or via the **System Properties** screen in the Management Console.



NOTE

System properties not configured in `JAVA_OPTS` cannot be used for properties used by JBoss Modules at boot time, nor can it be used for other JVM properties such as **`java.util.logging.manager`**.

Managed Domain

An important feature of the managed domain is the ability to define JVM settings at multiple levels. You can configure custom JVM settings at the host level, by server group, or by server instance. The more specialized child elements will override the parent configuration, allowing for the declaration of specific server configurations without requiring exclusions at the group or host level. This also allows the parent configuration to be inherited by the other levels until settings are either declared in the configuration files or passed at runtime.

Example 14.1. JVM settings in the domain configuration file

The following example shows a JVM declaration for a server group in the **domain.xml** configuration file.

```
<server-groups>
  <server-group name="main-server-group" profile="default">
    <jvm name="default">
      <heap size="64m" max-size="512m"/>
    </jvm>
    <socket-binding-group ref="standard-sockets"/>
  </server-group>
  <server-group name="other-server-group" profile="default">
    <jvm name="default">
      <heap size="64m" max-size="512m"/>
    </jvm>
    <socket-binding-group ref="standard-sockets"/>
  </server-group>
</server-groups>
```

In this instance a server group called **main-server-group** is declaring a heap size of 64 megabytes, and a maximum heap size of 512 megabytes. Any server that belongs to this group will inherit these settings. You can change these settings for the group as a whole, by the host, or the individual server.

Example 14.2. Domain settings in the host configuration file

The following example shows a JVM declaration for a server group in the **host.xml** configuration file.

```
<servers>
  <server name="server-one" group="main-server-group" auto-start="true">
    <jvm name="default"/>
  </server>
  <server name="server-two" group="main-server-group" auto-start="true">
    <jvm name="default">
      <heap size="64m" max-size="256m"/>
    </jvm>
    <socket-bindings port-offset="150"/>
  </server>
  <server name="server-three" group="other-server-group" auto-start="false">
    <socket-bindings port-offset="250"/>
  </server>
</servers>
```

In this instance, a server named **server-two** belongs to the server group named **main-server-group**, inheriting the JVM settings from the **default** JVM group. In the previous example, the main heap size for **main-server-group** was set at 512 megabytes. By declaring the lower maximum heap size of 256 megabytes, **server-two** can override the **domain.xml** settings to fine-tune performance as desired.

Standalone server settings at runtime

The JVM settings for standalone server instances can be declared at runtime by setting the **JAVA_OPTS** environment variable before starting the server. An example of setting the **JAVA_OPTS** environment variable at the Linux command-line is:

```
[user@host bin]$ export JAVA_OPTS="-Xmx1024M"
```

The same setting can be used in a Microsoft Windows environment, as follows:

```
C:\> set JAVA_OPTS="Xmx1024M"
```

Alternatively, JVM settings can be added to the **standalone.conf** file found in the **EAP_HOME/bin** folder, which contains examples of options to pass to the JVM.



WARNING

Setting the **JAVA_OPTS** environment variable will re-define the default values for the **JAVA_OPTS** environment variable. This can break or terminate the start of JBoss EAP.

[Report a bug](#)

14.1.2. Display the JVM Status in the Management Console

Prerequisites

- [Section 2.2.2, “Start JBoss EAP 6 as a Standalone Server”](#)
- [Section 2.2.4, “Start JBoss EAP 6 as a Managed Domain”](#)
- [Section 3.3.2, “Log in to the Management Console”](#)

Java Virtual Machine (JVM) status can be displayed in the Management Console for either the standalone server or a managed domain. The console shows the heap usage, non heap usage, and thread usage of the server. While the statistics are not displayed in real-time, you can refresh the console display to provide an up-to-date overview of JVM resources.

The JVM status shows the following values.

Table 14.1. JVM Status Attributes

Type	Description
Max	The maximum amount of memory that can be used for memory management. The maximum available memory is shown by the light grey bar.
Used	The amount of used memory. The amount of used memory is shown by the dark grey bar.
Committed	The amount of memory that is committed for the Java virtual machine to use. The committed memory is shown by the dark grey bar.

Procedure 14.1. Display the JVM Status in the Management Console

- **Display the JVM status for a standalone server instance**
Select the **Runtime** tab from the top of the screen. Expand the **Status** menu, then expand the **Platform** menu. Select **JVM**.
 - **Display the JVM status for a managed domain**
Select the **Runtime** tab from the top of the screen. Expand the **Server Status** menu, then expand the **Platform** menu. Select **JVM**.
2. The managed domain can provide visibility of all server instances in the server group, but will only allow you to view one server at a time by selecting from the server menu. To view the status of other servers in your server group, click **Change Server** left of the screen to select from the host and servers displayed in your group. Select the required server or host and the JVM details will change. Click **Close** to finish.

Result

The status of the JVM settings for the server instance are displayed.

[Report a bug](#)

14.1.3. Configuring JVM

The `<jvm></jvm>` tags support the usage of `<jvm-options></jvm-options>`, which can be used to add parameters to the JVM configuration using the `<option value="VALUE"/>` tag.

Example 14.3. Use of <jvm-options>

```

<jvm name="default">
  <heap size="1303m" max-size="1303m"/>
  <permgen max-size="256m"/>
  <jvm-options>
    <option value="-XX:+UseCompressedOops"/>
  </jvm-options>
</jvm>

```

Configuring JVM Using CLI

To configure JVM using CLI, use the following syntax:

```

# cd /server-group=main-server-group/jvm=default

# :add-jvm-option(jvm-option="-XX:+UseCompressedOops")
{
  "outcome" => "success",
  "result" => undefined,
  "server-groups" => {"main-server-group" => {"host" => {"master" => {
    "server-one" => {"response" => {
      "outcome" => "success",
      "response-headers" => {
        "operation-requires-restart" => true,
        "process-state" => "restart-required"
      }
    }
  }},
  "server-two" => {"response" => {
    "outcome" => "success",
    "response-headers" => {
      "operation-requires-restart" => true,
      "process-state" => "restart-required"
    }
  }
}
}}
}}
}

# :read-resource

# Expected Result:

[domain@localhost:9999 jvm=default] :read-resource
{
  "outcome" => "success",
  "result" => {
    "agent-lib" => undefined,
    "agent-path" => undefined,
    "env-classpath-ignored" => undefined,
    "environment-variables" => undefined,
    "heap-size" => "1303m",
    "java-agent" => undefined,
    "java-home" => undefined,
    "jvm-options" => ["-XX:+UseCompressedOops"],

```

```

"max-heap-size" => "1303m",
"max-permgen-size" => "256m",
"permgen-size" => undefined,
"stack-size" => undefined,
"type" => undefined
}
}

```

Removing jvm-options Entry

To remove jvm-options entry, use the following syntax:

```

# cd /server-group=main-server-group/jvm=default

# :remove-jvm-option(jvm-option="-XX:+UseCompressedOops")

# Expected Result:

[domain@localhost:9999 jvm=default] :remove-jvm-option(jvm-option="-XX:+UseCompressedOops")
{
  "outcome" => "success",
  "result" => undefined,
  "server-groups" => {"main-server-group" => {"host" => {"master" => {
    "server-one" => {"response" => {
      "outcome" => "success",
      "response-headers" => {
        "operation-requires-restart" => true,
        "process-state" => "restart-required"
      }
    }},
    "server-two" => {"response" => {
      "outcome" => "success",
      "response-headers" => {
        "operation-requires-restart" => true,
        "process-state" => "restart-required"
      }
    }
  }
}
}}
}}
}
}

```

Specifying 32-Bit or 64-Bit Architecture

In some environments, such as Hewlett-Packard HP-UX and Solaris, the **-d32** or **-d64** switch is used to specify whether to run in a 32-bit or 64-bit JVM respectively. The default is 32-bit if neither option is indicated.

Procedure 14.2. Specifying 64-Bit Architecture for a Standalone Server

1. Open the **EAP_HOME/bin/standalone.conf** file.
2. Add the following line to append the **-d64** option to **JAVA_OPTS**.

```
JAVA_OPTS="$JAVA_OPTS -d64"
```

Procedure 14.3. Specifying 64-Bit Architecture for a Managed Domain

When running in a managed domain, you can specify the 64-bit environment for host and process controllers in addition to server instances.

1. Set the host and process controllers to run in the 64-bit JVM.
 - a. Open the **EAP_HOME/bin/domain.conf** file.
 - b. Add the following line to append the **-d64** option to **JAVA_OPTS**. Ensure that this is inserted before **PROCESS_CONTROLLER_JAVA_OPTS** and **HOST_CONTROLLER_JAVA_OPTS** are set.

```
JAVA_OPTS="$JAVA_OPTS -d64"
```

Example 14.4. JVM options in domain.conf

```
#
# Specify options to pass to the Java VM.
#
if [ "x$JAVA_OPTS" = "x" ]; then
  JAVA_OPTS="-Xms64m -Xmx512m -XX:MaxPermSize=256m -
  Djava.net.preferIPv4Stack=true"
  JAVA_OPTS="$JAVA_OPTS -
  Djboss.modules.system.pkgs=$JBOSS_MODULES_SYSTEM_PKGS -
  Djava.awt.headless=true"
  JAVA_OPTS="$JAVA_OPTS -Djboss.modules.policy-permissions=true"
  JAVA_OPTS="$JAVA_OPTS -d64"
else
  echo "JAVA_OPTS already set in environment; overriding default settings with
  values: $JAVA_OPTS"
fi
```

2. Set the server instances to run in the 64-bit JVM.
 - a. Add **-d64** as a JVM option in the appropriate host.xml configuration files.

Example 14.5. JVM options in host.xml

```
<jvms>
  <jvm name="default">
    <heap size="64m" max-size="256m"/>
    <permgen size="256m" max-size="256m"/>
    <jvm-options>
      <option value="-server"/>
      <option value="-d64"/>
    </jvm-options>
  </jvm>
</jvms>
```

**NOTE**

The **-d64** option can be added for the appropriate hosts using the Management CLI with the following command:

```
/host=HOST_NAME/jvm=default:add-jvm-option(jvm-option="-d64")
```

[Report a bug](#)

14.1.4. About the Java Security Manager

The Java Security Manager is a class that manages the external boundary of the Java Virtual Machine (JVM) sandbox, controlling how code executing within the JVM can interact with resources outside the JVM. When the Java Security Manager is activated, the Java API checks with the security manager for approval before executing a wide range of potentially unsafe operations. The Java Security Manager uses a security policy to determine whether a given action will be allowed or denied.

[Report a bug](#)

14.1.5. About Java Security Policies

A Java Security policy is a set of defined permissions for different classes of code. The Java Security Manager compares actions requested by applications against the security policy. If an action is allowed by the policy, the Security Manager will permit that action to take place. If the action is not allowed by the policy, the Security Manager will deny that action. The security policy can define permissions based on the location of code, on the code's signature, or based on the subject's principals.

Security policy grant entries consist of the following configuration elements:

CodeBase

The URL location (excluding the host and domain information) where the code originates from. This parameter is optional.

SignedBy

The alias used in the keystore to reference the signer whose private key was used to sign the code. This can be a single value or a comma-separated list of values. This parameter is optional. If omitted, presence or lack of a signature has no impact on the Java Security Manager.

Principals

A list of ***principal_type/principal_name*** pairs, which must be present within the executing thread's principal set. The Principals entry is optional. If it is omitted, it signifies that the principals of the executing thread will have no impact on the Java Security Manager.

Permissions

A permission is the access which is granted to the code. Many permissions are provided as part of the Java Enterprise Edition 6 (Java EE 6) specification.

[Report a bug](#)

14.1.6. Write a Java Security Policy

An application called **policytool** is included with most JDK and JRE distributions, for the purpose of creating and editing Java security policies. Detailed information about **policytool** is linked from <http://docs.oracle.com/javase/6/docs/technotes/tools/>. Alternatively, you can also write a security policy using a text editor.

Procedure 14.4. Setup a new Java Security Manager Policy

1. Start **policytool**.

Start the **policytool** tool in one of the following ways.

- **Red Hat Enterprise Linux**

From your GUI or a command prompt, run **/usr/bin/policytool**.

- **Microsoft Windows Server**

Run **policytool.exe** from your Start menu or from the **bin** of your Java installation. The location can vary.

2. Create a policy.

To create a policy, select **Add Policy Entry**. Add the parameters you need, then click **Done**.



NOTE

Use VFS to specify paths for applications deployed on JBoss EAP. On Linux the path is: **vfs:/content/application.war**. On Microsoft Windows it is: **vfs:\${user.dir}/content/application.war**.

For example:

```
grant codeBase "vfs:/content/application.war/-" {
  permission java.util.PropertyPermission "*", "read";
};
```

3. Edit an existing policy

Select the policy from the list of existing policies, and select the **Edit Policy Entry** button. Edit the parameters as needed.

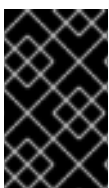
4. Delete an existing policy.

Select the policy from the list of existing policies, and select the **Remove Policy Entry** button.

[Report a bug](#)

14.1.7. Run JBoss EAP 6 Within the Java Security Manager

From JBoss EAP 6.4 and onwards, running JBoss EAP 6 within the Java Security Manager (JSM) is done using the **secmgr** option.



IMPORTANT

Direct usage of the **-Djava.security.manager** Java system property is no longer possible. This previous method used in older versions of JBoss EAP 6 to enable the Java Security Manager is now only supported as a fallback mechanism in the JBoss EAP startup scripts.

**NOTE**

From JBoss EAP 6.4 and onwards, custom security managers cannot be used.

The following procedure guides you through the steps of configuring your JBoss EAP 6 instance to run within the Java Security Manager using a specified security policy.

Prerequisites

- Before you follow this procedure, you need to write a security policy using the **policytool** application which is included in the Java Development Kit (JDK) or the Java SE Runtime Environment (JRE). Alternatively, you can write a security policy using a text editor.

Security policies will be needed for any user deployments that require permissions. This procedure assumes that your policy is located at ***EAP_HOME/bin/server.policy***.

- The domain or standalone server must be completely stopped before you edit any configuration files.

If you are using JBoss EAP 6 in a Managed Domain, you must perform the following procedure on each physical host or instance in your domain.

Procedure 14.5. Configure the Java Security Manager for JBoss EAP 6**1. Open the Configuration File**

Open the configuration file for editing. The configuration file you need to edit depends on whether you use a Managed Domain or standalone server, as well as your operating system.

- **Managed Domain**

- For Linux: ***EAP_HOME/bin/domain.conf***
- For Windows: ***EAP_HOME\bin\domain.conf.bat***

- **Standalone Server**

- For Linux: ***EAP_HOME/bin/standalone.conf***
- For Windows: ***EAP_HOME\bin\standalone.conf.bat***

2. Enable the Java Security Manager

Use one of the methods below to enable the Java Security Manager:

- Use the **-secmgr** option with your JBoss EAP 6 server startup script.
- Uncomment the **SECMGR="true"** line in the configuration file:

- **On Linux:**

```
# Uncomment this to run with a security manager enabled
SECMGR="true"
```

- **On Windows:**

```
rem # Uncomment this to run with a security manager enabled
set "SECMGR=true"
```

3. Specify the Java Security Policy

You can use **-Djava.security.policy** to specify the exact location of your security policy. It should go onto one line only, with no line break. Using **==** when setting **-Djava.security.policy** specifies that the security manager will use *only* the specified policy file. Using **=** specifies that the security manager will use the specified policy *combined with* the policy set in the **policy.url** section of **JAVA_HOME/jre/lib/security/java.security**.

In your relevant JBoss EAP 6 configuration file, add your security policy Java options. If you are using a Managed Domain, ensure that this is inserted before where **PROCESS_CONTROLLER_JAVA_OPTS** and **HOST_CONTROLLER_JAVA_OPTS** are set.

- On Linux:

```
JAVA_OPTS="$JAVA_OPTS -Djava.security.policy==EAP_HOME/bin/server.policy -
Djboss.home.dir=EAP_HOME"
```

- On Windows:

```
set "JAVA_OPTS=%JAVA_OPTS% -
Djava.security.policy==EAP_HOME\bin\server.policy -Djboss.home.dir=EAP_HOME"
```

4. Start the Domain or Server

Start the domain or server as normal.

[Report a bug](#)

14.1.8. IBM JDK and the Java Security Manager

Some versions of the IBM JDK use a default policy provider which does not work correctly with a JBoss EAP security policy. If you are having problems using an IBM JDK to host JBoss EAP with the Java Security Manager enabled, you must change the JRE configuration to use the standard policy provider.

To modify the JRE configuration for the IBM JDK, edit the **JAVA_HOME/jre/lib/security/java.security** file, and set the **policy.provider** value to **sun.security.provider.PolicyFile**.

```
policy.provider=sun.security.provider.PolicyFile
```

[Report a bug](#)

14.1.9. Debug Security Manager Policies

You can enable debugging information to help you troubleshoot security policy-related issues. The **java.security.debug** option configures the level of security-related information reported. The command **java -Djava.security.debug=help** will produce help output with the full range of debugging options. Setting the debug level to **all** is useful when troubleshooting a security-related failure whose cause is completely unknown, but for general use it will produce too much information. A sensible general default is **access:failure**.

Procedure 14.6. Enable general debugging

- This procedure will enable a sensible general level of security-related debug information. Add the following line to the server configuration file.
 - If the JBoss EAP 6 instance is running in a managed domain the line is added to the

If the JBoss EAP 6 instance is running in a managed domain, the line is added to the **bin/domain.conf** file for Linux or the **bin\domain.conf.bat** file for Windows.

- If the JBoss EAP 6 instance is running as a standalone server, the line is added to the **bin/standalone.conf** file for Linux, or the **bin\standalone.conf.bat** file for Windows.

Linux

```
JAVA_OPTS="$JAVA_OPTS -Djava.security.debug=access:failure"
```

Windows

```
set "JAVA_OPTS=%JAVA_OPTS% -Djava.security.debug=access:failure"
```

Result

A general level of security-related debug information has been enabled.

[Report a bug](#)

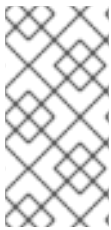
CHAPTER 15. WEB SUBSYSTEM

15.1. CONFIGURE THE WEB SUBSYSTEM

The Web subsystem can be configured using the Management Console or the Management CLI.

In the Management Console, click the **Configuration** tab at the top of the screen, expand the **Subsystems** menu and then expand the **Web** menu. Click on the **Servlet/HTTP** menu item to configure **Global** settings, also the **JSP**, **Connector** and **Virtual Servers** settings. Click on the **Web Services** menu item to configure the web services subsystem.

In the Management CLI, all Web Subsystem parameters are configured using the standard command format.



NOTE

The **mod_cluster** component is only available if your profile is **ha** or **full-ha**, in a managed domain, or if you start your standalone server with the **standalone-ha** or **standalone-full-ha** profile. For details of **mod_cluster** configuration see [Section 17.6.2, "Configure the mod_cluster Subsystem"](#).

[Report a bug](#)

15.2. CONFIGURE THE HTTP SESSION TIMEOUT

The HTTP session timeout defines the period after which a HTTP session is considered to have become invalid because there was no activity within the specified period. Changing the HTTP session timeout requires that all affected JBoss EAP instances be restarted. Until that is done, the original HTTP session timeout value applies.

The HTTP session timeout can be configured in several places. In order of precedence these are:

- Application - defined in the application's **web.xml** configuration file. For details see *Configure the HTTP Timeout per Application* in the *Development Guide*
- Server - specified via the **default-session-timeout** attribute. This setting is only available from JBoss EAP 6.4.
- Default - 30 minutes.

Procedure 15.1. Configure the HTTP Session Timeout using the Management Console

1. Click the **Configuration** tab, then navigate to **Subsystems, Web**, and click on the **Servlet/HTTP** menu item.
2. Click the **Global** tab in the **Servlet/HTTP Configuration** panel.
3. Click the **Edit** option.
4. Enter the new value for the **Default session timeout**.
5. Click the **Save** button.
6. Reload the JBoss EAP server.

Procedure 15.2. Configure the HTTP Session Timeout using the Management CLI



NOTE

Add the prefix **/host=HOST_NAME** to the command for a managed domain.

1. Specify the desired HTTP Session Timeout value.

```
/subsystem=web:write-attribute(name=default-session-timeout, value=timeout)
```

2. Reload the JBoss EAP server.

```
reload
```

[Report a bug](#)

15.3. SERVLET/HTTP CONFIGURATION

Procedure 15.3. Servlet/HTTP Configuration

1. In the Management Console, click the **Configuration** tab at the top of the screen, expand the **Subsystems** menu and then expand the **Web** menu.
2. Click on the **Servlet/HTTP** menu item.
3. Click on the name of the component, then click on **Edit**.
4. Click the **Advanced** button to view advanced options.

Following are the advanced Servlet/HTTP configuration options.

If the Management CLI commands are to be applied to a profile, add the prefix **/profile=PROFILE**.

Example 15.1. Configure the Name of this Instance

```
[domain@localhost:9999 /] /profile=full-ha/subsystem=web:write-attribute(name=instance-id,value=worker1)
```

Global Configuration Options

Default Session Timeout

Available from JBoss EAP 6.4. The web container's default session timeout.

Management CLI attribute: **default-session-timeout**

Default Virtual Server

The web container's default virtual server.

Management CLI attribute: **default-virtual-server**

Instance ID

The identifier used to enable session affinity in load balancing scenarios. The identifier must be unique across all JBoss EAP servers in the cluster and is appended to the generated session identifier. This allows the front-end proxy to forward the specific session to the same JBoss EAP instance. The instance ID is not set as a default.

Management CLI attribute: **instance-id**

Native

Add the native initialization listener to the web container.

Management CLI attribute: **native**. Values: **true** or **false**. Default: **true**.

JSP Configuration Options

Check Interval

Interval at which to check for JSP updates using a background thread, specified in seconds. Default: **0**, which means **disabled**.

Management CLI attribute: **check-interval**

Development

If **true**, enables Development Mode, which produces more verbose debugging information. Default: **false**.

Management CLI attribute: **development**

Disabled

If **true**, the Java ServerPages (JSP) container is disabled. This is useful if you do not use any JSPs. Default: **false**.

Management CLI attribute: **disabled**

Display Source Fragment

If **true**, the JSP source fragment is displayed when a runtime error occurs. Default: **true**.

Management CLI attribute: **display-source-fragment**

Dump SMAP

If **true**, JSR 045 SMAP data is written to a file. Default: **false**

Management CLI attribute: **dump-smap**

Generate Strings as Char Arrays

If **true**, string constants are generated as **char** arrays. Default: **false**

Management CLI attribute: **generate-strings-as-char-arrays**

Error on Use Bean Invalid Class Attribute

If **true**, enables the output of errors when a bad class is used in useBean. Default: **false**

Management CLI attribute: **error-on-use-bean-invalid-class-attribute**

Java Encoding

Specifies the encoding used for Java sources. Default: **UTF8**

Management CLI attribute: **java-encoding**

Keep Generated

If **true**, keeps generated Servlets. Default: **true**.

Management CLI attribute: **keep-generated**

Mapped File

If **true**, static content is generated with one print statement per input line, to ease debugging. Default: **true**.

Management CLI attribute: **true**

Modification Test Interval

Minimum amount of time between two tests for updates, specified in seconds. Default: **4**

Management CLI attribute: **modification-test-interval**

Recompile on Fail

If **true**, failed JSP compilations will be retried on each request. Default: **false**.

Management CLI attribute: **recompile-on-fail**

Scratch Directory

Specifies the location of a different work directory.

Management CLI attribute: **scratch-dir**

SMAP

If **true**, JSR 045 SMAP is enabled. Default: **true**

Management CLI attribute: **smap**

Source VM

Java development kit version with which the source files are compatible. Default: **1.5**

Management CLI attribute: **source-vm**

Tag Pooling

If **true**, tag pooling is enabled. Default: **true**

Management CLI attribute: **tag-pooling**

Target VM

Java development kit version with which the class files are compatible. Default: **1.5**

Management CLI attribute: **target-vm**

Trim Spaces

Trim some spaces from the generated Servlet. Default: **false**.

Management CLI attribute: **trim-spaces**

X Powered By

If **true**, the JSP engine is advertised using the **x-powered-by** HTTP header. Default: **true**.

Management CLI attribute: **x-powered-by**

Use one of the following commands to view the settings for either of these connectors:

```
[standalone@localhost:9999 /] /subsystem=web/connector=http:read-resource-description
```

```
[standalone@localhost:9999 /] /subsystem=web/connector=ajp:read-resource-description
```

To configure a connector, select the **Connectors** tab and click **Add**. To remove a connector, select its entry and click **Remove**. To edit a connector, select its entry and click **Edit**.

When you create a new connector using the Management CLI, its options are all set at once, as in the following command:

Example 15.2. Create a New Connector

```
[domain@localhost:9999 /] /profile=full-ha/subsystem=web/connector=ajp:add(socket-binding=ajp,scheme=http,protocol=AJP/1.3,secure=false,name=ajp,max-post-size=2097152,enabled=true,enable-lookups=false,redirect-port=8433,max-save-post-size=4096)
```

Default Connector attributes

Bytes Sent

The number of byte sent by the connector.

Management CLI attribute: **bytesSent**

Proxy Port

The port that will be used when sending a redirect.

Management CLI attribute: **proxy-port**

Secure

Indicates if content sent or received by the connector is secured from the user perspective. Default: **false**.

Management CLI attribute: **secure**

Virtual Server

The list of virtual servers that can be accessed through this connector. Default: Allow all virtual servers.

Management CLI attribute: **virtual-server**

Error Count

Number of errors that occur when processing requests by the connector.

Management CLI attribute: **errorCount**

Maximum Time

Maximum time spent to process a request.

Management CLI attribute: **maxTime**

Socket Binding

The named socket binding to which the connector is to be bound. A socket binding is a mapping between a socket name and a network port. Socket bindings are configured for each standalone server, or via socket binding groups in a managed domain. A socket binding group is applied to a server group.

Management CLI attribute: **socket-binding**

Scheme

The web connector scheme (such as HTTP or HTTPS).

Management CLI attribute: **scheme**

Name

A unique name for the connector.

Management CLI attribute: **name**

Maximum Post Size

Maximum size in bytes of a POST request that can be parsed by the container. Default: **2097152**

Management CLI attribute: **max-post-size**

Request Count

Number of the request processed by the connector.

Management CLI attribute: **requestCount**

Proxy Name

The host name that will be used when sending a redirect. Default: **null**.

Management CLI attribute: **proxy-name**

Enabled

Defines whether the connector is started on startup. Default: **true**

Management CLI attribute: **enabled**

Protocol

The web connector protocol to use, either AJP or HTTP. For each protocol you can either specify the API and leave it to the server to determine which implementation is used, or specify the fully-qualified class name.

Management CLI attribute: **protocol**

HTTP

API options: **HTTP/1.1** or **HTTP/1.0**.

If the client does not support HTTP/1.1, the connector will return HTTP/1.1 in its initial response, then fall back to HTTP/1.0.

Fully Qualified Connector Name options:

- JIO: **org.apache.coyote.http11.Http11Protocol**
- NIO2: **org.apache.coyote.http11.Http11NioProtocol**
- APR: **org.apache.coyote.http11.Http11AprProtocol**

AJP

API option: **AJP/1.3**

Fully Qualified Connector Name options:

- JIO: **org.apache.coyote.ajp.AjpProtocol**
- APR: **org.apache.coyote.ajp.AjpAprProtocol**



NOTE

APR requires the Native Components package be installed and active. For detailed instructions on how to do so, see the *JBoss EAP Installation Guide*.

Enable Lookups

Enable DNS lookups for the Servlet API.

Management CLI attribute: **enable-lookups**

Executor

The name of the executor that should be used for the processing threads of this connector. If undefined defaults to using an internal pool.

Management CLI attribute: **executor**

Processing Time

Processing time used by the connector, measured in milliseconds.

Management CLI attribute: **processingTime**

Proxy Binding

The socket binding to define the host and port that will be used when sending a redirect. Default: **null**.

Management CLI attribute: **proxy-binding**

Redirect Port

The port for redirection to a secure connector. Default: **443**

Defers to **redirect-binding** when set concurrently.

Management CLI attribute: **redirect-port**

Bytes Received

Number of bytes received by the connector (POST data).

Management CLI attribute: **bytesReceived**

Redirect Binding

Redirect binding is similar to redirect port in terms of behavior except that it requires specification of a socket-binding name in *value* instead of a port number. The **redirect-binding** option provides higher configuration flexibility because it allows the use of pre-defined socket binding (https, AJP etc.) to the specific port for redirection. It gives the same results as **redirect-port** option.

Takes precedence over **redirect-port** when set concurrently.

Management CLI attribute: **redirect-binding**

Maximum Connections

The maximum number of concurrent connections that can be processed by the connector with optimum performance. The default value depends on the connector used.

Management CLI attribute: **max-connections**

Maximum Save Post Size

Maximum size in bytes of a POST request that will be saved during certain authentication schemes. Default: **4096**.

Management CLI attribute: **max-save-post-size**

To configure virtual servers, click the **Virtual Servers** tab. Use the **Add** button to add a new virtual server. To edit or remove a virtual server, select its entry and click the **Edit** or **Remove** button.

When you add a new virtual server using the Management CLI, all required options are set at once, as in the following command.

Example 15.3. Add a New Virtual Server

```
/profile=full-ha/subsystem=web/virtual-server=default-host/:add(enable-welcome-root=true,default-web-module=ROOT.war,alias=["localhost","example.com"],name=default-host)
```


Virtual Servers Options

Access Log

The element describing how the access log information must be logged.

Management CLI attribute: **access-log**

- To add **access-log** attribute enter the following management CLI command:

```
/subsystem=web/virtual-server=default-host/configuration=access-log:add()
```

- The **access-log** attribute has several child attributes. To list these children and their configuration options, enter the following management CLI command:

```
/subsystem=web/virtual-server=default-host/configuration=access-log:read-resource-description()
```

pattern

A formatting layout identifying the various information fields from the request and response to be logged, or the word **common** or **combined** to select a standard format. Values for the **pattern** attribute are made up of format tokens. If not specified, a default of **common** is used.

rotate

Tell the valve if it should rotate the output or not. If not specified, a default of **true** is used.

prefix

Define the prefix to be used to name the log file. If not specified, a default of **access_log** is used.

extended

Uses the **ExtendedAccessLogValve** instead the **AccessLogValve**. If not specified, a default of **false** is used.

resolve-hosts

Tell the valve whether to resolve the host names or not. If not specified, a default of **false** is used. Unless the **lookups** on the connector for **resolve-host** is enabled, this option will not work. This can be enabled by setting **resolve-host** to **true**.

Alias

A list of hostnames supported by this virtual server. In the Management Console, use one hostname per line.

Management CLI attribute: **alias**

Default Web Module

The module whose web application will be deployed at the root node of this virtual server, and will be displayed when no directory is given in the HTTP request. Default: **ROOT.war**

Management CLI attribute: **default-web-module**

Enable Welcome Root

This element defines whether or not the bundled welcome directory is used as the root web context. Default: **false**

Management CLI attribute: **enable-welcome-root**

Name

A unique name for the virtual server, for display purposes.

Management CLI attribute: **name**

Rewrite

The element describing what the rewrite valve must do with requests corresponding to the virtual host. **rewrite** describes how requests would be rewritten before processing. It adds the **RewriteValve** to the Virtual Host defined by **virtual-server**.

flag

A **RewriteRule** can have its behavior modified by one or more flags. Flags are included in square brackets at the end of the rule, and multiple flags are separated by commas.

pattern

Pattern is a perl compatible regular expression, which is applied to the URL of the request.

substitution

The substitution of a rewrite rule is the string which is substituted for (or replaces) the original URL which Pattern matched.

Management CLI attribute: **rewrite**

SSO

SSO configuration for this virtual server. Default (in case of http): **true**.

Management CLI attribute: **sso**

- To add **SSO** configuration, enter the following management CLI command:

```
/subsystem=web/virtual-server=default-host/configuration=sso:add()
```

- The **SSO** attribute has several child attributes. To list these children and their configuration options, enter the following management CLI command:

```
/subsystem=web/virtual-server=default-host/configuration=sso:read-resource-description()
```

http-only

The cookie **http-only** flag will be set.

cache-container

Enables clustered **SSO** using the specified clustered cache container.

reauthenticate

Enables reauthentication with the realm when using **SSO**.

domain

The cookie domain that will be used.

cache-name

Name of the cache to use in the cache container.

[Report a bug](#)

15.4. REPLACE THE DEFAULT WELCOME WEB APPLICATION

JBoss EAP 6 includes a Welcome application, which displays when you open the URL of the server at port 8080. You can replace this application with your own web application by following this procedure.

Procedure 15.4. Replace the Default Welcome Web Application With Your Own Web Application

1. **Disable the Welcome application.**

Use the Management CLI script **EAP_HOME/bin/jboss-cli.sh** to run the following command. You may need to change the profile to modify a different managed domain profile, or remove the **/profile=default** portion of the command for a standalone server.

```
/profile=default/subsystem=web/virtual-server=default-host:write-attribute(name=enable-welcome-root,value=false)
```

2. **Configure your Web application to use the root context.**

To configure your web application to use the root context (/) as its URL address, modify its **jboss-web.xml**, which is located in the **META-INF/** or **WEB-INF/** directory. Replace its **<context-root>** directive with one that looks like the following.

```
<jboss-web>
  <context-root>/</context-root>
</jboss-web>
```

3. **Deploy your application.**

Deploy your application to the server group or server you modified in the first step. The application is now available on **http://SERVER_URL:PORT/**.

[Report a bug](#)

15.5. SYSTEM PROPERTIES IN JBOSSWEB

This section lists the system properties that may be used to modify the default JBossWeb behavior. The system-properties can be set in the JBoss Enterprise Web Application configuration. You must restart it to get them applied to the web sub system.

Following is an example on how to modify the system-properties in JBossWeb

```
standalone@localhost:9999 [/] ./system-
property=org.apache.catalina.JSESSIONID:add(value="MYID")
{"outcome" => "success"}
standalone@localhost:9999 [/] shutdown
Communication error: Channel closed
Closed connection to localhost:9999
```

For some properties, you can restart it using a reload command.

Following is an example to restart using a reload command.

```
[standalone@localhost:9999 /] reload
{
  "outcome" => "success",
  "response-headers" => {
    "operation-requires-reload" => true,
    "process-state" => "reload-required"
  }
}
```

Table 15.1. Servlet container and connectors

Attribute	Description
jvmRoute	Provides a default value for the <i>jvmRoute</i> attribute. It does not override the automatically generated value used when using ha read with using configuration like standalone-ha.xml It supports reload .
org.apache.tomcat.util.buf.StringCache.byte.enabled	If true, the String cache is enabled for ByteChunk. If the value is not specified, the default value of false is used.
org.apache.tomcat.util.buf.StringCache.char.enabled	If true, the String cache is enabled for CharChunk. If the value is not specified, the default value of false is used.
org.apache.tomcat.util.buf.StringCache.cacheSize	The size of the String cache. If the value is not specified, the default value of 5000 is used.
org.apache.tomcat.util.buf.StringCache.maxStringSize	The maximum length of String that will be cached. If the value is not specified, the default value of 128 is used.
org.apache.tomcat.util.http.FastDateFormat.CACHE_SIZE	The size of the cache to use parsed and formatted date value. If the value is not specified, the default value of 1000 is used.

Attribute	Description
org.apache.catalina.core.StandardService.DELAY_CONNECTOR_STARTUP	If true, the connector startup is not done automatically. It is useful in embedded mode.
org.apache.catalina.connector.Request.SESSION_ID_CHECK	If true, the Servlet container verifies that a session exists in a context with the specified session id before creating a session with that id.
org.apache.coyote.USE_CUSTOM_STATUS_MSG_IN_HEADER	If true, custom HTTP status messages are used within HTTP headers. Users must ensure that any such message is ISO-8859-1 encoded, particularly if user provided input is included in the message, to prevent a possible XSS vulnerability. If value is not specified the default value of false is used.
org.apache.tomcat.util.http.Parameters.MAX_COUNT	The maximum amount of parameters that can be parsed in a post body. If exceeded, parsing fails using an <code>IllegalStateException</code> . The default value is 512 parameters.
org.apache.tomcat.util.http.MimeHeaders.MAX_COUNT	The maximum amount of headers that can be sent in the HTTP request. If exceeded, parsing will fail using an <code>IllegalStateException</code> . The default value is 128 headers.
org.apache.tomcat.util.net.MAX_THREADS	The maximum number of threads a connector is going to use to process requests. The default value is $32 \times \text{Runtime.getRuntime().availableProcessors()}$. ($512 \times \text{Runtime.getRuntime().availableProcessors()}$ for the JIO connector)
org.apache.coyote.http11.Http11Protocol.MAX_HEADER_SIZE	The maximum size of the HTTP headers, in bytes. If exceeded, parsing will fail using an <code>ArrayOutOfBoundsException</code> . The default value is 8192 bytes.
org.apache.coyote.http11.Http11Protocol.COMPRESSION	Allows using simple compression with the HTTP connector. The default value is off, and compression can be enabled using the value on to enable it conditionally, or force to always enable it.
org.apache.coyote.http11.Http11Protocol.COMPRESSION_RESTRICTED_UA	User agents regexps that will not receive compressed content. The default value is empty.
org.apache.coyote.http11.Http11Protocol.COMPRESSION_MIME_TYPES	Content type prefixes of compressible content. The default value is <code>text/html,text/xml,text/plain</code> .
org.apache.coyote.http11.Http11Protocol.COMPRESSION_MIN_SIZE	Minimum size of content that will be compressed. The default value is 2048 bytes.

Attribute	Description
org.apache.coyote.http11.DEFAULT_CONNECTION_TIMEOUT	Default socket timeout. The default value is 60000 ms.
org.jboss.as.web.deployment.DELETE_WORK_DIR_ONCONTEXTDESTROY	Use this property to remove .java and .class files to ensure that JSP sources are recompiled. The default value is false . Default socket timeout for keep alive. The default value is -1 ms, which means it will use the default socket timeout.
org.apache.tomcat.util.buf.StringCache.trainThreshold	Specifies the number of times toString() must be invoked before activating cache. The default value is 100000 .

Table 15.2. EL

Attribute	Description
org.apache.el.parser.COERCE_TO_ZERO	If true, when coercing expressions to numbers "" and null will be coerced to zero as required by the specification. If value is not specified, the default value of true is used.

Table 15.3. JSP

Attribute	Description
org.apache.jasper.compiler.Generator.VAR_EXPRESSIONFACTORY	The name of the variable to use for the expression language expression factory. If value is not specified, the default value of <code>_el_expressionfactory</code> is used.
org.apache.jasper.compiler.Generator.VAR_INSTANCEMANAGER	The name of the variable to use for the instance manager factory. If value is not specified, the default value of <code>_jsp_instancemanager</code> is used.
org.apache.jasper.compiler.Parser.STRICT_QUOTE_ESCAPING	If false, the requirements for escaping quotes in JSP attributes are relaxed so that a missing required quote does not cause an error. If value is not specified, the specification compliant default of true is used.
org.apache.jasper.Constants.DEFAULT_TAG_BUFFER_SIZE	Any tag buffer that expands beyond <code>org.apache.jasper.Constants.DEFAULT_TAG_BUFFER_SIZE</code> is destroyed and a new buffer is created of the default size. If value is not specified, the default value of 512 is used.

Attribute	Description
org.apache.jasper.runtime.JspFactoryImpl.USE_POOL	If true, a ThreadLocal PageContext pool is used. If value is not specified, the default value of true is used.
org.apache.jasper.runtime.JspFactoryImpl.POOL_SIZE	The size of the ThreadLocal PageContext. If value is not specified, the default value of 8 is used.
org.apache.jasper.Constants.JSP_SERVLET_BASE	The base class of the Servlets generated from the JSPs. If value is not specified, the default value of org.apache.jasper.runtime.HttpJspBase is used.
org.apache.jasper.Constants.SERVICE_METHOD_NAME	The name of the service method called by the base class. If value is not specified, the default value of _jspService is used.
org.apache.jasper.Constants.SERVLET_CLASSPATH	The name of the ServletContext attribute that provides the classpath for the JSP. If value is not specified, the default value of org.apache.catalina.jsp_classpath is used.
org.apache.jasper.Constants.JSP_FILE	The name of the request attribute for <jsp-file> element of a servlet definition. If present on a request, this overrides the value returned by request.getServletPath() to select the JSP page to be executed. If value is not specified, the default value of org.apache.catalina.jsp_file is used.
org.apache.jasper.Constants.PRECOMPILE	The name of the query parameter that causes the JSP engine to just pregenerate the servlet but not invoke it. If value is not specified, the default value of org.apache.catalina.jsp_precompile is used.
org.apache.jasper.Constants.JSP_PACKAGE_NAME	The default package name for compiled jsp pages. If value not specified, the default value of org.apache.jsp is used.
org.apache.jasper.Constants.TAG_FILE_PACKAGE_NAME	The default package name for tag handlers generated from tag files. If value is not specified, the default value of org.apache.jsp.tag is used.
org.apache.jasper.Constants.TEMP_VARIABLE_NAME_PREFIX	Prefix to use for generated temporary variable names. If value is not specified, the default value of _jspx_temp is used.
org.apache.jasper.Constants.USE_INSTANCE_MANAGER_FOR_TAGS	If true, the instance manager is used to obtain tag handler instances. If value is not specified, true is used.

Attribute	Description
org.apache.jasper.Constants.INJECT_TAGS	If true, annotations specified in tags will be processed and injected. This can have a performance impact when using simple tags, or if tag pooling is disabled. If value is not specified, false is used.

Table 15.4. Security

Attribute	Description
org.apache.catalina.connector.RECYCLE_FACADES	If this is true or if a security manager is in use a new facade object is created for each request. If value is not specified, the default value of false is used.
org.apache.catalina.connector.CoyoteAdapter.ALLOW_BACKSLASH	If this is true the '\' character is permitted as a path delimiter. If value is not specified, the default value of false is used.
org.apache.tomcat.util.buf.UDecoder.ALLOW_ENCODED_SLASH	If this is true '%2F' and '%5C' is permitted as path delimiters. If value is not specified, the default value of false is used.

Table 15.5. Specification

Attribute	Description
org.apache.catalina.STRICT_SERVLET_COMPLIANCE	<p>If value is not specified, true is used. If this is true the following actions will occur:</p> <ul style="list-style-type: none"> any wrapped request or response object passed to an application dispatcher is checked to ensure that it has wrapped the original request or response. (SRV.8.2 / SRV.14.2.5.1) a call to <code>Response.getWriter()</code> if no character encoding has been specified results in subsequent calls to <code>Response.getCharacterEncoding()</code> returning ISO-8859-1 and the Content-Type response header will include a <code>charset=ISO-8859-1</code> component. (SRV.15.2.22.1) every request that is associated with a session causes the session's last accessed time to be updated regardless of whether or not the request explicitly accesses the session. (SRV.7.6)

Attribute	Description
org.apache.catalina.core.StandardWrapperValve.SERVLET_STATS	If true or if org.apache.catalina.STRICT_SERVLET_COMPLIANCE is true, the wrapper will collect the JSR-77 statistics for individual servlets. If value is not specified, the default value of false is used.
org.apache.catalina.session.StandardSession.ACTIVITY_CHECK	If this is true or if org.apache.catalina.STRICT_SERVLET_COMPLIANCE is true Tomcat tracks the number of active requests for each session. When determining if a session is valid, any session with at least one active request is always be considered valid. If value is not specified, the default value of false is used.

[Report a bug](#)

15.6. ABOUT HTTP-ONLY SESSION MANAGEMENT COOKIES

The **http-only** attribute for session management cookies mitigates the risk of security vulnerabilities by restricting access from non-HTTP APIs (such as JavaScript). This restriction helps mitigate the threat of session cookie theft via cross-site scripting attacks. On the client side, the cookies cannot be accessed using JavaScript or other scripting methods. This applies only to session management cookies and not other browser cookies. By default, the **http-only** attribute is enabled.

If it has not yet been done, you need to add SSO to the virtual server in the web subsystem to use the **http-only** attribute.

Example 15.4. Add SSO to the Virtual Server

Enter the following Management CLI command to add SSO to the virtual server in the web subsystem.

```
/subsystem=web/virtual-server=default-host/configuration=sso:add
```



NOTE

If this command results in a "JBAS014803: Duplicate resource" failure, it means SSO is already added to the virtual server configuration. You can ignore this error and continue.



NOTE

JSESSIONID and JSESSIONIDSSO are session tracking cookies. By default, they are **http-only** and must not be accessed by scripts.

Example 15.5. Verify the http-only Attribute

Enter the following Management CLI command to verify the value of the **http-only** attribute.

■

```
/subsystem=web/virtual-server=default-host/configuration=sso:read-resource
{
  "outcome" => "success",
  "result" => {
    "cache-container" => undefined,
    "cache-name" => undefined,
    "domain" => undefined,
    "http-only" => true,
    "reauthenticate" => undefined
  },
  "response-headers" => {"process-state" => "reload-required"}
}
```

Example 15.6. Enable the **http-only** Attribute

Enter the following Management CLI command to enable the **http-only** attribute.

```
/subsystem=web/virtual-server=default-host/configuration=sso:write-attribute(name=http-
only,value=true)
```

[Report a bug](#)

CHAPTER 16. WEB SERVICES SUBSYSTEM

16.1. CONFIGURE WEB SERVICES OPTIONS

The Web Services options can be configured using the Management Console or the Management CLI.

In the Management Console, click the **Configuration** tab at the top of the screen, expand the **Subsystems** menu and then expand the **Web** menu. Click on the **Web Services** menu item to configure the web services subsystem. The options are explained in the table below.

Table 16.1. Web Services Configuration Options

Option	Description	CLI Command
Modify WSDL Address	Whether the WSDL address can be modified by applications. Defaults to true .	<pre>/profile=full- ha/subsystem=webservices/: write- attribute(name=modify- wsdl-address,value=true)</pre>
WSDL Host	The WSDL contract of a JAX-WS Web Service includes a <code><soap:address></code> element which points to the location of the endpoint. If the value of <code><soap:address></code> is a valid URL, it is not overwritten unless modify-wsdl-address is set to true . If the value of <code><soap:address></code> is not a valid URL, it is overwritten using the values of wsdl-host and either wsdl-port or wsdl-secure-port . If wsdl-host is set to jbossws.undefined.host , the requester's host address is used when the <code><soap-address></code> is rewritten. Defaults to <code>}\${jboss.bind.address:127.0.0.1}</code> , which uses 127.0.0.1 if no bind address is specified when JBoss EAP 6 is started.	<pre>/profile=full- ha/subsystem=webservices/: write-attribute(name=wsdl- host,value=127.0.0.1)</pre>
WSDL Port	The non-secure port that is used to rewrite the SOAP address. If this is not set (the default), the port is identified by querying the list of installed connectors.	<pre>/profile=full- ha/subsystem=webservices/: write-attribute(name=wsdl- port,value=80)</pre>
WSDL Secure Port	The secure port that is used to rewrite the SOAP address. If this is not set (the default), the port is identified by querying the list of installed connectors.	<pre>/profile=full- ha/subsystem=webservices/: write-attribute(name=wsdl- secure-port,value=443)</pre>

**NOTE**

You may need to change the profile to modify a different managed domain profile, or remove the `/profile=full-ha` part of the command for a standalone server.

Web Services Subsystem

To enable logging with Apache CXF, configure the following system property in **standalone/domain.xml** file:

```
<system-properties>
<property name="org.apache.cxf.logging.enabled" value="true"/>
</system-properties>
```

[Report a bug](#)

16.2. OVERVIEW OF HANDLERS AND HANDLER CHAINS

Each endpoint config may be associated with **PRE** and **POST** handler chains. Each handler chain may include JAXWS-compliant handlers. For outbound messages, PRE handler chain handlers are executed before any handler attached to the endpoints using standard JAXWS means, such as the **@HandlerChain** annotation. POST handler chain handlers are executed after usual endpoint handlers. For inbound messages, the opposite applies. JAX-WS is a standard API for XML-based web services, and is documented at <http://jcp.org/en/jsr/detail?id=224>.

A handler chain may also include a **protocol-bindings** attribute, which sets the protocols which trigger the chain to start.

A JAXWS handler is a child element **handler** within a handler chain. The handler takes a **class** attribute, which is the fully-qualified classname of the handler class. When the endpoint is deployed, an instance of that class is created for each referencing deployment. Either the deployment class loader or the class loader for module **org.jboss.as.webservices.server.integration** must be able to load the handler class.

Procedure 16.1. How to add handler-chains and handlers via the CLI

1. Start the JBoss EAP CLI

```
EAP_HOME/bin/jboss-cli.sh
```

2. Add handler chain and handlers via JBoss CLI:

Example 16.1. Add a handler chain

```
[standalone@localhost:9999 /] /subsystem=webservices/endpoint-config=Standard-Endpoint-Config/post-handler-chain=my-handlers:add(protocol-bindings="##SOAP11_HTTP")
```

Example 16.2. Add a handler

```
[standalone@localhost:9999 /] /subsystem=webservices/endpoint-config=Standard-Endpoint-Config/post-handler-chain=my-handlers/handler=foo-handler:add(class="org.jboss.ws.common.invocation.RecordingServerHandler")
```

Example 16.3. Add a handler

```
[standalone@localhost:9999 /] /subsystem=webservices/endpoint-config=Standard-Endpoint-Config/post-handler-chain=my-handlers/handler=bar-handler:add(class="com.arjuna.webservices11.wsarj.handler.InstanceIdentifierInHandler")
```

3. Reload the server:

```
[standalone@localhost:9999 /] reload
```

4. Confirm the handler-chain and handlers were added correctly:

Example 16.4. Read a handler-chain

```
[standalone@localhost:9999 /] /profile=default/subsystem=webservices/endpoint-config=Standard-Endpoint-Config/post-handler-chain=my-handlers:read-resource
```

Example 16.5. Read a handler

```
[standalone@localhost:9999 /] /profile=default/subsystem=webservices/endpoint-config=Standard-Endpoint-Config/post-handler-chain=my-handlers/handler=bar-handler:read-resource
```

The options used in the commands above can be modified as required to add or modify handlers.

The handlers available in JBoss EAP can be found in the *API Documentation* javadocs:

https://access.redhat.com/documentation/en-US/JBoss_Enterprise_Application_Platform/6.4/html/API_Documentation/files/javadoc/javax/xml/ws/ha

More information about handlers, handler-chains, endpoints and related issues can be also found in the *JBoss EAP Development Guide* available at https://access.redhat.com/documentation/en-us/red_hat_jboss_enterprise_application_platform/?version=6.4

[Report a bug](#)

CHAPTER 17. HTTP CLUSTERING AND LOAD BALANCING

17.1. HTTP SERVER NAME CONVENTIONS

The following tables outline the naming and location conventions used when discussing HTTPD-related topics.

Table 17.1. Apache HTTP Server provided by operating system

Operating System	<i>HTTPD_CONF.D</i>	<i>HTTPD_CONF</i>	<i>HTTPD_MODULES</i>
Red Hat Enterprise Linux (httpd)	/etc/httpd/conf.d	/etc/httpd/conf	/etc/httpd/modules
HPUX (Web Server Suite)	See note below.	/opt/hpws/apache/conf	/opt/hpws/apache/modules



NOTE

There is no **conf.d** in HP-UX's Web server Suite for Apache HTTP Server. You can create one, however:

Procedure 17.1. title

1. Create **/opt/hpws/apache/conf.d**.
2. Add **Include conf.d/*.conf** into the **httpd.conf** file.

Table 17.2. JBoss Enterprise Web Server

Operating System	<i>HTTPD_CONF.D</i>	<i>HTTPD_CONF</i>	<i>HTTPD_MODULE S</i>	Distribution
Red Hat Enterprise Linux	/EWS_HOME/httpd/conf.d	/EWS_HOME/httpd/conf	/EWS_HOME/httpd/modules	zip
Red Hat Enterprise Linux	/etc/httpd/conf.d	/etc/httpd/conf	/usr/lib/httpd/modules	rpm
Solaris	/EWS_HOME/etc/httpd/conf.d	/EWS_HOME/etc/httpd/conf	/EWS_HOME/lib/httpd/modules	zip
Windows	/EWS_HOME/etc/httpd/conf.d	/EWS_HOME/etc/httpd/conf	/EWS_HOME/lib/httpd/modules	zip

**NOTE****Path variances:**

- If you are using a 64-bit architecture, amend the filepaths from the table above to use the **lib64/** directory.
- If you are using a Red Hat Enterprise Linux 7 installation, the **httpd** directory is named **httpd22**.

[Report a bug](#)

17.2. INTRODUCTION

17.2.1. About High-Availability and Load Balancing Clusters

Clustering refers to using multiple resources, such as servers, as though they were a single entity. The two main types of clustering are *Load balancing (LB)* and *High-availability (HA)*. In a LB cluster, all resources run at the same time, and a management layer spreads the work load across them.

In HA clustering, one resource runs, and another is available to step in if the first one becomes unavailable. The purpose of HA clustering is to reduce the consequences of hardware, software, or network outages.

JBoss EAP 6 supports clustering at several different levels. Some of the components of the runtime and your applications that can be made highly available are:

- Instances of the Application Server
- Web applications, when used in conjunction with the internal JBoss Web Server, Apache HTTP Server, Microsoft IIS, or Oracle iPlanet Web Server.
- Stateful, stateless, and entity Enterprise JavaBeans (EJBs)
- Single Sign On (SSO) Mechanisms
- Distributed cache
- HTTP sessions
- JMS services and Message-driven beans (MDBs)

Clustering is made available to JBoss EAP 6 by two subsystems: **jgroups** and **modcluster**. The **ha** and **full-ha** profiles have these systems enabled. In JBoss EAP 6 these services start up and shut down on demand, but they will only start up if an application configured as **distributable** is deployed on the servers.

Infinispan is provided as the cache provider in JBoss EAP 6. Infinispan manages clustering and replication caches for JBoss EAP 6.

[Report a bug](#)

17.2.2. Components Which Can Benefit from High Availability

High Availability (HA) falls into a few broad categories in JBoss EAP 6.

The Container

Several instances of JBoss EAP 6 (running as a standalone server) or a server group's members (running as part of a managed domain) can be configured to be highly available. This means that if one instance or member is stopped or disappears from the cluster, its work load is moved to a peer. The work load can be managed in such a way to provide load-balancing functionality as well, so that servers or server groups with more or better resources can take on a larger portion of the work load, or additional capacity can be added during times of high load.

The Web Server

The web server itself can be clustered for HA, using one of several compatible load balancing mechanisms. The most flexible is **mod_cluster** connector, which is tightly integrated with the JBoss EAP 6 container. Other choices include Apache **mod_jk** or **mod_proxy** connectors, or the ISAPI connector and NSAPI connector.

The Application

Deployed applications can be made highly-available because of the Java Enterprise Edition 6 (Java EE 6) specification. Stateless or stateful session EJBs can be clustered so that if the node which is involved in the work disappears, another node can take over, and in the case of stateful session beans, preserve the state.

[Report a bug](#)

17.2.3. Overview of HTTP Connectors

JBoss EAP 6 has the ability to use load-balancing and high-availability mechanisms built into external web servers, such as Apache Web Server, Microsoft IIS, and Oracle iPlanet. JBoss EAP 6 communicates with the external web server using a connector. These connectors are configured within the web subsystem of JBoss EAP 6.

The web servers include software modules which control the way that HTTP requests are routed to JBoss EAP 6 nodes. Each of these modules varies in how it works and how it is configured. The modules are configured to balance work loads across multiple JBoss EAP 6 nodes, to move work loads to alternate servers in case of a failure event, or both. These abilities are called *Load Balancing* and *High Availability (HA)*.

JBoss EAP 6 supports several different connectors. The one you choose depends on the web server in use and the functionality you need.

The table below lists the differences between the different HTTP connectors which are compatible with JBoss EAP 6. For the most current information about supported configurations for HTTP connectors, see <https://access.redhat.com/articles/111663>.

Table 17.3. HTTP connector features and constraints

Connector	Web server	Supported operating systems	Supported protocols	Adapts to deployment status	Supports sticky session

Connector	Web server	Supported operating systems	Supported protocols	Adapts to deployment status	Supports sticky session
mod_cluster	httpd in JBoss Enterprise Web Server, httpd provided by operating system (Red Hat Enterprise Linux, Hewlett-Packard HP-UX)	Red Hat Enterprise Linux, Microsoft Windows Server, Oracle Solaris, Hewlett-Packard HP-UX	HTTP, HTTPS, AJP	Yes. Detects deployment and undeployment of applications and dynamically decides whether to direct client requests to a server based on whether the application is deployed on that server.	Yes
mod_jk	httpd in JBoss Enterprise Web Server, httpd provided by operating system (Red Hat Enterprise Linux, Hewlett-Packard HP-UX)	Red Hat Enterprise Linux, Microsoft Windows Server, Oracle Solaris, Hewlett-Packard HP-UX	AJP	No. Directs client requests to the container as long as the container is available, regardless of application status.	Yes
mod_proxy	httpd in JBoss Enterprise Web Server	Red Hat Enterprise Linux, Microsoft Windows Server, Oracle Solaris	HTTP, HTTPS, AJP	No. Directs client requests to the container as long as the container is available, regardless of application status.	Yes
ISAPI connector	Microsoft IIS	Microsoft Windows Server	AJP	No. Directs client requests to the container as long as the container is available, regardless of application status.	Yes

Connector	Web server	Supported operating systems	Supported protocols	Adapts to deployment status	Supports sticky session
NSAPI connector	Oracle iPlanet Web Server	Oracle Solaris	AJP	No. Directs client requests to the container as long as the container is available, regardless of application status.	Yes

Learn more about each HTTP Connector

- [Section 17.6.1, "About the `mod_cluster` HTTP Connector"](#)
- [Section 17.7.1, "About the Apache `mod_jk` HTTP Connector"](#)
- [Section 17.8.1, "About the Apache `mod_proxy` HTTP Connector"](#)
- [Section 17.9.1, "About the Internet Server API \(ISAPI\)"](#)
- [Section 17.10.1, "About the Netscape Server API \(NSAPI\)"](#)

JBoss EAP 6 supported configurations are available here: <https://access.redhat.com/articles/111663>.

[Report a bug](#)

17.2.4. Node types

HTTP connector node

A *worker node*, sometimes referred to as a node, is a JBoss EAP 6 server instance which accepts requests from one or more client-facing HTTP servers that act as a proxy. JBoss EAP 6 can accept HTTP, HTTPS and AJP requests from Apache HTTP Server, Microsoft IIS or Oracle iPlanet Web Server (formerly Netscape Web Server).

For an overview of HTTP connectors supported by JBoss EAP 6 and how to configure them, see [Section 17.2.3, "Overview of HTTP Connectors"](#).

Cluster node

Cluster node is a specialization of the worker node. Such a cluster may be load-balancing, high-availability or both. In a load-balancing cluster, a central manager distributes work loads amongst its nodes equally, by some situation-specific measurement of equality. In a high-availability cluster, some nodes are actively doing work, and others are waiting to step in if one of the active nodes leaves the cluster.

Example 17.1. Apache HTTP Server with `mod_cluster` configured

For instance, you might have a setup where Apache HTTP Server with `mod_cluster` configured acts as a load-balancing proxy to client's requests. These requests are forwarded to particular cluster nodes according to their current load. In case the sticky sessions are disabled, all incoming requests are distributed according to the current load. In case sticky sessions are enabled (default),

subsequent requests within an already created session are routed to the worker node with which the session was created. If one of these cluster nodes dies or becomes overloaded, Apache HTTP Server with `mod_cluster` acting as a load-balancing proxy will forward client's request to another cluster node. Due to the session replication within the cluster, client's data will not be lost.

Similarly, you might have a simpler setup with worker nodes that are not clustered. If a worker node dies, its neighbor does not have the former session data, yet client will not get any non-HTTP 200 error code.

In the case of sticky session configuration, it does not matter to the load-balancing proxy whether or not its worker nodes are clustered.

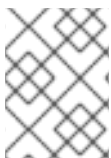
[Report a bug](#)

17.3. CONNECTOR CONFIGURATION

17.3.1. Define Thread Pools for HTTP Connector in JBoss EAP 6

Summary

Thread Pools in JBoss EAP 6 can be shared between different components using the Executor model. These pools can be shared not only by different (HTTP) connectors, but also by other components within JBoss EAP 6 that support the Executor model. Getting the HTTP connector thread pool to match your current web performance requirements is tricky and requires close monitoring of the current thread pool and the current and anticipated web load demands. In this task, you will learn how to set the a thread pool for an HTTP Connector using the Executor model. You will learn how to set this using both the Management CLI and by modifying the XML configuration file.



NOTE

If you are running JBoss EAP in domain mode, add the prefix `/profile=PROFILE_NAME` to all Management CLI commands in this procedure.

Procedure 17.2. Setup a thread pool for an HTTP Connector

1. Define a thread factory

Open up your configuration file (**`standalone.xml`** if modifying for a standalone server or **`domain.xml`** if modifying for a domain based configuration. This file will be in the **`EAP_HOME/standalone/configuration`** or the **`EAP_HOME/domain/configuration`** folder).

Add the following subsystem entry, changing the values to suit your server requirements.

```
<subsystem xmlns="urn:jboss:domain:threads:1.1">
  <thread-factory name="http-connector-factory" thread-name-pattern="HTTP-%t"
    priority="9" group-name="uq-thread-pool"/>
</subsystem>
```

If you prefer to use the Management CLI to do this task, then execute the following command in a CLI command prompt:

```
[standalone@localhost:9999 /] ./subsystem=threads/thread-factory=http-connector-
factory:add(thread-name-pattern="HTTP-%t", priority="9", group-name="uq-thread-pool")
```

2. Create an executor

You can use one of six in-built executor classes to act as the executor for this factory. The six executors are:

- **unbounded-queue-thread-pool**: This type of thread pool always accepts tasks. If fewer than the maximum number of threads are running, a new thread is started up to run the submitted task; otherwise, the task is placed into an unbounded FIFO queue to be executed when a thread is available.



NOTE

The single-thread executor type provided by **Executors.singleThreadExecutor()** is essentially an unbounded-queue executor with a thread limit of one. This type of executor is deployed using the **unbounded-queue-thread-pool-executor** element.

- **bounded-queue-thread-pool**: This type of executor maintains a fixed-length queue and two pool sizes: a **core** size and a **maximum** size. When a task is accepted, if the number of running pool threads is less than the **core** size, a new thread is started to execute the task. If space remains in the queue, the task is placed in the queue. If the number of running pool threads is less than the **maximum** size, a new thread is started to execute the task. If blocking is enabled on the executor, the calling thread will block until space becomes available in the queue. The task is delegated to the handoff executor, if a handoff executor is configured. Otherwise, the task is rejected.
- **blocking-bounded-queue-thread-pool**: A thread pool executor with a bounded queue where threads submitting tasks may block. Such a thread pool has a core and maximum size and a specified queue length. When a task is submitted, if the number of running threads is less than the core size, a new thread is created. Otherwise, if there is room in the queue, the task is enqueued. Otherwise, if the number of running threads is less than the maximum size, a new thread is created. Otherwise, the caller blocks until room becomes available in the queue.
- **queueless-thread-pool**: Sometimes, a simple thread pool is required to run tasks in separate threads, reusing threads as they complete their tasks with no intervening queue. This type of pool is ideal for handling tasks which are long-running, perhaps utilizing blocking I/O, since tasks are always started immediately upon acceptance rather than accepting a task and then delaying its execution until other running tasks have completed. This type of executor is declared using the **queueless-thread-pool-executor** element.
- **blocking-queueless-thread-pool**: A thread pool executor with no queue where threads submitting tasks may block. When a task is submitted, if the number of running threads is less than the maximum size, a new thread is created. Otherwise, the caller blocks until another thread completes its task and accepts the new one.
- **scheduled-thread-pool**: This is a special type of executor whose purpose is to execute tasks at specific times and time intervals, based on the **java.util.concurrent.ScheduledThreadPoolExecutor** class. This type of executor is configured with the **scheduled-thread-pool-executor** element:

In this example, we will use the **unbounded-queue-thread-pool** to act as the executor. Modify the values of **max-threads** and **keepalive-time** parameters to suit your server needs.

```
<unbounded-queue-thread-pool name="uq-thread-pool">
  <thread-factory name="http-connector-factory" />
```

```
<max-threads count="10" />
<keepalive-time time="30" unit="seconds" />
</unbounded-queue-thread-pool>
```

Or if you prefer to use the Management CLI:

```
[standalone@localhost:9999 /] ./subsystem=threads/unbounded-queue-thread-pool=uq-
thread-pool:add(thread-factory="http-connector-factory", keepalive-time={time=30,
unit="seconds"}, max-threads=30)
```

3. Make the HTTP web connector use this thread pool

In the same configuration file, locate the HTTP connector element under the web subsystem and modify it to use the thread pool defined in the previous steps.

```
<connector name="http" protocol="HTTP/1.1" scheme="http" socket-binding="http"
executor="uq-thread-pool" />
```

Again, if you prefer to use the Management CLI:

```
[standalone@localhost:9999 /] ./subsystem=web/connector=http:write-
attribute(name=executor, value="uq-thread-pool")
```

4. Restart the server

Restart the server (standalone or domain) so that the changes can take effect. Use the following Management CLI commands to confirm if the changes from the steps above have taken place:

```
[standalone@localhost:9999 /] ./subsystem=threads:read-resource(recursive=true)
{
  "outcome" => "success",
  "result" => {
    "blocking-bounded-queue-thread-pool" => undefined,
    "blocking-queueless-thread-pool" => undefined,
    "bounded-queue-thread-pool" => undefined,
    "queueless-thread-pool" => undefined,
    "scheduled-thread-pool" => undefined,
    "thread-factory" => {"http-connector-factory" => {
      "group-name" => "uq-thread-pool",
      "name" => "http-connector-factory",
      "priority" => 9,
      "thread-name-pattern" => "HTTP-%t"
    }},
    "unbounded-queue-thread-pool" => {"uq-thread-pool" => {
      "keepalive-time" => {
        "time" => 30L,
        "unit" => "SECONDS"
      },
      "max-threads" => 30,
      "name" => "uq-thread-pool",
      "thread-factory" => "http-connector-factory"
    }
  }
}
[standalone@localhost:9999 /] ./subsystem=web/connector=http:read-
```

```
resource(recursive=true)
{
  "outcome" => "success",
  "result" => {
    "configuration" => undefined,
    "enable-lookups" => false,
    "enabled" => true,
    "executor" => "uq-thread-pool",
    "max-connections" => undefined,
    "max-post-size" => 2097152,
    "max-save-post-size" => 4096,
    "name" => "http",
    "protocol" => "HTTP/1.1",
    "proxy-name" => undefined,
    "proxy-port" => undefined,
    "redirect-port" => 443,
    "scheme" => "http",
    "secure" => false,
    "socket-binding" => "http",
    "ssl" => undefined,
    "virtual-server" => undefined
  }
}
```

Result

You have successfully created a thread factory and an executor and modified your HTTP Connector to use this thread pool.

[Report a bug](#)

17.4. WEB SERVER CONFIGURATION

17.4.1. About the Standalone Apache HTTP Server

JBoss EAP 6 is tested and supported with the Apache HTTP server which is included with certified versions of Red Hat Enterprise Linux 6. Apache HTTP server is also available for other operating systems, such as Microsoft Windows Server. However, since Apache HTTP server is a separate product produced by the Apache Foundation, it was previously difficult to be sure that the version of Apache HTTP server a customer used was compatible with JBoss EAP.

A standalone Apache HTTP server bundle is now available as a separate download with JBoss EAP 6. This simplifies installation and configuration in environments other than Red Hat Enterprise Linux, or on systems which already have a configured Apache HTTP server and want to use a separate instance for web applications. You can download this Apache HTTP server as a separate download in the Customer Service Portal, listed under the available JBoss EAP 6 downloads for your installation platform.

[Report a bug](#)

17.4.2. HTTPD Variable Conventions

Table 17.4. Native

Product	HTTPD_CONF	HTTPD_MODULES
Red Hat Enterprise Linux	/etc/httpd/conf	/etc/httpd/modules
HPUX	/opt/hpws/apache/conf	/opt/hpws/apache/modules

Table 17.5. EWS

Product	HTTPD_CONF	HTTPD_MODULES
Red Hat Enterprise Linux	/HTTPD_HOME/EWS-ROOT/httpd/conf	/HTTPD_HOME/EWS-ROOT/httpd/modules
Solaris	/HTTPD_HOME/EWS-ROOT/etc/httpd/conf	/HTTPD_HOME/EWS-ROOT/lib/httpd/modules or /HTTPD_HOME/EWS-ROOT/lib64/httpd/modules
Windows	/HTTPD_HOME/EWS-ROOT/etc/httpd/conf	/HTTPD_HOME/EWS-ROOT/lib/httpd/modules or /HTTPD_HOME/EWS-ROOT/lib64/httpd/modules

[Report a bug](#)

17.4.3. Install Apache HTTP Server in Red Hat Enterprise Linux 5, 6, and 7 (Zip)

Prerequisites

- Root-level or administrator access.
- A supported version of Java installed.
- The following packages installed:
 - **krb5-workstation**
 - **mod_auth_kerb** (required for Kerberos functionality)
 - **elinks** (required for the **apachectl** functionality)
 - **apr-util-devel** (Apache Portability Runtime (APR))
 - **apr-util-ldap** (Red Hat Enterprise Linux 7 *only*, required for LDAP authentication functionality)

The **Apache HTTP Server** Zip archive contains symbolic links to several Kerberos modules, which is why

the **mod_auth_kerb** package is a prerequisite. If Kerberos functionality is not required, there is no need to install the **mod_auth_kerb** package and the associated symbolic link can be deleted:
EAP_HOME/httpd/modules/mod_auth_kerb.so.

Procedure 17.3. Install the Apache HTTP Server

1. **Navigate to the JBoss EAP downloads list for your platform, on the Red Hat Customer Portal.**

Log in to the Customer Portal and navigate to the **Software Downloads** page. Select the appropriate **Product** and **Version**.

2. **Choose the Apache HTTP Server binary from the list.**

Find the **Apache HTTP Server** option for your operating system and architecture. Click the **Download** link. A Zip file containing the Apache HTTP Server distribution downloads to your computer.

3. **Extract the Zip to the system where the Apache HTTP Server binary will run.**

Extract the Zip file on your preferred server, to a temporary location. The Zip file will contain the **httpd** directory under a *jboss-ews-version-number* folder. Copy the **httpd** folder and place it inside the *EAP_HOME* directory.

Your Apache HTTP Server is now located in the *EAP_HOME/httpd/* directory. This directory is referred to as *HTTPD_HOME*.

4. **Run the Post-installation script and create the `apache` user and group accounts**

In a terminal emulator, navigate to the *EAP_HOME/httpd* directory and execute the following command with **root** user privileges.

```
./postinstall
```

Next, verify that the **apache** user exists on the system by running the following command:

```
id apache
```

If the user does not exist then it will need to be added, along with the appropriate usergroup. In order to achieve this, execute the following with **root** user privileges:

```
getent group apache >/dev/null || groupadd -g 48 -r apache  
getent passwd apache >/dev/null || useradd -r -u 48 \  
-g apache -s /sbin/nologin -d HTTPD_HOME/httpd/www -c "Apache" apache
```

Once this is completed, if the **apache** user will be running the Apache HTTP Server service, then the ownership of the HTTP directories will need to be changed to reflect this:

```
chown -R apache:apache httpd
```

To test that the above commands have been successful, check that the **apache** user has execution permission to the Apache HTTP Server install path.

```
ls -l
```

The output should be similar to:


```
drwxrwxr-- 11 apache apache 4096 Feb 14 06:52 httpd
```

5. Configure the Apache HTTP Server.

Prior to starting the Apache HTTP Server, configure it to meet the needs of your organization. You can use the documentation available from the Apache Foundation at <http://httpd.apache.org/> for general guidance.

6. Start the Apache HTTP Server.

Start the Apache HTTP Server using the following command:

```
HTTPD_HOME/httpd/sbin/apachectl start
```

7. Stop the Apache HTTP Server.

To stop the Apache HTTP Server, issue the following command:

```
HTTPD_HOME/httpd/sbin/apachectl stop
```

[Report a bug](#)

17.4.4. Install Apache HTTP Server in Red Hat Enterprise Linux (RHEL) 5, 6, and 7 (RPM)

Prerequisites

- Root-level access.
- The latest version of elinks package installed (required for the apachectl functionality).
- Subscribe to Red Hat Enterprise Linux (RHEL) channels (to install Apache HTTP Server from RHEL channels).
- Subscribe to **jbappplatform-6-ARCH-server-VERS-rpm** Red Hat Network (RHN) channel (to install EAP specific distribution of Apache HTTP Server).

You can install Apache HTTP Server using either of the following methods:

- From Red Hat Enterprise Linux (RHEL) channels: An active subscription to Red Hat Enterprise Linux (RHEL) channels is necessary to install Apache HTTP server.
- From **jbappplatform-6-ARCH-server-VERS-rpm** channel (JBoss EAP specific distribution): JBoss EAP distributes its own version of the Apache HTTP Server. An active subscription to **jbappplatform-6-ARCH-server-VERS-rpm** channel is necessary to install the JBoss EAP specific distribution of Apache HTTP Server.

Procedure 17.4. Install and Configure Apache HTTP Server in Red Hat Enterprise Linux 5 and 6 (RPM)

1. Install httpd

To install the JBoss EAP specific version of **httpd** package run the following command:

```
yum install httpd
```

To install **httpd** explicitly from Red Hat Enterprise Linux (RHEL) channels run the following command:

```
yum install httpd --disablerepo=jbappplatform-6-*
```



NOTE

You must run only one of the above commands to install the **httpd** package on your system.

2. Set the Service Boot Behavior

You can define the service behavior for the **httpd** service at boot from the command line or with the service configuration graphical tool. Run the following command to define the behavior:

```
chkconfig httpd on
```

To use the service configuration tool run the following command and change the service setting in the displayed window:

```
system-config-services
```

3. Start httpd

Start **httpd** using the following command:

```
service httpd start
```

4. Stop httpd

Stop **httpd** using the following command:

```
service httpd stop
```

Procedure 17.5. Install and Configure Apache HTTP Server in Red Hat Enterprise Linux 7 (RPM)

1. Install httpd22

To install the JBoss EAP specific version of **httpd22** package run the following command:

```
yum install httpd22
```

2. Set the Service Boot Behavior

Run the following command to start the **httpd22** service at boot:

```
systemctl enable httpd22.service
```

3. Start httpd22

Start **httpd22** using the following command:

```
systemctl start httpd22.service
```

4. Stop httpd22

Stop **httpd22** using the following command:

```
systemctl stop httpd22.service
```

[Report a bug](#)

17.4.5. Manage Apache HTTP Server Service for Microsoft Windows Server Environment

Procedure 17.6. Install the Apache HTTP Server service for Microsoft Windows Server environment

- Install the Apache HTTP Server service using this command.

```
cd /D "%EWS_HOME%\bin"  
httpd -k install
```

This command installs an Apache HTTP Server service named Apache2.2.

To specify a different name for the service, for example, ApacheBalancer, use the following command.

```
cd /D "%EWS_HOME%\bin"  
httpd -k install -n ApacheBalancer
```

Procedure 17.7. Start the Apache HTTP Server service for Microsoft Windows Server environment

- To start a service, you can either use `httpd.exe` or service manager.

Using `httpd.exe`:

```
cd /D "%EWS_HOME%\bin"  
httpd -k start -n Apache2.2
```

Using service manager:

```
net start Apache2.2
```

Procedure 17.8. Stop the Apache HTTP Server service for Microsoft Windows Server environment

- To stop a service, you can either use `httpd.exe` or service manager.

Using `httpd.exe`:

```
cd /D "%EWS_HOME%\bin"  
httpd -k stop -n Apache2.2
```

Using service manager:

```
net stop Apache2.2
```

Procedure 17.9. Uninstall the Apache HTTP Server service for Microsoft Windows Server environment

- To uninstall a service, it must be referenced by name. For example, to uninstall the service names ApacheBalancer, use the following command.

```
cd /D "%EWS_HOME%\bin"
httpd -k uninstall -n ApacheBalancer
```

[Report a bug](#)

17.4.6. mod_cluster Configuration on Apache HTTP Server

Summary

The mod_cluster connector is an Apache HTTP Server-based load balancer. It uses a communication channel to forward requests from the Apache HTTP Server to one of a set of application server nodes. The following derivatives can be set to configure mod_cluster.



NOTE

There is no need to use ProxyPass directives because mod_cluster automatically configures the URLs that must be forwarded to Apache HTTP Server.

Table 17.6. mod_cluster Derivatives

Derivative	Description	Values
CreateBalancers	Defines how the balancers are created in the Apache HTTP Server VirtualHosts. This allows directives like: ProxyPass /balancer://mycluster1/ .	0: Create all VirtualHosts defined in Apache HTTP Server 1: Do not create balancers (at least one ProxyPass or ProxyMatch is required to define the balancer names) 2: Create only the main server Default: 2 While using the value 1, do not forget to configure the balancer in the ProxyPass directive, because the default is an empty stickysession and nofailover=Off and the values received via the MCMP CONFIG message are ignored.
UseAlias	Check that the alias corresponds to the server name.	0: Ignore aliases 1: Check aliases Default: 0
LBstatusRecalTime	Time interval in seconds for loadbalancing logic to recalculate the status of a node.	Default: 5 seconds
WaitBeforeRemove	Time in seconds before a removed node is forgotten by httpd.	Default: 10 seconds

Derivative	Description	Values
ProxyPassMatch/ProxyPass	<p>ProxyPassMatch and ProxyPass are mod_proxy directives which, when using ! (instead of the back-end URL), prevent reverse-proxy in the path. This is used to allow Apache HTTP Server to serve static content. For example,</p> <p>ProxyPassMatch ^(/.*\.gif)\$!</p> <p>The above example allows the Apache HTTP Server to serve the .gif files directly.</p>	

A hot-standby node in the mod_cluster logic is the last resort node to which all requests are routed if all other nodes are down. This is similar to the hot-standby logic in mod_proxy.

To configure a hot-standby node, replace the dynamic-load-provider in mod_cluster subsystem with a simple-load-provider with factor set to 0, for example:

```
<subsystem xmlns="urn:jboss:domain:modcluster:1.2">
  <mod-cluster-config advertise-socket="modcluster" connector="ajp">
    - <dynamic-load-provider>
    -   <load-metric type="busyness"/>
    - </dynamic-load-provider>
    + <simple-load-provider factor="0"/>
  </mod-cluster-config>
</subsystem>
```

In mod_cluster-manager console, the node is displayed with OK status and Load: 0. For more information, refer *Apache mod_cluster-manager Application* section in the *JBoss Enterprise Application Platform Development Guide*.

For instance, if there are three nodes:

- Node A, Load: 10
- Node B, Load: 10
- Node C, Load: 0

The load will be balanced between nodes A and B. If both the nodes are unavailable, node C will take the load.

mod_manager

The context of a mod_manager directive is VirtualHost in all cases, except when mentioned otherwise. **server config** context implies that the directive must be outside a VirtualHost configuration. If not, an error message is displayed and the Apache HTTP Server does not start.

Table 17.7. mod_manager Derivatives

Derivative	Description	Values
EnableMCPMReceive	Allow the VirtualHost to receive the MCPM from the nodes. Include EnableMCPMReceive in the Apache HTTP Server configuration to allow mod_cluster to work. Save it in the VirtualHost where you configure advertising.	
MemManagerFile	The base name for the names that mod_manager uses to store configuration, generate keys for shared memory or locked files. This must be an absolute path name; the directories are created if needed. It is recommended that these files are placed on a local drive and not an NFS share. Context: server config	\$server_root/logs/
Maxcontext	The maximum number of contexts supported by mod_cluster Context: server config	Default: 100
Maxnode	The maximum number of nodes supported by mod_cluster. Context: server config	Default: 20
Maxhost	The maximum number of hosts (aliases) supported by mod_cluster. It also includes the maximum number of balancers. Context: server config	Default: 20
Maxsessionid	The number of active sessionid stored to provide the number of active sessions in the mod_cluster-manager handler. A session is inactive when mod_cluster does not receive any information from the session within 5 minutes. Context: server config This field is for demonstration and debugging purposes only.	0: the logic is not activated.
MaxMCMPMaxMessSize	The maximum size of MCMP messages from other Max directives	Calculated from other Max directives. Min: 1024

Derivative	Description	Values
ManagerBalancerName	The name of balancer to use when the JBoss EAP instance does not provide a balancer name.	mycluster
PersistSlots	Tells mod_slotmem to persist nodes, aliases and contexts in files. Context: server config	Off
CheckNonce	Switch check of nonce when using mod_cluster-manager handler.	on/off Default: on - Nonce checked
AllowDisplay	Switch additional display on mod_cluster-manager main page.	on/off Default: off - only version is displayed
AllowCmd	Allow commands using mod_cluster-manager URL.	on/off Default: on - Commands allowed
ReduceDisplay	Reduce the information displayed on the main mod_cluster-manager page, so that more nodes can be displayed on the page.	on/off Default: off - full information is displayed
SetHandler mod_cluster-manager	Displays information about the node that mod_cluster sees from the cluster. The information includes generic information and additionally counts the number of active sessions. <pre> <Location /mod_cluster- manager> SetHandler mod_cluster-manager Order deny,allow Allow from 127.0.0.1 </Location> </pre>	on/off Default: off

**NOTE**

When accessing the location defined in **httpd.conf**:

Transferred: Corresponds to the POST data sent to the back-end server.

Connected: Corresponds to the number of requests that have been processed when the `mod_cluster` status page was requested.

Num_sessions: Corresponds to the number of sessions `mod_cluster` report as active (on which there was a request within the past 5 minutes). This field is not present when `Maxsessionid` is zero and is for demonstration and debugging purposes only.

[Report a bug](#)

17.4.7. Use an External Web Server as the Web Front-end for JBoss EAP 6 Applications

Overview

For reasons to use an external web server as the web front-end, as well as advantages and disadvantages of the different HTTP connectors supported by JBoss EAP 6, refer to [Section 17.2.3, "Overview of HTTP Connectors"](#). In some situations, you can use the Apache HTTP Server that comes with your operating system. Otherwise, you can use the Apache HTTP Server that ships as part of JBoss Enterprise Web Server.

After you have decided which web server and HTTP connector to use, refer to one of the following procedures:

- [Section 17.4.3, "Install Apache HTTP Server in Red Hat Enterprise Linux 5, 6, and 7 \(Zip\)"](#)
- [Section 17.6.3, "Install the `mod_cluster` Module Into Apache HTTP Server or JBoss Enterprise Web Server \(Zip\)"](#)
- [Section 17.7.3, "Install the `mod_jk` Module Into the Apache HTTP Server \(ZIP\)"](#)
- [Section 17.9.3, "Configure Microsoft IIS to Use the ISAPI Connector"](#)
- [Section 17.10.2, "Configure the NSAPI Connector on Oracle Solaris"](#)
- [Section 17.4.8, "Configure JBoss EAP 6 to Accept Requests From External Web Servers"](#)

[Report a bug](#)

17.4.8. Configure JBoss EAP 6 to Accept Requests From External Web Servers

Overview

JBoss EAP 6 does not need to know which proxy it is accepting requests from, only the port and protocol to look for. This is not true of **`mod_cluster`**, which is more tightly coupled to the configuration of JBoss EAP 6. But the following task works for **`mod_jk`**, **`mod_proxy`**, **ISAPI connector**, and **NSAPI connector**. Substitute the protocols and ports in the examples with the ones you need to configure.

To configure JBoss EAP 6 for **`mod_cluster`**, refer to [Section 17.6.6, "Configure a `mod_cluster` Worker Node"](#).

Prerequisites

- You need to be logged into the Management CLI or Management Console to perform this task. The exact steps in the task use the Management CLI, but the same basic procedure is used in the Management Console.
- You need a list of which protocols you will be using, whether HTTP, HTTPS, or AJP.

Procedure 17.10. Edit Configuration and add Socket Bindings

1. Configure the `jvmRoute` system property.

For a standalone mode instance, remove the prefix `/host=NODE_NAME`. Replace **NODE_NAME** with the name of the host.

```
/host=NODE_NAME/system-property=jvmRoute/:add(value=NODE_NAME)
```

2. List the connectors available in the web subsystem.



NOTE

This step is only necessary if you are not using the **ha** or **full-ha** profiles for either a standalone server, or a server group in a Managed Domain. Those configurations already include all of the necessary connectors.

In order for an external web server to be able to connect to JBoss EAP 6's web server, the web subsystem needs a connector. Each protocol needs its own connector, which is tied to a socket group.

To list the connectors currently available, issue the following command:

```
/subsystem=web:read-children-names(child-type=connector)
```

If there is no line indicating the connector you need (HTTP, HTTPS, AJP), you need to add the connector.

3. Read the configuration of a connector.

To see the details of how a connector is configured, you can read its configuration. The following command reads the configuration of the AJP connector. The other connectors have similar configuration output.

```
/subsystem=web/connector=ajp:read-resource(recursive=true)
{
  "outcome" => "success",
  "result" => {
    "enable-lookups" => false,
    "enabled" => true,
    "max-post-size" => 2097152,
    "max-save-post-size" => 4096,
    "protocol" => "AJP/1.3",
    "redirect-port" => 8443,
    "scheme" => "http",
    "secure" => false,
    "socket-binding" => "ajp",
    "ssl" => undefined,
```

```

    "virtual-server" => undefined
  }
}

```

4. Add the necessary connectors to the web subsystem.

To add a connector to the web subsystem, it must have a socket binding. The socket binding is added to the socket binding group used by your server or server group. The following steps assume that your server group is **server-group-one** and that your socket binding group is **standard-sockets**.

a. Add a socket to the socket binding group.

To add a socket to the socket binding group, issue the following command, replacing the protocol and port with the ones you need.

```
/socket-binding-group=standard-sockets/socket-binding=ajp:add(port=8009)
```

b. Add the socket binding to the web subsystem.

Issue the following command to add a connector to the web subsystem, substituting the socket binding name and protocol with the ones you need.

```
/subsystem=web/connector=ajp:add(socket-binding=ajp, protocol="AJP/1.3",
enabled=true, scheme="http")
```

[Report a bug](#)

17.5. CLUSTERING

17.5.1. Use TCP Communication for the Clustering Subsystem

By default, cluster nodes monitor each other's status using the UDP protocol. Some networks only allow TCP to be used. In this situation, you can add the **TCPPING** protocol stack to your configuration and use it as the default mechanism. These configuration options are available in the command-line based Management CLI.

The **mod_cluster** subsystem also uses UDP communication by default, and you can choose to use TCP here as well.

Refer to the following two procedures to configure JGroups and mod_cluster subsystems to use TCP for network communication:

- [Section 17.5.2, "Configure the JGroups Subsystem to Use TCP"](#)
- [Section 17.5.3, "Disable Advertising for the mod_cluster Subsystem"](#)

[Report a bug](#)

17.5.2. Configure the JGroups Subsystem to Use TCP

By default, the JGroups subsystem communicates using multicast UDP. Use the following procedure to configure the JGroups subsystem to use unicast TCP instead.

To configure the **mod_cluster** subsystem to use TCP as well, see [Section 17.5.3, "Disable Advertising for the mod_cluster Subsystem"](#).

1. Modify the following script to suit your environment.

Copy the following script into a text editor. If you use a different profile on a managed domain, change the profile name. If you use a standalone server, remove the `/profile=full-ha` portion of the commands. Modify the properties listed at the bottom of the command as follows. Each of these properties is optional.

initial_hosts

A comma-separated list of hosts, using the syntax `HOST[PORT]`, that are considered well-known and will be available to look up the initial membership.

port_range

If desired, you can assign a port range. If you assign a port range of 2, and the initial port for a host is 7600, then TCPPING will attempt to contact the host on ports 7600-7602. The port range applies to each address specified in `initial_hosts`. This property is optional.

timeout

An optional timeout value, in milliseconds, for cluster members.

num_initial_members

The number of nodes before the cluster is considered to be complete. This property is optional.

```
batch
## If tcp is already added then you can remove it ##
/profile=full-ha/subsystem=jgroups/stack=tcp:remove
/profile=full-ha/subsystem=jgroups/stack=tcp:add(transport={"type" =>"TCP", "socket-
binding" => "jgroups-tcp"})
/profile=full-ha/subsystem=jgroups/stack=tcp/:add-protocol(type=TCPPING)
/profile=full-ha/subsystem=jgroups/stack=tcp/:add-protocol(type=MERGE2)
/profile=full-ha/subsystem=jgroups/stack=tcp/:add-protocol(type=FD_SOCKET,socket-
binding=jgroups-tcp-fd)
/profile=full-ha/subsystem=jgroups/stack=tcp/:add-protocol(type=FD)
/profile=full-ha/subsystem=jgroups/stack=tcp/:add-protocol(type=VERIFY_SUSPECT)
/profile=full-ha/subsystem=jgroups/stack=tcp/:add-protocol(type=BARRIER)
/profile=full-ha/subsystem=jgroups/stack=tcp/:add-protocol(type=pbcast.NAKACK)
/profile=full-ha/subsystem=jgroups/stack=tcp/:add-protocol(type=UNICAST2)
/profile=full-ha/subsystem=jgroups/stack=tcp/:add-protocol(type=pbcast.STABLE)
/profile=full-ha/subsystem=jgroups/stack=tcp/:add-protocol(type=pbcast.GMS)
/profile=full-ha/subsystem=jgroups/stack=tcp/:add-protocol(type=UFC)
/profile=full-ha/subsystem=jgroups/stack=tcp/:add-protocol(type=MFC)
/profile=full-ha/subsystem=jgroups/stack=tcp/:add-protocol(type=FRAG2)
/profile=full-ha/subsystem=jgroups/stack=tcp/:add-protocol(type=RSVP)
/profile=full-ha/subsystem=jgroups:write-attribute(name=default-stack,value=tcp)
run-batch
/profile=full-
ha/subsystem=jgroups/stack=tcp/protocol=TCPPING/property=initial_hosts/:add(value="HostA[
600],HostB[7600]")
/profile=full-
ha/subsystem=jgroups/stack=tcp/protocol=TCPPING/property=port_range/:add(value=0)
/profile=full-
ha/subsystem=jgroups/stack=tcp/protocol=TCPPING/property=timeout/:add(value=3000)
```

```
/profile=full-
ha/subsystem=jgroups/stack=tcp/protocol=TCPPING/property=num_initial_members/:add(value
=3)
```

2. Run the script in batch mode.



WARNING

The servers running the profile have to be shutdown before executing the batch file.

In a terminal emulator, navigate to the directory containing the **jboss-cli.sh** script and enter the command

```
./jboss-cli.sh -c --file=SCRIPT_NAME
```

where *SCRIPT_NAME* is the name and path containing the script.

Result

The **TCPPING** stack is now available to the JGroups subsystem. If it is used, the JGroups subsystem uses TCP for all network communication. To configure the **mod_cluster** subsystem to use TCP as well, see [Section 17.5.3, "Disable Advertising for the mod_cluster Subsystem"](#).

[Report a bug](#)

17.5.3. Disable Advertising for the mod_cluster Subsystem

By default, the **mod_cluster** subsystem's balancer uses multicast UDP to advertise its availability to the background workers. If you wish, you can disable advertisement. Use the following procedure to configure this behavior.

Procedure 17.11.

1. Modify the Apache HTTP Server configuration.

Modify the Apache HTTP Server configuration to disable server advertising and to use a proxy list instead. The proxy list is configured on the worker, and contains all of the **mod_cluster**-enabled Web servers to which the worker can talk.

The **mod_cluster** configuration for the Web server is located in *HTTPD_HOME*. See [Section 17.6.3, "Install the mod_cluster Module Into Apache HTTP Server or JBoss Enterprise Web Server \(Zip\)"](#) and [Section 17.6.5, "Configure Server Advertisement Properties for Your mod_cluster-enabled Web Server"](#) for more information about the file itself. Open the file containing the virtual host which listens for MCPM requests (using the **EnableMCPMReceive** directive), and disable server advertising by changing the **ServerAdvertise** directive as follows.

```
ServerAdvertise Off
```

2. **Disable advertising within the `mod_cluster` subsystem of JBoss EAP 6, and provide a list of proxies.**

You can disable advertising for the `mod_cluster` subsystem and provide a list of proxies, by using the web-based Management Console or the command-line Management CLI. The list of proxies is necessary because the `mod_cluster` subsystem will not be able to automatically discover proxies if advertising is disabled.

o **Management Console**

If you use a managed domain, you can only configure `mod_cluster` in profiles where it is enabled, such as the `ha` and `full-ha` profiles.

1. Log in to the Management Console and select the **Configuration** tab at the top of the screen. If you use a managed domain, select either the `ha` or `full-ha` profile from the **Profile** drop-down menu at the top left.
2. Expand the **Subsystems** menu then expand the **Web** menu and select `mod_cluster`.
3. Click **Edit** under the **Advertising** tab under `mod_cluster`. To disable advertising, clear the check box next to **Advertise**, and click **Save**.

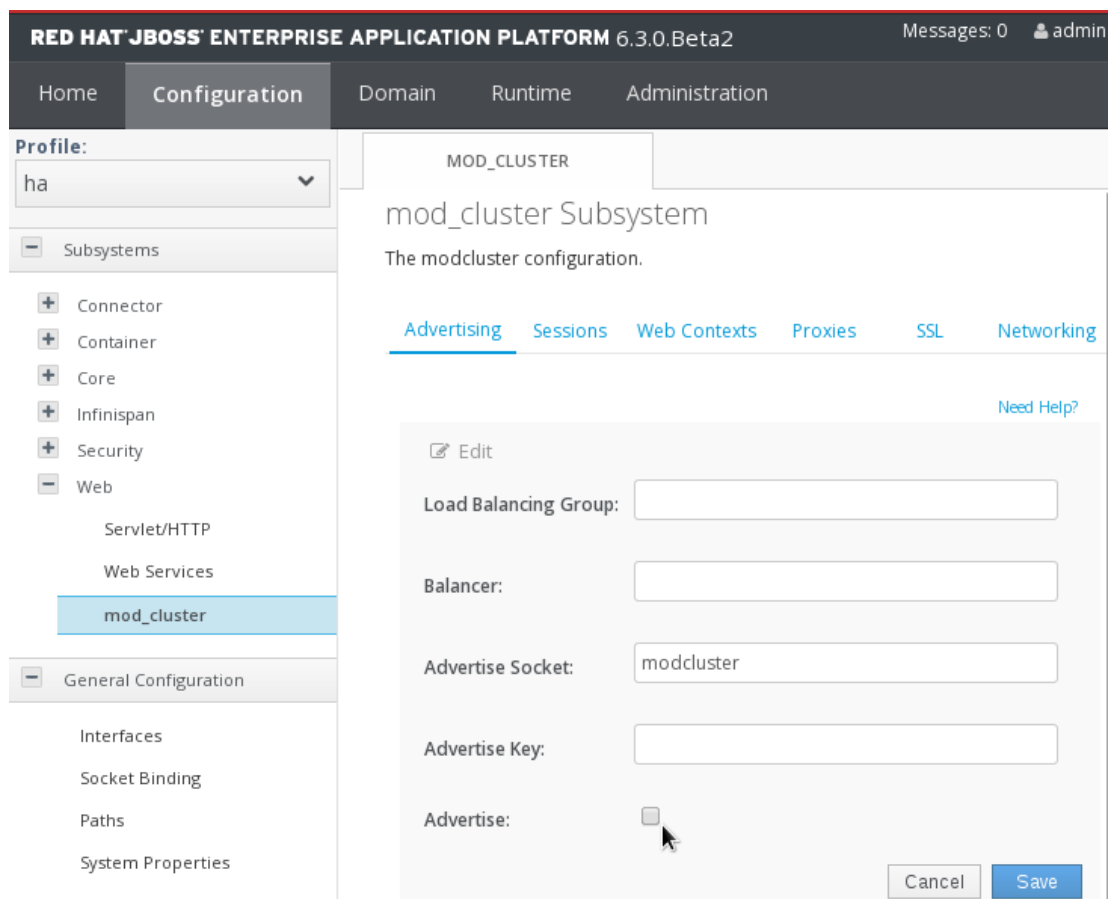


Figure 17.1. `mod_cluster` Advertising Configuration Screen

4. Click the **Proxies** tab. Click **Edit** and enter a list of proxy servers in the **Proxy List** field. The correct syntax is a comma-separated list of **HOSTNAME:PORT** strings, like the following:

```
10.33.144.3:6666,10.33.144.1:6666
```

Click the **Save** button to finish.

- o **Management CLI**

The following two Management CLI commands create the same configuration as the Management Console instructions above. They assume that you run a managed domain and that your server group uses the **full-ha** profile. If you use a different profile, change its name in the commands. If you use a standalone server using the **standalone-ha** profile, remove the **/profile=full-ha** portion of the commands.

```
/profile=full-ha/subsystem=modcluster/mod-cluster-config=configuration/:write-attribute(name=advertise,value=false)
```

```
/profile=full-ha/subsystem=modcluster/mod-cluster-config=configuration/:write-attribute(name=proxy-list,value="10.33.144.3:6666,10.33.144.1:6666")
```

Result

The Apache HTTP Server balancer no longer advertises its presence to worker nodes and UDP multicast is no longer used.



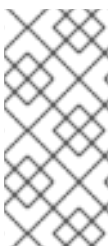
NOTE

In order to set the attribute **advertise="false"**, you must also set the attribute **proxy-list="address:port"**. If the **proxy-list** attribute is empty, the **advertise="false"** attribute is ignored. To disable the `mod_cluster` subsystem altogether, you may remove it from the server configuration.

[Report a bug](#)

17.5.4. Switch UDP to TCP for HornetQ Clustering

The following example uses the default `standalone-full-ha.xml` file shipped with EAP 6.



NOTE

If security is enabled, you must set the `cluster-password` attribute:

```
<cluster-password>${jboss.messaging.cluster.password:ChangeMe}</cluster-password>
```

1. Remove the `broadcast-groups` and `discovery-groups`:

```
<broadcast-groups>
  <broadcast-group name="bg-group1">
    <socket-binding>messaging-group</socket-binding>
    <broadcast-period>5000</broadcast-period>
    <connector-ref>netty</connector-ref>
  </broadcast-group>
</broadcast-groups>
<discovery-groups>
  <discovery-group name="dg-group1">
    <socket-binding>messaging-group</socket-binding>
    <refresh-timeout>10000</refresh-timeout>
  </discovery-group>
</discovery-groups>
```

2. Optionally, remove the "messaging-group" socket-binding:

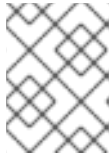
```
<socket-binding name="messaging-group" port="0" multicast-
address="{jboss.messaging.group.address:231.7.7.7}" multicast-
port="{jboss.messaging.group.port:9876}"/>
```

3. Configure the appropriate Netty connector(s) - one for each of the other nodes in the cluster.

For example, if the cluster is 3 nodes then configure 2 Netty connectors, etc., if the cluster is 2 nodes then configure 1 Netty connector, etc. Here is a sample configuration for a 3-node cluster:

```
<netty-connector name="other-cluster-node1" socket-binding="other-cluster-node1"/>
<netty-connector name="other-cluster-node2" socket-binding="other-cluster-node2"/>
```

4. Configure the related socket bindings.



NOTE

The system property substitution can be used for either "host" or "port", if required.

```
<outbound-socket-binding name="other-cluster-node1">
  <remote-destination host="otherNodeHostName1" port="5445"/>
</outbound-socket-binding>
<outbound-socket-binding name="other-cluster-node2">
  <remote-destination host="otherNodeHostName2" port="5445"/>
</outbound-socket-binding>
```

5. Configure the cluster-connection to use these connectors instead of the discovery-group, which is used by default:

```
<cluster-connection name="my-cluster">
  <address>jms</address>
  <connector-ref>netty</connector-ref>
  <static-connectors>
    <connector-ref>other-cluster-node1</connector-ref>
    <connector-ref>other-cluster-node2</connector-ref>
  </static-connectors>
</cluster-connection>
```

This process has to be repeated on each of the cluster nodes so that each node has connectors to every other node in the cluster.



NOTE

Do not configure a node with a connection to itself. This is considered as a misconfiguration.

17.6. WEB, HTTP CONNECTORS, AND HTTP CLUSTERING

17.6.1. About the `mod_cluster` HTTP Connector

The `mod_cluster` module enables load balancing and is referred to as a `connector`. To learn about other connectors, see one of the following:

- [Section 17.7.1, “About the Apache `mod_jk` HTTP Connector”](#)
- [Section 17.9.1, “About the Internet Server API \(ISAPI\)”](#)
- [Section 17.10.1, “About the Netscape Server API \(NSAPI\)”](#)

The `mod_cluster` connector has several advantages over other connectors.

- The `mod_cluster Management Protocol (MCMP)` is an additional connection between the JBoss Enterprise Application Platform 6 servers and the Apache HTTP Server with the `mod_cluster` module enabled. It is used by the JBoss Enterprise Application Platform servers to transmit server-side load balance factors 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 JBoss EAP 6 servers to join the load balancing arrangement without manual configuration.
- JBoss EAP 6 performs the load-balancing factor 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 JBoss EAP 6 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](#)

17.6.2. Configure the `mod_cluster` Subsystem

The `mod_cluster` subsystem can be configured via the Management Console and Management CLI. In this topic the various configuration options are described, grouped as they appear in the Management Console. Example Management CLI commands are provided for each option.



NOTE

The `mod_cluster` configuration page is only visible for **ha** and **full-ha** profiles. For a managed domain these profiles are **ha** and **full-ha**, and for a standalone server they are **standalone-ha** and **standalone-full-ha**.

Management Console

Click the **Configuration** tab. If you are configuring a managed domain, select the correct profile from the **Profile** drop-down list. Expand the **Subsystems** menu, then expand the **Web** menu and select **mod_cluster**.

Table 17.8. mod_cluster Advertising Configuration Options

Option	Description	CLI Command
Load Balancing Group	If this is not null, requests are sent to a specific load balancing group on the load balancer. Leave this blank if you do not want to use load balancing groups. This is unset by default.	<pre>/subsystem=modcluster/mod-cluster-config=configuration/:write-attribute(name=load-balancing-group,value=myGroup)</pre>
Balancer	This attribute specifies what mod_proxy balancer is to be automatically configured by mod_cluster on the Apache HTTP Server. The default is none , in which case the default of mycluster is used (balancer://mycluster/ when expressed in mod_proxy terms). This default value is configured on the Apache HTTP Server side with the ManagerBalancerName directive. If you use two different balancer attribute values on the JBoss EAP 6 worker instances, there will be two different mod_proxy balancers created by mod_cluster automatically on the Apache HTTP Server.	<pre>/subsystem=modcluster/mod-cluster-config=configuration/:write-attribute(name=balancer,value=myBalancer)</pre>
Advertise Socket	The name of the socket binding to use for cluster advertising.	<pre>/subsystem=modcluster/mod-cluster-config=configuration/:write-attribute(name=advertise-socket,value=modcluster)</pre>
Advertise Security Key	A string containing the security key for advertising.	<pre>/subsystem=modcluster/mod-cluster-config=configuration/:write-attribute(name=advertise-security-key,value=myKey)</pre>
Advertise	Whether or not advertising is enabled. Defaults to true .	<pre>/subsystem=modcluster/mod-cluster-config=configuration/:write-attribute(name=advertise,value=true)</pre>

Table 17.9. `mod_cluster` Session Configuration Options

Option	Description	CLI Command
Sticky Session	Whether to use sticky sessions for requests. This means that after the client makes a connection to a specific node, further communication is routed to that same node unless it becomes unavailable. This defaults to true , which is the recommended setting.	<code>/subsystem=modcluster/mod-cluster-config=configuration/:write-attribute(name=sticky-session,value=true)</code>
Sticky Session Force	If true , a request is not redirected to a new node if its initial node becomes unavailable but instead it fails. This defaults to false .	<code>/subsystem=modcluster/mod-cluster-config=configuration/:write-attribute(name=sticky-session-force,value=false)</code>
Sticky Session Remove	Remove session information on failover. This defaults to false .	<code>/subsystem=modcluster/mod-cluster-config=configuration/:write-attribute(name=sticky-session-remove,value=false)</code>

Table 17.10. `mod_cluster` Web Context Configuration Options

Option	Description	CLI Command
Auto Enable Contexts	Whether to add new contexts to mod_cluster by default or not. This defaults to true . If you change the default and need to enable context manually, the Web Application can enable its context using the enable() MBean method, or via the mod_cluster manager, which is a web application which runs on the httpd proxy on a named virtual host or port which is specified in that httpd's configuration.	<code>/subsystem=modcluster/mod-cluster-config=configuration/:write-attribute(name=auto-enable-contexts,value=true)</code>
Excluded Contexts	A comma-separated list of contexts that mod_cluster should ignore. If no host is indicated, the host is assumed to be localhost . ROOT indicates the root context of the Web Application. The default value is ROOT,invoker,jbossws,juddi,console .	<code>/subsystem=modcluster/mod-cluster-config=configuration/:write-attribute(name=excluded-contexts,value="ROOT,invoker,jbossws,juddi,console")</code>

Table 17.11. `mod_cluster` Proxy Configuration Options

Option	Description	CLI Command
Proxy URL	If defined, this value will be prepended to the URL of MCMP commands.	<pre>/subsystem=modcluster/mod-cluster-config=configuration/:write-attribute(name=proxy-url,value=myhost)</pre>
Proxy List	A comma-separated list of httpd proxy addresses, in the format hostname:port . This indicates the list of proxies that the mod_cluster process will attempt to communicate with initially.	<pre>/subsystem=modcluster/mod-cluster-config=configuration/:write-attribute(name=proxy-list,value="127.0.0.1,127.0.0.2")</pre>


Configure SSL Communication for `mod_cluster`

By default, `mod_cluster` communication happens over an unencrypted HTTP link. If you set the connector scheme to **HTTPS** (refer to [Table 17.9, “`mod_cluster` Session Configuration Options”](#)), the settings below tell `mod_cluster` where to find the information to encrypt the connection.

Table 17.12. `mod_cluster` SSL Configuration Options

Option	Description	CLI Command
Key Alias	The key alias, which was chosen when the certificate was created.	<pre>/subsystem=modcluster/mod-cluster-config=configuration/ssl=configuration/:write-attribute(name=key-alias,value=jboss)</pre>

Option	Description	CLI Command
Password	<p>This password is the keystore password for both keystores: certificate-key-file (Key File) and ca-certificate-file (Cert File) and the key/certificate entry specified with Key Alias inside Cert File.</p>  <p>NOTE</p> <p>@ca-certificate-password is the truststore password and value remains undefined if you have not specified it.</p>	<pre>/subsystem=modcluster/mod-cluster- config=configuration/ssl=configuration/:write-attribute(name=password,value=changeit)</pre>
CA Cert File/Trust Store	Trust store used to validate the web server certificate.	<pre>/subsystem=modcluster/mod-cluster- config=configuration/ssl=configuration/:write-attribute(name=ca-certificate-file,value=\${user.home}/jboss.crt)</pre>
Key Store	Key store that holds the certificate and private key that identifies this instance.	<pre>/subsystem=modcluster/mod-cluster- config=configuration/ssl=configuration/:write-attribute(name=certificate-key-file,value=\${user.home}/.keystore)</pre>
Cipher Suite	The allowed encryption cipher suite.	<pre>/subsystem=modcluster/mod-cluster- config=configuration/ssl=configuration/:write-attribute(name=cipher-suite,value=ALL)</pre>

Option	Description	CLI Command
Revocation URL	The URL of the Certificate Authority revocation list.	<pre>/subsystem=modcluster/mod-cluster-config=configuration/ssl=configuration/:write-attribute(name=ca-revocation-url,value=jboss.crl)</pre>
Protocol	<p>The SSL protocols, which are enabled.</p> <p>You can also specify a combination of protocols, which is comma separated. For example, TLSv1, TLSv1.1, TLSv1.2.</p> <div style="border: 1px solid #ccc; background-color: #fff9c4; padding: 10px; margin-top: 10px;">  <p>WARNING</p> <p>Red Hat recommends that you explicitly disable SSL in favor of TLSv1.1 or TLSv1.2 in all affected packages.</p> </div>	<pre>/subsystem=modcluster/mod-cluster-config=configuration/ssl=configuration/:write-attribute(name=protocol,value="TLSv1, TLSv1.1, TLSv1.2")</pre>

Configure `mod_cluster` Networking Options

The available `mod_cluster` networking options control several different timeout behaviors for different types of services with which the `mod_cluster` service communicates.

Table 17.13. `mod_cluster` Networking Configuration Options

Option	Description	CLI Command
--------	-------------	-------------

Option	Description	CLI Command
Node Timeout	<p>Timeout, in seconds, for proxy connections to a worker. This is the time that mod_cluster will wait for the back-end response before returning an error. If the node-timeout attribute is undefined, the httpd ProxyTimeout directive is used. If ProxyTimeout is undefined, the httpd Timeout directive is used, which defaults to 300 seconds.</p>	<pre>/subsystem=modcluster/mod-cluster-config=configuration/:write-attribute(name=node-timeout,value=-1)</pre>
Socket Timeout	<p>Number of seconds to wait for a response from an httpd proxy to MCMP commands before timing out, and flagging the proxy as in error.</p>	<pre>/subsystem=modcluster/mod-cluster-config=configuration/:write-attribute(name=socket-timeout,value=20)</pre>
Stop Context Timeout	<p>The amount of time, measure in units specified by stopContextTimeoutUnit, for which to wait for clean shutdown of a context (completion of pending requests for a distributable context; or destruction/expiration of active sessions for a non-distributable context).</p>	<pre>/subsystem=modcluster/mod-cluster-config=configuration/:write-attribute(name=stop-context-timeout,value=10)</pre>
Session Draining Strategy	<p>Whether to drain sessions before undeploying a web application.</p> <p>DEFAULT</p> <p>Drain sessions before web application undeploy only if the web application is non-distributable.</p> <p>ALWAYS</p> <p>Always drain sessions before web application undeploy, even for distributable web applications.</p> <p>NEVER</p> <p>Do not drain sessions before web application undeploy, even for non-distributable web application.</p>	<pre>/subsystem=modcluster/mod-cluster-config=configuration/:write-attribute(name=session-draining-strategy,value=DEFAULT)</pre>

Option	Description	CLI Command
Max Attempts	Number of times an httpd proxy will attempt to send a given request to a node before giving up. The minimum value is 1 , meaning try only once. The mod_proxy default is also 1 , which means that no retry occurs.	<pre>/subsystem=modcluster/mod-cluster-config=configuration/:write-attribute(name=max-attempts,value=1)</pre>
Flush Packets	Whether or not to enable packet flushing to the Web server. Defaults to false .	<pre>/subsystem=modcluster/mod-cluster-config=configuration/:write-attribute(name=flush-packets,value=false)</pre>
Flush Wait	How long, in seconds, to wait before flushing packets to the Web server. Defaults to -1 . A value of -1 means to wait forever before flushing packets.	<pre>/subsystem=modcluster/mod-cluster-config=configuration/:write-attribute(name=flush-wait,value=-1)</pre>
Ping	How long, in seconds, to wait for a response to a ping from a worker. Defaults to 10 seconds.	<pre>/subsystem=modcluster/mod-cluster-config=configuration/:write-attribute(name=ping,value=10)</pre>
SMAX	Soft maximum idle connection count (the same as smax in mod_proxy documentation). The maximum value depends on the httpd thread configuration, and can be either ThreadsPerChild or 1 .	<pre>profile=full-ha/subsystem=modcluster/mod-cluster-config=configuration/:write-attribute(name=smax,value=ThreadsPerChild)</pre>
TTL	Time to live (in seconds) for idle connections above smax, default is 60	<pre>/subsystem=modcluster/mod-cluster-config=configuration/:write-attribute(name=ttl,value=-1)</pre>

mod_cluster Load Provider Configuration Options

The following **mod_cluster** configuration options are not available in the management console, but can only be set using the Management CLI.

A simple load provider is used if no dynamic load provider is present. It assigns each cluster member a load factor of **1**, and distributes work evenly without applying a load balancing algorithm. To add it, use the following Management CLI command.

```
[standalone@localhost:9990 /] /subsystem=modcluster/mod-cluster-config=configuration/:write-attribute(name=simple-load-provider, value=1)
```

A dynamic load provider can be configured to use a variety of algorithms in combination, in order to determine which worker receives the next request. You can create a load provider and configure it to suit your environment, and you can have more than one load metric active simultaneously by adding them via the CLI. The default dynamic load provider uses **busyness** as the determining load metric. The dynamic load provider options and possible load metrics are shown below.

Table 17.14. mod_cluster Dynamic Load Provider Options

Option	Description	CLI Command
Decay	The factor by which historical metrics should decay in significance.	<pre>/subsystem=modcluster/mod-cluster-config=configuration/dynamic-load-provider=configuration/:write-attribute(name=decay,value=2)</pre>
History	The number of historic load metric records to consider when determining the load.	<pre>/subsystem=modcluster/mod-cluster-config=configuration/dynamic-load-provider=configuration/:write-attribute(name=history,value=9)</pre>

Option	Description	CLI Command
Load Metric	The default load metric included with the dynamic load provider in JBoss EAP 6 is busyness , which calculates the load of the worker from the amount of threads in the thread pool being busy serving requests. You can set the capacity of this metric by which the actual load is divided: <code>calculated_load / capacity</code> . You can set multiple load metrics within the dynamic load provider.	<pre> /subsystem=modcluster/mod -cluster- config=configuration/dynamic -load- provider=configuration/load- metric=busyness/:write- attribute(name=capacity,value=1.0) /subsystem=modcluster/mod -cluster- config=configuration/dynamic -load- provider=configuration/load- metric=busyness/:write- attribute(name=type,value=b usyness) /subsystem=modcluster/mod -cluster- config=configuration/dynamic -load- provider=configuration/load- metric=busyness/:write- attribute(name=weight,value =1) </pre>

Load Metric Algorithms

cpu

The **cpu** load metric uses average CPU load to determine which node receives the next work load.

mem

The **mem** load metric uses free native memory as a load metric. Usage of this metric is discouraged because it provides a value that includes buffers and cache, so it is always a very low figure on every decent system with good memory management.

heap

The **heap** load metric uses the heap usage to determine which worker receives the next work load.

sessions

The **session** load metric uses the number of active sessions as a metric.

requests

The **requests** load metric uses the number of client requests to determine which worker receives the next work load. For instance, capacity 1000 means that 1000 requests/sec is considered to be a full load.

send-traffic

The **send-traffic** load metric uses the amount of traffic sent from the worker to the clients. E.g. the default capacity of 512 indicates that the node should be considered under full load if the average outbound traffic is 512 KB/s or higher.

receive-traffic

The receive-traffic load metric uses the amount of traffic sent to the worker from the clients. E.g. the default capacity of 1024 indicates that the worker should be considered under full load if the average inbound traffic is 1024 KB/s or higher.

busyness

This metric represents the amount of threads from the thread pool being busy serving requests.

Example 17.2. Add a Load Metric

To add a load metric, use the **add-metric** command.

```
/subsystem=modcluster/mod-cluster-config=configuration/:add-metric(type=sessions)
```

Example 17.3. Set a Value for an Existing Metric

To set a value for an existing metric, use the **write-attribute** command.

```
/subsystem=modcluster/mod-cluster-config=configuration/dynamic-load-provider=configuration/load-metric=cpu/:write-attribute(name="weight",value="3")
```

Example 17.4. Change the Value of an Existing Metric

To change the value of an existing metric, use the **write-attribute** command.

```
/subsystem=modcluster/mod-cluster-config=configuration/dynamic-load-provider=configuration/load-metric=cpu/:write-attribute(name="type",value="busyness")
```

Example 17.5. Remove an Existing Metric

To remove an existing metric, use the **remove-metric** command.

```
/subsystem=modcluster/mod-cluster-config=configuration/:remove-metric(type=sessions)
```

[Report a bug](#)

17.6.3. Install the mod_cluster Module Into Apache HTTP Server or JBoss Enterprise Web Server (Zip)

Prerequisites

- To perform this task, you must be using Apache HTTP Server installed in Red Hat Enterprise Linux 6, or JBoss Enterprise Web Server, or the standalone Apache HTTP Server included as a separate downloadable component of JBoss EAP 6.
- If you need to install Apache HTTP Server in Red Hat Enterprise Linux 6, use the instructions from the *Red Hat Enterprise Linux 6 Deployment Guide*.
- If you need to install the standalone Apache HTTP Server included as a separate downloadable component of JBoss EAP 6, refer to [Section 17.4.3, “Install Apache HTTP Server in Red Hat Enterprise Linux 5, 6, and 7 \(Zip\)”](#).
- If you need to install JBoss Enterprise Web Server, use the instructions from the *JBoss Enterprise Web Server Installation Guide*.
- Download the **Webserver Connector Natives** package for your operating system and architecture from the Red Hat Customer Portal at <https://access.redhat.com>. This package contains the mod_cluster binary web server modules precompiled for your operating system. After you extract the archive, the modules are located in the **EAP_HOME/modules/system/layers/base/native/lib/httpd/modules** directory.

The **etc/** directory contains some example configuration files, and the **share/** directory contains some supplemental documentation.

- You must be logged in with administrative (root) privileges.



NOTE

If you use a 64 bit system the mod_cluster binary web server modules will be located here: **EAP_HOME/modules/system/layers/base/native/lib64/httpd/modules**. You must use this path whenever you need access to the modules.

Procedure 17.12. Install the mod_cluster Module

1. Determine your Apache HTTP Server configuration location.

Your Apache HTTP Server configuration location will be different depending on whether you are using Red Hat Enterprise Linux's Apache HTTP Server, the standalone Apache HTTP Server included as a separate downloadable component with JBoss EAP 6, or the Apache HTTP Server available in JBoss Enterprise Web Server. It is one of the following three options, and is referred to in the rest of this task as *HTTPD_HOME*.

- Apache HTTP Server - **/etc/httpd/**
- JBoss EAP 6 Apache HTTP Server - This location is chosen by you, based on the requirements of your infrastructure.
- JBoss Enterprise Web Server Apache HTTP Server - **EWS_HOME/httpd/**

2. Copy the modules to the Apache HTTP Server modules directory.

Copy the four modules (the files ending in **.so**) from the **EAP_HOME/modules/system/layers/base/native/lib/httpd/modules** directory of the extracted Webserver Natives archive to the **HTTPD_MODULES/** directory.

3. **For JBoss Enterprise Web Server, disable the `mod_proxy_balancer` module.**

If you use JBoss Enterprise Web Server, the **mod_proxy_balancer** module is enabled by default. It is incompatible with `mod_cluster`. To disable it, edit the **HTTPD_CONF/httpd.conf** and comment out the following line by placing a **#** (hash) symbol before the line which loads the module. The line is shown without the comment and then with it, below.

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

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

Save and close the file.

4. **Configure the `mod_cluster` module.**

The Webserver Natives archive contains a sample **mod_cluster.conf** file (**EAP_HOME/modules/system/layers/base/native/etc/httpd/conf**). This file can be used as a guide or copied and edited to create a **HTTPD_CONF.D/JBoss_HTTP.conf** file.



NOTE

Using the name **JBoss_HTTP.conf** is an arbitrary convention in this document. The configuration file will be loaded, regardless of its name, if it is saved in the **conf.d/** directory with the **.conf** extension.

Add the following to your configuration file:

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

This causes Apache HTTP Server to automatically load the modules that **mod_cluster** needs in order to function.

5. **Create a proxy server listener.**

Continue editing **HTTPD_CONF.D/JBoss_HTTP.conf** and add the following minimal configuration, replacing the values in capital letters with suitable values for your environment.

```
Listen IP_ADDRESS:PORT
<VirtualHost IP_ADDRESS:PORT>
  <Location />
    Order deny,allow
    Deny from all
    Allow from *.MYDOMAIN.COM
  </Location>

  KeepAliveTimeout 60
  MaxKeepAliveRequests 0
  EnableMCPMReceive

  ManagerBalancerName mycluster
```

```
ServerAdvertise On
```

```
</VirtualHost>
```

These directives create a new virtual server which listens on **IP_ADDRESS:PORT**, allows connections from **MYDOMAIN.COM**, and advertises itself as a balancer called **mycluster**. These directives are covered in detail in the documentation for Apache Web Server. To learn more about the **ServerAdvertise** and **EnableMCPMReceive** directives, and the implications of server advertisement, see [Section 17.6.5, "Configure Server Advertisement Properties for Your mod_cluster-enabled Web Server"](#).

Save the file and exit.

6. Restart the Apache HTTP Server.

The way to restart the Apache HTTP Server depends on whether you are using Red Hat Enterprise Linux's Apache HTTP Server or the Apache HTTP Server included in JBoss Enterprise Web Server. Choose one of the two methods below.

- **Red Hat Enterprise Linux 6 Apache HTTP Server**

Issue the following command:

```
[root@host]# service httpd restart
```

- **JBoss Enterprise Web Server HTTP Server**

JBoss Enterprise Web Server runs on both Red Hat Enterprise Linux and Microsoft Windows Server. The method for restarting the Apache HTTP Server is different for each.

- **Red Hat Enterprise Linux**

In Red Hat Enterprise Linux, JBoss Enterprise Web Server installs its Apache HTTP Server as a service. To restart the Apache HTTP Server, issue the following two commands:

```
[root@host ~]# service httpd stop
[root@host ~]# service httpd start
```

- **Microsoft Windows Server**

Issue the following commands in a command prompt with administrative privileges:

```
C:\> net stop httpd
C:\> net start httpd
```

Result

The Apache HTTP Server is now configured as a load balancer, and can work with the **mod_cluster** subsystem running JBoss EAP 6. To configure JBoss EAP 6 to be aware of **mod_cluster**, see [Section 17.6.6, "Configure a mod_cluster Worker Node"](#).

[Report a bug](#)

17.6.4. Install the mod_cluster Module Into Apache HTTP Server or JBoss Enterprise Web Server (RPM)

Prerequisites

- To perform this task, you must be using the Apache HTTP Server installed in Red Hat Enterprise Linux 6, JBoss Enterprise Web Server, or the standalone Apache HTTP Server included as a separate downloadable component of JBoss EAP 6.
- If you need to install Apache HTTP Server in Red Hat Enterprise Linux 6, use the instructions from the *Red Hat Enterprise Linux 6 Deployment Guide*.
- If you need to install the standalone Apache HTTP Server included as a separate downloadable component of JBoss EAP 6, refer to [Section 17.4.3, “Install Apache HTTP Server in Red Hat Enterprise Linux 5, 6, and 7 \(Zip\)”](#).
- If you need to install JBoss Enterprise Web Server, use the instructions from the *JBoss Enterprise Web Server Installation Guide*.
- You must be logged in with administrative (root) privileges.
- You must have an active subscription to the **jbappplatform-6-ARCH-server-VERS-rpm** RHN channel.

The RPM installation method is similar between Red Hat Enterprise Linux 5 and 6, only requiring minor variations for Red Hat Enterprise Linux 6 users who have Apache HTTP Server 2.2.15 installed.

1. Install the **mod_cluster-native** package using YUM:

```
yum install mod_cluster-native
```

2. **Apache HTTP Server:**

- If you choose to stay on Apache HTTP Server 2.2.15, you must disable the **mod_proxy_balancer** module loaded by default by commenting the **LoadModule proxy_balancer_module** line in the `httpd.conf` file.

Either edit the file manually or use the following command:

```
sed -i 's/^LoadModule proxy_balancer_module/#LoadModule proxy_balancer_module;/s/$/' /etc/httpd/conf/httpd.conf
```

- If you choose to upgrade to Apache HTTP Server 2.2.26, install the latest version using the following command.

```
yum install httpd
```

3. To have the Apache HTTP Server service start at boot, enter the following command:

- For Red Hat Enterprise Linux 5 and 6:

```
service httpd add
```

- For Red Hat Enterprise Linux 7:

```
systemctl enable httpd22.service
```

4. Start the `mod_cluster` balancer with the following command:

- For Red Hat Enterprise Linux 5 and 6:

```
service httpd start
```

- For Red Hat Enterprise Linux 7:

```
systemctl start httpd22.service
```

[Report a bug](#)

17.6.5. Configure Server Advertisement Properties for Your mod_cluster-enabled Web Server

Summary

For instructions on configuring your web server to interact with the mod_cluster load balancer, see [Section 17.6.3, “Install the mod_cluster Module Into Apache HTTP Server or JBoss Enterprise Web Server \(Zip\)”](#). One aspect of the configuration which needs more explanation is *server advertisement*.

When server advertisement is active, the web server broadcasts messages containing the IP address and port number specified in the mod_cluster virtual host. To configure these values, see [Section 17.6.3, “Install the mod_cluster Module Into Apache HTTP Server or JBoss Enterprise Web Server \(Zip\)”](#). If UDP multicast is not available on your network, or you prefer to configure workers with a static list of proxy servers, you can disable server advertisement and manually configure the worker nodes. See [Section 17.6.6, “Configure a mod_cluster Worker Node”](#) for information on configuring a worker.

The changes in this procedure need to be made to the **httpd.conf** associated with your Apache HTTP Server instance. This is often **/etc/httpd/conf/httpd.conf** in Red Hat Enterprise Linux, or may be in the **etc/** directory of your standalone Apache HTTP Server instance.

Procedure 17.13. Edit the httpd.conf file and implement the changes

1. **Disable the `AdvertiseFrequency` parameter, if it exists.**

If you have a line like the following in your **<VirtualHost>** statement, comment it out by putting a **#** (hash) character before the first character. The value may be different from **5**.

```
AdvertiseFrequency 5
```

2. **Add the directive to disable server advertisement.**

Add the following directive inside the **<VirtualHost>** statement, to disable server advertisement.

```
ServerAdvertise Off
```

3. **Enable the ability to receive MCPM messages.**

Add the following directive to allow the Web server to receive MCPM messages from the worker nodes.

```
EnableMCPMReceive
```

4. **Restart the Web server.**

Restart the Web server by issuing one of the following, depending on whether you use Red Hat Enterprise Linux or Microsoft Windows Server.

- **Red Hat Enterprise Linux**

```
[root@host ]# service httpd restart
```

- **Microsoft Windows Server**

```
net stop Apache2.2
net start Apache2.2
```

Result

The web server no longer advertises the IP address and port of your `mod_cluster` proxy. To reiterate, you need to configure your worker nodes to use a static address and port to communicate with the proxy. See [Section 17.6.6, “Configure a `mod_cluster` Worker Node”](#) for more details.

[Report a bug](#)

17.6.6. Configure a `mod_cluster` Worker Node

Summary

A `mod_cluster` worker node consists of a JBoss EAP 6 server. This server can be part of a server group in a Managed Domain, or a standalone server. A separate process runs within JBoss EAP 6, which manages all of the worker nodes of the cluster. This is called the master. For more conceptual information about nodes, see [Section 17.2.4, “Node types”](#). For an overview of web server load balancing, see to [Section 17.2.3, “Overview of HTTP Connectors”](#).

Worker nodes in a managed domain share an identical configuration across a server group. Worker nodes running as standalone servers are configured individually. The configuration steps are otherwise identical.

Worker Node Configuration

- A standalone server must be started with the **standalone-ha** or **standalone-full-ha** profile.
- A server group in a managed domain must use the **ha** or **full-ha** profile, and the **ha-sockets** or **full-ha-sockets** socket binding group. JBoss EAP 6 ships with a cluster-enabled server group called **other-server-group** which meets these requirements.



NOTE

Where Management CLI commands are given, they assume you use a managed domain. If you use a standalone server, remove the **/profile=full-ha** portion of the commands.

Procedure 17.14. Configure a Worker Node

1. **Configure the network interfaces.**

By default, the network interfaces all default to **127.0.0.1**. Every physical host that hosts either a standalone server or one or more servers in a server group needs its interfaces to be configured to use its public IP address, which the other servers can see.

To change the IP address of a JBoss EAP 6 host, you need to shut it down and edit its configuration file directly. This is because the Management API which drives the Management Console and Management CLI relies on a stable management address.

Follow these steps to change the IP address on each server in your cluster to the master's public IP address.

- a. Start the JBoss EAP server using the profile described earlier in this topic.
- b. Launch the Management CLI, using the ***EAP_HOME/bin/jboss-cli.sh*** command in Linux or the ***EAP_HOME/bin/jboss-cli.bat*** command in Microsoft Windows Server. Type **connect** to connect to the domain controller on the localhost, or **connect IP_ADDRESS** to connect to a domain controller on a remote server.
- c. Modify the external IP address for the **management**, **public** and **unsecure** interfaces by typing the following commands. Be sure to replace **EXTERNAL_IP_ADDRESS** in the command with the actual external IP address of the host.

```
/interface=management:write-attribute(name=inet-
address,value="{jboss.bind.address.management:EXTERNAL_IP_ADDRESS}")
/interface=public:write-attribute(name=inet-
address,value="{jboss.bind.address.public:EXTERNAL_IP_ADDRESS}")
/interface=unsecure:write-attribute(name=inet-
address,value="{jboss.bind.address.unsecure:EXTERNAL_IP_ADDRESS}")
reload
```

You should see the following result for each command.

```
"outcome" => "success"
```

- d. For hosts that participate in a managed domain but are not the master, you must change the host name from **master** to a unique name. This name must be unique across slaves and will be used for the slave to identify to the cluster, so make a note of the name you use.
 - i. Start the JBoss EAP slave host using the following syntax:

```
bin/domain.sh --host-config=HOST_SLAVE_XML_FILE_NAME
```

For example:

```
bin/domain.sh --host-config=host-slave01.xml
```

- ii. Launch the Management CLI.
- iii. Use the following syntax to replace the host name:

```
/host=master:write-attribute(name="name",value=UNIQUE_HOST_SLAVE_NAME)
```

For example:

```
/host=master:write-attribute(name="name",value="host-slave01")
```

You should see the following result.

```
"outcome" => "success"
```

This modifies the XML in the **host-slave01.xml** file as follows:

```
<host name="host-slave01" xmlns="urn:jboss:domain:1.6">
```

- e. For newly configured hosts that need to join a managed domain, you must remove the **local** element and add the **remote** element **host** attribute that points to the domain controller. This step does not apply for a standalone server.

- i. Start the JBoss EAP slave host using the following syntax:

```
bin/domain.sh --host-config=HOST_SLAVE_XML_FILE_NAME
```

For example:

```
bin/domain.sh --host-config=host-slave01.xml
```

- ii. Launch the Management CLI.
iii. Use the following syntax specify the domain controller:

```
/host=UNIQUE_HOST_SLAVE_NAME/:write-remote-domain-  
controller(host=DOMAIN_CONTROLLER_IP_ADDRESS,port=${jboss.domain.master  
.port:9999},security-realm="ManagementRealm")
```

For example:

```
/host=host-slave01/:write-remote-domain-  
controller(host="192.168.1.200",port=${jboss.domain.master.port:9999},security-  
realm="ManagementRealm")
```

You should see the following result.

```
"outcome" => "success"
```

This modifies the XML in the **host-slave01.xml** file as follows:

```
<domain-controller>  
  <remote host="192.168.1.200" port="${jboss.domain.master.port:9999}" security-  
  realm="ManagementRealm"/>  
</domain-controller>
```

2. Configure authentication for each slave server.

Each slave server needs a username and password created in the domain controller's or standalone master's **ManagementRealm**. On the domain controller or standalone master, run the **EAP_HOME/bin/add-user.sh** command. Add a user with the same username as the slave, to the **ManagementRealm**. When asked if this user will need to authenticate to an external JBoss EAP 6 instance, answer **yes**. An example of the input and output of the command is below, for a slave called **slave1**, with password **changeme**.

```
user:bin user$ ./add-user.sh
```

What type of user do you wish to add?

- a) Management User (mgmt-users.properties)
 - b) Application User (application-users.properties)
- (a): a

```

Enter the details of the new user to add.
Realm (ManagementRealm) :
Username : slave1
Password : changeme
Re-enter Password : changeme
About to add user 'slave1' for realm 'ManagementRealm'
Is this correct yes/no? yes
Added user 'slave1' to file '/home/user/jboss-eap-6.0/standalone/configuration/mgmt-
users.properties'
Added user 'slave1' to file '/home/user/jboss-eap-6.0/domain/configuration/mgmt-
users.properties'
Is this new user going to be used for one AS process to connect to another AS process e.g.
slave domain controller?
yes/no? yes
To represent the user add the following to the server-identities definition <secret
value="Y2hhbmdlbWU=" />

```

3. **Copy the Base64-encoded<secret> element from the `add-user.sh` output.**

If you plan to specify the Base64-encoded password value for authentication, copy the **<secret>** element value from the last line of the `add-user.sh` output as you will need it in the step below.

4. **Modify the slave host's security realm to use the new authentication.**

You can specify the secret value in one of the following ways:

- **Specify the Base64-encoded password value in the server configuration file using the Management CLI.**
 - a. Launch the Management CLI, using the `EAP_HOME/bin/jboss-cli.sh` command in Linux or the `EAP_HOME/bin/jboss-cli.bat` command in Microsoft Windows Server. Type **connect** to connect to the domain controller on the localhost, or **connect IP_ADDRESS** to connect to a domain controller on a remote server.
 - b. Specify the secret value by typing the following command. Be sure to replace the **SECRET_VALUE** with the secret value returned from the `add-user` output from the previous step.

```

/host=master/core-service=management/security-realm=ManagementRealm/server-
identity=secret:add(value="SECRET_VALUE")
reload --host=master

```

You should see the following result for each command.

```
"outcome" => "success"
```

- **Configure the host to get the password from the vault.**

- a. Use the `vault.sh` script to generate a masked password. It will generate a string like the following:
VAULT::secret::password::ODVmYmJjNGMtZDU2ZC00YmNILWE4ODMtZjQ1NWNmNDU4ZDc1TEIORV9CUkVBS3ZhdWx0.

You can find more information on password vaults in the *Security Architecture* and other JBoss EAP security documentation.

- b. Launch the Management CLI, using the ***EAP_HOME/bin/jboss-cli.sh*** command in Linux or the ***EAP_HOME/bin/jboss-cli.bat*** command in Microsoft Windows Server. Type **connect** to connect to the domain controller on the localhost, or **connect IP_ADDRESS** to connect to a domain controller on a remote server.
- c. Specify the secret value by typing the following command. Be sure to replace the **SECRET_VALUE** with the masked password generated in the previous step.

```
/host=master/core-service=management/security-realm=ManagementRealm/server-identity=secret:add(value="{VAULT::secret::password::SECRET_VALUE}")
reload --host=master
```

You should see the following result for each command.

```
"outcome" => "success"
```



NOTE

When creating a password in the vault, it must be specified in plain text, not Base64-encoded.

- o **Specify the password as a system property.**

The following examples use **server.identity.password** as the system property name for the password.

- a. Specify the system property for the password in the server configuration file using the Management CLI.
 - i. Launch the Management CLI, using the ***EAP_HOME/bin/jboss-cli.sh*** command in Linux or the ***EAP_HOME/bin/jboss-cli.bat*** command in Microsoft Windows Server. Type **connect** to connect to the domain controller on the localhost, or **connect IP_ADDRESS** to connect to a domain controller on a remote server.
 - ii. Type the following command to configure the secret identity to use the system property.

```
/host=master/core-service=management/security-realm=ManagementRealm/server-identity=secret:add(value="{server.identity.password}")
reload --host=master
```

You will see the following result for each command.

```
"outcome" => "success"
```

- b. When you specify the password as a system property, you can configure the host in either of the following ways:

- Start the server entering the password in plain text as a command line argument, for example:

```
-Dserver.identity.password=changeme
```

**NOTE**

The password must be entered in plain text and will be visible to anyone who issues a **ps -ef** command.

- Place the password in a properties file and pass the properties file URL as a command line argument.

- i. Add the key/value pair to a properties file. For example:

```
server.identity.password=changeme
```

- ii. Start the server with the command line arguments

```
--properties=URL_TO_PROPERTIES_FILE
```

5. **Restart the server.**

The slave will now authenticate to the master using its host name as the username and the encrypted string as its password.

Result

Your standalone server, or servers within a server group of a managed domain, are now configured as `mod_cluster` worker nodes. If you deploy a clustered application, its sessions are replicated to all cluster nodes for failover, and it can accept requests from an external Web server or load balancer. Each node of the cluster discovers the other nodes using automatic discovery, by default. To configure automatic discovery, and the other specific settings of the **mod_cluster** subsystem, see [Section 17.6.2, "Configure the mod_cluster Subsystem"](#). To configure the Apache HTTP Server, see [Section 17.4.7, "Use an External Web Server as the Web Front-end for JBoss EAP 6 Applications"](#).

[Report a bug](#)

17.6.7. Migrate Traffic between Clusters

Summary

After creating a new cluster using JBoss EAP 6, you can migrate traffic from the previous cluster to the new one as part of an upgrade process. In this task, you will see the strategy that can be used to migrate this traffic with minimal outage or downtime.

Prerequisites

- A new cluster setup: [Section 17.6.2, "Configure the mod_cluster Subsystem"](#) (we will call this cluster: ClusterNEW).
- An old cluster setup that is being made redundant (we will call this cluster: ClusterOLD).

Procedure 17.15. Upgrade Process for Clusters - Load Balancing Groups

1. Setup your new cluster using the steps described in the prerequisites.
2. In both ClusterNEW and ClusterOLD, ensure that the configuration option **sticky-session** is set to **true** (this option is set to **true** by default). Enabling this option means that all new

requests made to a cluster node in any of the clusters will continue to go to the respective cluster node.

```
/profile=full-ha/subsystem=modcluster/mod-cluster-config=configuration/:write-attribute(name=sticky-session,value=true)
```

- Assuming that all the cluster nodes in ClusterOLD are members of ClusterOLD load balancing group. One can set this configuration either via CLI or with an xml configuration (either ha or full-ha profiles in domain mode and either standalone-ha or standalone-full-ha in standalone mode):

```
/profile=full-ha/subsystem=modcluster/mod-cluster-config=configuration/:write-attribute(name=load-balancing-group,value=ClusterOLD)
```

```
<subsystem xmlns="urn:jboss:domain:modcluster:1.2">
  <mod-cluster-config load-balancing-group="ClusterOLD" advertise-socket="modcluster"
connector="ajp">
    <dynamic-load-provider>
      <load-metric type="busyness"/>
    </dynamic-load-provider>
  </mod-cluster-config>
</subsystem>
```

- Add the nodes in ClusterNEW to the mod_cluster configuration individually using the process described here: [Section 17.6.6, "Configure a mod_cluster Worker Node"](#). Additionally use the aforementioned procedure and set their load balancing group to ClusterNEW.

At this point, one can see an output similar to the undermentioned shortened example on the mod_cluster-manager console:

```
mod_cluster/<version>

LBGroup ClusterOLD: [Enable Nodes] [Disable Nodes] [Stop Nodes]
Node node-1-jvmroute (ajp://node1.oldcluster.example:8009):
  [Enable Contexts] [Disable Contexts] [Stop Contexts]
  Balancer: qacluster, LBGroup: ClusterOLD, Flushpackets: Off, ..., Load: 100
  Virtual Host 1:
    Contexts:
      /my-deployed-application-context, Status: ENABLED Request: 0 [Disable]
[Stop]

Node node-2-jvmroute (ajp://node2.oldcluster.example:8009):
  [Enable Contexts] [Disable Contexts] [Stop Contexts]
  Balancer: qacluster, LBGroup: ClusterOLD, Flushpackets: Off, ..., Load: 100
  Virtual Host 1:
    Contexts:
      /my-deployed-application-context, Status: ENABLED Request: 0 [Disable]
[Stop]

LBGroup ClusterNEW: [Enable Nodes] [Disable Nodes] [Stop Nodes]
Node node-3-jvmroute (ajp://node3.newcluster.example:8009):
```

```
[Enable Contexts] [Disable Contexts] [Stop Contexts]
Balancer: qacluster, LBGroup: ClusterNEW, Flushpackets: Off, ..., Load: 100
Virtual Host 1:
  Contexts:
    /my-deployed-application-context, Status: ENABLED Request: 0 [Disable]
```

[Stop]

```
Node node-4-jvmroute (ajp://node4.newcluster.example:8009):
[Enable Contexts] [Disable Contexts] [Stop Contexts]
Balancer: qacluster, LBGroup: ClusterNEW, Flushpackets: Off, ..., Load: 100
Virtual Host 1:
  Contexts:
    /my-deployed-application-context, Status: ENABLED Request: 0 [Disable]
```

[Stop]

- There are old active sessions within the ClusterOLD group and any new sessions are created either within the ClusterOLD or CLusterNEW group. Next, we want to disable the whole ClusterOLD group, so as we can power down its cluster nodes without causing any error to currently active client's sessions.

Click on the [Disable Nodes] link for LBGroup ClusterOLD on mod_cluster-manager web console.

From this point on, only requests belonging to already established sessions will be routed to members of ClusterOLD load balancing group. Any new client's sessions will be created in the ClusterNEW group only. As soon as there are no active sessions within ClusterOLD group, we can safely remove its members.



NOTE

Using [Stop Nodes] would command the load balancer to stop routing any requests to this domain immediately. This will force a failover to another load balancing group which will cause session data loss to clients, provided there is no session replication between ClusterNEW and ClusterOLD.

Default Load Balancing Group

In case the current ClusterOLD setup does not contain any load balancing group settings (one can see LBGroup; on mod_cluster-manager console), one can still take advantage of disabling the ClusterOLD nodes. In this case, click on [Disable Contexts] for each of the Cluster OLD nodes. Contexts of these nodes will be disabled and once there are no active sessions present, they will be ready for removal. New client's sessions will be created only on nodes with enabled contexts, presumably Cluster NEW members in this example.

Using JBoss EAP CLI

In addition to the possibility of using mod_cluster-manager web console, one can leverage CLI in order to disable a particular context. The undermentioned operation is called stop-context, but it makes the cluster node to send DISABLE-APP command to the load balancer, having exactly the same effect as clicking on [Disable] link next to a particular context on mod_cluster-manager console (note that virtual host aliases, e.g. default-host were removed from the aforementioned mod_cluster-manager console output example).

```
/profile=full-ha/subsystem=modcluster/:stop-context(context=/my-deployed-application-context,
virtualhost=default-host, waittime=50)
```

Conclusion

To stop a particular context, cluster node or a whole load balancing group means to force the balancer to stop routing any request to it immediately, thus forcing failover to another available context. To disable a particular context, cluster node or a whole load balancing group means to tell the balancer that no new sessions should be created on this particular context/node/load balancing group.

Result

You have successfully upgraded a JBoss EAP 6 Cluster.

[Report a bug](#)

17.6.8. Configure `fail_on_status` Parameter for `mod_cluster`

The `fail_on_status` parameter lists those HTTP status codes which, when returned by a worker node in a cluster, will mark that node as having failed. The load balancer will then send future requests to another worker node in the cluster. The failed worker node will remain in a **NOTOK** state until it sends the load balancer a **STATUS** message.



NOTE

The `fail_on_status` parameter cannot be used with HP-UX v11.3 hpws httpd B.2.2.15.15 from Hewlett-Packard as it does not support the feature.

The `fail_on_status` parameter must be configured in the `httpd` configuration file of your load balancer. Multiple HTTP status codes for `fail_on_status` can be specified as a comma-separated list. The following example specifies the HTTP status codes **203** and **204** for `fail_on_status`.

Example 17.6. `fail_on_status` Configuration

```
ProxyPass / balancer://MyBalancer stickysession=JSESSIONID|jsessionid nofailover=on
failonstatus=203,204
ProxyPassReverse / balancer://MyBalancer
ProxyPreserveHost on
```

[Report a bug](#)

17.7. APACHE MOD_JK

17.7.1. About the Apache `mod_jk` HTTP Connector

Apache `mod_jk` is a HTTP connector which is provided for customers who need it for compatibility purposes. It provides load balancing, and is a part of the natives package, **Red Hat JBoss Enterprise Application Platform 6.X.0 Webserver Connector Natives** (zip installation) which is available on the Red Hat Customer Portal at <https://access.redhat.com>. `mod_jk` can be installed from the RPMs. For details of installing from RPM, refer [Section 17.7.4, "Install the `mod_jk` Module Into the Apache HTTP Server \(RPM\)"](#). For supported platforms, see <https://access.redhat.com/articles/111663>. The `mod_jk` connector is maintained by Apache, and its documentation is located at <http://tomcat.apache.org/connectors-doc/>.

JBoss EAP 6 can accept workloads from an Apache HTTP proxy server. The proxy server accepts client requests from the web front-end, and passes the work to participating JBoss EAP 6 servers. If sticky

sessions are enabled, the same client request always goes to the same JBoss EAP 6 server, unless the server is unavailable.

Unlike the JBoss **mod_cluster** HTTP connector, an Apache **mod_jk** HTTP connector does not know the status of deployments on servers or server groups, and cannot adapt where it sends its work accordingly.

mod_jk communicates over the AJP 1.3 protocol. **mod_cluster** supports other protocols. For more information, refer Table HTTP connector features and constraints in [Section 17.2.3, “Overview of HTTP Connectors”](#).



NOTE

mod_cluster is a more advanced load balancer than **mod_jk**. **mod_cluster** provides all of the functionality of **mod_jk** and additional features. For more information about **mod_cluster**, see [Section 17.6.1, “About the **mod_cluster** HTTP Connector”](#).

Next step: Configure a JBoss EAP 6 instance to participate in a **mod_jk load balancing group**

- [Section 17.4.8, “Configure JBoss EAP 6 to Accept Requests From External Web Servers”](#)
- [Section 17.7.3, “Install the **mod_jk** Module Into the Apache HTTP Server \(ZIP\)”](#)

[Report a bug](#)

17.7.2. Configure JBoss EAP 6 to Communicate with Apache **mod_jk**

Overview

The **mod_jk** HTTP connector has a single component, the **mod_jk.so** module loaded by the web server. This module receives client requests and forwards them to the container, in this case JBoss EAP 6. JBoss EAP 6 must also be configured to accept these requests and send replies back to the web server.

Configuring the Apache HTTP Server is covered in [Section 17.7.3, “Install the **mod_jk** Module Into the Apache HTTP Server \(ZIP\)”](#).

In order for JBoss EAP 6 to be able to communicate with the Apache HTTP server, it must have the AJP/1.3 connector enabled. This connector is present by default in the following configurations:

- In a managed domain, in server groups using the **ha** and **full-ha** profiles, and the **ha** or **full-ha** socket binding group. The **other-server-group** server group is configured correctly in a default installation.
- In a standalone server, the **standalone-ha** and **standalone-full-ha** profiles are configured for clustered configurations. To start the standalone server with one of these profiles, issue the following command, from the **EAP_HOME/** directory. Substitute the appropriate profile name.

```
EAP_HOME/bin/standalone.sh --server-config=standalone-ha.xml
```

For Windows, enter the following command:

```
EAP_HOME\bin\standalone.bat --server-config=standalone-ha.xml
```

[Report a bug](#)

17.7.3. Install the mod_jk Module Into the Apache HTTP Server (ZIP)

Prerequisites

- To perform this task, you must be using Apache HTTP Server installed on a supported environment or the Apache HTTP Server installed from JBoss Enterprise Web Server. Note that the JBoss Enterprise Web Server is part of the JBoss EAP 6 distribution.
- If you need to install the Red Hat Enterprise Linux native Apache HTTP Server, use the instructions in the *Red Hat Enterprise Linux Deployment Guide*.
- If you need to install the HP-UX native Apache HTTP Server, use the instructions in the *HP-UX Web Server Suite Installation Guide*, available at <https://h20392.www2.hp.com/portal/swdepot/displayInstallInfo.do?productNumber=HPUXWSATW232>.
- If you need to install JBoss Enterprise Web Server, use the instructions in the *JBoss Enterprise Web Server Installation Guide*.
- If you are using Apache HTTP Server, download the JBoss EAP 6 Native Components package for your platform from the Red Hat Customer Portal at <https://access.redhat.com>. This package contains both the **mod_jk** and **mod_cluster** precompiled binaries. If you are using JBoss Enterprise Web Server, it already includes the binary for **mod_jk**.
- If you are using Red Hat Enterprise Linux (RHEL) 5 and native Apache HTTP server (httpd 2.2.3), load the mod_perl module prior to loading mod_jk module.
- You must be logged in with administrative (root) privileges.
- To view the HTTPD variable conventions, see [Section 17.4.2, "HTTPD Variable Conventions"](#)

Procedure 17.16. Install the mod_jk Module

1. Configure the mod_jk module.

- Create a new file called ***HTTPD_HOME/conf.d/mod-jk.conf*** and add the following to it:



NOTE

The **JkMount** directive specifies which URLs Apache HTTP Server must forward to the mod_jk module. Based on the directive's configuration, mod_jk sends the received URL to the correct workers.

To serve static content directly, and only use the load balancer for Java applications, the URL path must be ***/application/****. To use mod_jk as a load balancer, use the value ***/****, to forward all URLs to mod_jk.

```
# 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/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 -ForwardDirectories

# JkRequestLogFormat
JkRequestLogFormat "%w %V %T"

# Mount your applications
# The default setting only sends Java application data to mod_jk.
# Use the commented-out line to send all URLs through mod_jk.
# JkMount /* loadbalancer
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
Order deny,allow
Deny from all
Allow from 127.0.0.1
</Location>
```

Look over the values and ensure they are reasonable for your setup. When you are satisfied, save the file.

b. **Specify a JKMountFile directive**

In addition to the JKMount directive in the **mod-jk.conf**, you can specify a file which contains multiple URL patterns to be forwarded to mod_jk.

- i. Add the following to the **HTTPD_HOME/conf/mod-jk.conf** file:

```
# You can 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/uriworkermap.properties
```

- ii. Create a new file called **HTTPD_CONF/uriworkermap.properties**, with a line for each URL pattern to be matched. The following example shows examples of the syntax of the file.

```
# Simple worker configuration file
/*=loadbalancer
```

c. Copy the `mod_jk.so` file to the httpd's modules directory**NOTE**

This is only necessary if the Apache HTTP server does not have `mod_jk.so` in its `modules/` directory. You can skip this step if you are using the Apache HTTP server included as a download as part of JBoss EAP 6.

Extract the Native Web Server Connectors Zip package. Locate the `mod_jk.so` file in either the `EAP_HOME/modules/system/layers/base/native/lib/httpd/modules/` or the `EAP_HOME/modules/system/layers/base/native/lib64/httpd/modules/` directories, depending on whether your operating system is 32-bit or 64-bit.

Copy the file to the `HTTPD_MODULES/` directory.

2. Configure the `mod_jk` worker nodes.

- a. Create a new file called `HTTPD_CONF/workers.properties`. Use the following example as your starting point, and modify the file to suit your needs.

```
# 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
```

For a detailed description of the syntax of the `workers.properties` file, and advanced configuration options, see [Section 17.7.5, "Configuration Reference for Apache mod_jk Workers"](#).

3. Restart the Web Server.

The way to restart the web server depends on whether you are using Red Hat Enterprise Linux's Apache HTTP server or the Apache HTTP server included in JBoss Enterprise Web Server. Choose one of the following methods.

- **Red Hat Enterprise Linux's Apache HTTP Server**

Issue the following command:

```
[root@host]# service httpd restart
```

- **JBoss Enterprise Web Server Apache HTTP Server**

JBoss Enterprise Web Server runs on both Red Hat Enterprise Linux and Microsoft Windows Server. The method for restarting the web server is different for each.

- **Red Hat Enterprise Linux, installed from RPM**

In Red Hat Enterprise Linux, JBoss Enterprise Web Server installs its web server as a service. To restart the web server, issue the following two commands:

```
[root@host ~]# service httpd stop
[root@host ~]# service httpd start
```

- **Red Hat Enterprise Linux, installed from Zip**

If you have installed the JBoss Enterprise Web Server Apache HTTP server from a Zip archive, use the **apachectl** command to restart the web server. Replace *EWS_HOME* with the directory where you unzipped JBoss Enterprise Web Server Apache HTTP server.

```
[root@host ~]# EWS_HOME/httpd/sbin/apachectl restart
```

- **Microsoft Windows Server**

Issue the following commands in a command prompt with administrative privileges:

```
net stop Apache2.2
net start Apache2.2
```

- **Solaris**

Issue the following commands in a command prompt with administrative privileges. Replace *EWS_HOME* with the directory where you unzipped JBoss Enterprise Web Server Apache HTTP server.

```
[root@host ~] EWS_HOME/httpd/sbin/apachectl restart
```

Result

The Apache HTTP server is now configured to use the `mod_jk` load balancer. To configure JBoss EAP 6 to be aware of `mod_jk`, see [Section 17.4.8, "Configure JBoss EAP 6 to Accept Requests From External Web Servers"](#).

[Report a bug](#)

17.7.4. Install the `mod_jk` Module Into the Apache HTTP Server (RPM)

Prerequisites

- To perform this task, you must be using the Apache HTTP Server installed in Red Hat Enterprise Linux 6, JBoss Enterprise Web Server, or the standalone Apache HTTP Server included as a separate downloadable component of JBoss EAP 6.
- You must have an active subscription to the **jbappplatform-6-ARCHITECTURE-server-RHEL_VERSION-rpm** channel.
- If you need to install Apache HTTP Server in Red Hat Enterprise Linux 6, use the instructions from the *Red Hat Enterprise Linux 6 Deployment Guide* .
- If you need to install the standalone Apache HTTP Server included as a separate downloadable component of JBoss EAP 6, refer to [Section 17.4.3, “Install Apache HTTP Server in Red Hat Enterprise Linux 5, 6, and 7 \(Zip\)”](#) .
- If you need to install JBoss Enterprise Web Server, use the instructions from the *JBoss Enterprise Web Server Installation Guide*.
- You must be logged in with administrative (root) privileges.

Procedure 17.17. Red Hat Enterprise Linux 5: mod_jk with Apache HTTP Server 2.2.3

1. Install mod_jk-ap22 1.2.37 and its dependency mod_perl from the **jbappplatform-6-ARCHITECTURE-server-5-rpm** channel:

```
yum install mod_jk
```

2. **Optional:** Copy the sample configuration files for use:

```
cp /usr/share/doc/mod_jk-ap22-1.2.37/mod_jk.conf.sample /etc/httpd/conf.d/mod_jk.conf
```

```
cp /usr/share/doc/mod_jk-ap22-1.2.37/workers.properties.sample  
/etc/httpd/conf/workers.properties
```

These files should be edited to suit your needs.

3. Start the server:

```
service httpd start
```

**NOTE**

The following error message indicates that your mod_jk module had been loaded before mod_perl was present:

```
Cannot load /etc/httpd/modules/mod_jk.so into server: /etc/httpd/modules/mod_jk.so:
undefined symbol: ap_get_server_description
```

To ensure mod_perl module is loaded before mod_jk module add the following to the **/etc/httpd/conf.d/mod_jk.conf**:

```
<IfModule !perl_module>
    LoadModule perl_module modules/mod_perl.so
</IfModule>
LoadModule jk_module modules/mod_jk.so
```

Procedure 17.18. Red Hat Enterprise Linux 5: mod_jk with JBoss EAP Apache HTTP Server 2.2.26

1. Install both mod_jk and the latest Apache HTTP Server 2.2.26 provided by the **jbappplatform-6-ARCHITECTURE-server-5-rpm** channel with this command:

```
yum install mod_jk httpd
```

2. **Optional:** Copy the sample configuration files for use:

```
cp /usr/share/doc/mod_jk-ap22-1.2.37/mod_jk.conf.sample /etc/httpd/conf.d/mod_jk.conf
```

```
cp /usr/share/doc/mod_jk-ap22-1.2.37/workers.properties.sample
/etc/httpd/conf/workers.properties
```

These files should be edited to suit your needs.

3. Start the server:

```
service httpd start
```

Procedure 17.19. Red Hat Enterprise Linux 6: mod_jk with JBoss EAP Apache HTTP Server 2.2.26

1. Install mod_jk-ap22 1.2.37 and Apache HTTP Server 2.2.26 httpd package from the **jbappplatform-6-ARCHITECTURE-server-6-rpm** channel (any existing versions will be updated):

```
yum install mod_jk httpd
```

2. **Optional:** Copy the sample configuration files for use:

```
cp /usr/share/doc/mod_jk-ap22-1.2.37/mod_jk.conf.sample /etc/httpd/conf.d/mod_jk.conf
```

```
cp /usr/share/doc/mod_jk-ap22-1.2.37/workers.properties.sample
/etc/httpd/conf/workers.properties
```

These files should be edited to suit your needs.

3. Start the server:

```
service httpd start
```

Procedure 17.20. Red Hat Enterprise Linux 6: mod_jk with Apache HTTP Server 2.2.15

1. Install mod_jk with Apache HTTP Server 2.2.15 with the following command:

```
yum install mod_jk
```

2. **Optional:** Copy the sample configuration files for use:

```
cp /usr/share/doc/mod_jk-ap22-1.2.37/mod_jk.conf.sample /etc/httpd/conf.d/mod_jk.conf
```

```
cp /usr/share/doc/mod_jk-ap22-1.2.37/workers.properties.sample  
/etc/httpd/conf/workers.properties
```

These files should be edited to suit your needs.

3. Start the server:

```
service httpd start
```

Procedure 17.21. Red Hat Enterprise Linux 7: mod_jk with JBoss EAP Apache HTTP Server 2.2.26

1. Install mod_jk-ap22 1.2.37 and Apache HTTP Server 2.2.26 httpd22 package from the **jbappplatform-6-ARCHITECTURE-server-6-rpm** channel (any existing versions will be updated):

```
yum install mod_jk
```

2. **Optional:** Copy the sample configuration files for use:

```
cp /usr/share/doc/mod_jk-ap22-1.2.37/mod_jk.conf.sample /etc/httpd22/conf.d/mod_jk.conf
```

```
cp /usr/share/doc/mod_jk-ap22-1.2.37/workers.properties.sample  
/etc/httpd22/conf/workers.properties
```

These files should be edited to suit your needs.

3. Start the server:

```
systemctl start httpd22.service
```

[Report a bug](#)

17.7.5. Configuration Reference for Apache mod_jk Workers

The **workers.properties** file defines the behavior of the workers which mod_jk passes client requests to.

In Red Hat Enterprise Linux, the file resides in `/etc/httpd/conf/workers.properties`. The `workers.properties` file defines where the different application servers are located, and the way the work load should be balanced across them.

The configuration is divided into three sections. The first section deals with global properties, which apply to all workers. The second section contains settings which apply to a specific Load Balancer. The third section contains settings which apply to a specific worker node balanced by the Load Balancer.

The general structure of a property is `worker.WORKER_NAME.DIRECTIVE`, where `WORKER_NAME` is a unique name for the worker, and `DIRECTIVE` is the setting to be applied to the worker.

Configuration reference for Apache mod_jk Load Balancers

Templates specify default per-Load Balancer settings. You can override the template within the Load Balancer settings itself. You can see an example of Load Balancer templates in [Example 17.7, “Example workers.properties file”](#).

Table 17.15. Global properties

Property	Description
worker.list	The list of Load Balancers names used by mod_jk. These Load Balancers are available to receive requests.

Table 17.16. Mandatory Directives

Property	Description
type	<p>The type of the Load Balancer. The default type is ajp13. Other possible values are ajp14, lb, status.</p> <p>For more information on these directives, refer to the Apache Tomcat Connector AJP Protocol Reference at http://tomcat.apache.org/connectors-doc/ajp/ajpv13a.html.</p>

Table 17.17. Load Balancing Directives

Property	Description
balance_workers	Specifies the worker nodes that the load balancer must manage. You can use the directive multiple times for the same load balancer. It consists of a comma-separated list of worker node names. This is set per Load Balancer, not per worker node.
sticky_session	Specifies whether requests from the same session are always routed to the same worker. The default is 1 , meaning that sticky sessions are enabled. To disable sticky sessions, set it to 0 . Sticky sessions should usually be enabled, unless all of your requests are truly stateless. This is set per Load Balancer, not per worker node.

Table 17.18. Connection Directives

Property	Description
host	The hostname or IP address of the Load Balancer. The Load Balancer must support the ajp protocol stack. The default value is localhost .
port	The port number of the remote server instance listening for defined protocol requests. The default value is 8009 , which is the default listening port for AJP13 Load Balancers. The default value for AJP14 Load Balancers is 8011 .
ping_mode	<p>The conditions under which connections are probed for network status. The probe uses an empty AJP13 packet for CPing, and expects a CPong in response. Specify the conditions by using a combination of directive flags. The flags are not separated by a comma or any white-space. The ping_mode can be any combination of C, P, I, and A.</p> <ul style="list-style-type: none"> ● C - Connect. Probe the connection one time after connecting to the server. Specify the timeout using the value of connect_timeout. Otherwise, the value of ping_timeout is used. ● P - Prepost. Probe the connection before sending each request to the server. Specify the timeout using the prepost_timeout directive. Otherwise, the value of ping_timeout is used. ● I - Interval. Probe the connection at an interval specified by connection_ping_interval, if present. Otherwise, the value of ping_timeout is used. ● A - All. A shortcut for CPI, which specifies that all connection probes are used.
ping_timeout, connect_timeout, prepost_timeout, connection_ping_interval	The timeout values for the connection probe settings above. The value is specified in milliseconds, and the default value for ping_timeout is 10000.
lbfactor	Specifies the load-balancing factor for an individual Load Balancer, and only applies to a member worker node of a load balancer. This is useful to give a more powerful server more of the work load. To give a worker 3 times the default load, set this to 3:worker.my_worker.lbfactor=3

Example 17.7. Example **workers.properties** file

```

worker.balancer1.type=lb
worker.balancer2.type=lb

worker.balancer1.sticky_sessions=1
worker.balancer1.balance_workers=node1
worker.balancer2.sticky_session=1
worker.balancer2.balance_workers=node2,node3

worker.nodetemplate.type=ajp13
worker.nodetemplate.port=8009

worker.node1.template=nodetemplate
worker.node1.host=localhost

```

```
worker.node1.ping_mode=C  
worker.node1.connection_ping_interval=9000  
worker.node1.lbfactor=1  
  
worker.node2.template=nodetemplate  
worker.node2.host=192.168.1.1  
worker.node2.ping_mode=A  
  
worker.node3.template=nodetemplate  
worker.node3.host=192.168.1.2
```

The example above demonstrates the use of multiple Load Balancers to serve the content on behalf of a web server. The reasons for such configuration can be:

- To have different contexts to be served by different Load Balancers, providing a development environment in which all the developers share the same web server but own a Load Balancer of their own.
- To have different virtual hosts served by different processes, providing a clear separation between sites belonging to different companies.
- To provide load balancing, that is, run multiple Load Balancers each on its own machine and divide the requests between them.

Further configuration details for Apache `mod_jk` are out of the scope of this document. Refer to the Apache documentation at <http://tomcat.apache.org/connectors-doc/> for further instructions.

[Report a bug](#)

17.8. APACHE MOD_PROXY

17.8.1. About the Apache `mod_proxy` HTTP Connector

Apache provides two different proxying and load balancing modules for its `httpd`: **`mod_proxy`** and **`mod_jk`**. To learn more about **`mod_jk`**, refer to [Section 17.7.1, “About the Apache `mod_jk` HTTP Connector”](#). JBoss EAP 6 supports use of either of these, although **`mod_cluster`**, the JBoss HTTP connector, more closely couples JBoss EAP 6 and the external `httpd`, and is the recommended HTTP connector. Refer to [Section 17.2.3, “Overview of HTTP Connectors”](#) for an overview of all supported HTTP connectors, including advantages and disadvantages.

Unlike **`mod_jk`**, **`mod_proxy`** supports connections over HTTP and HTTPS protocols. Each of them also support the AJP protocol.

`mod_proxy` can be configured in standalone or load-balanced configurations, and it supports the notion of sticky sessions.

The **`mod_proxy`** module requires JBoss EAP 6 to have the HTTP, HTTPS or AJP web connector configured. This is part of the Web subsystem. Refer to [Section 15.1, “Configure the Web Subsystem”](#) for information on configuring the Web subsystem.

**NOTE**

mod_cluster is a more advanced load balancer than **mod_proxy**. **mod_cluster** provides all of the functionality of **mod_proxy** and additional features. For more information about **mod_cluster**, see [Section 17.6.1, “About the mod_cluster HTTP Connector”](#).

[Report a bug](#)

17.8.2. Install the mod_proxy HTTP Connector into Apache HTTP Server

Overview

mod_proxy is a load-balancing module provided by Apache. This task presents a basic configuration. For more advanced configuration, or additional details, see Apache's **mod_proxy** documentation at https://httpd.apache.org/docs/2.2/mod/mod_proxy.html. For more details about **mod_proxy** from the perspective of JBoss EAP 6, see [Section 17.8.1, “About the Apache mod_proxy HTTP Connector”](#) and [Section 17.2.3, “Overview of HTTP Connectors”](#).

Prerequisites

- Apache HTTP server either from JBoss Enterprise Web Server or provided by operating system needs to be installed. A standalone Apache HTTP server is provided as a separate download in the Red Hat Customer Portal, in the JBoss EAP 6 download area. See [Section 17.4.3, “Install Apache HTTP Server in Red Hat Enterprise Linux 5, 6, and 7 \(Zip\)”](#) for information about this Apache HTTP server if you wish to use it.
- The **mod_proxy** modules need to be installed. Apache HTTP server typically comes with the **mod_proxy** modules already included. This is the case on Red Hat Enterprise Linux and the Apache HTTP Server that comes with the JBoss Enterprise Web Server.
- You need **root** or administrator privileges to modify the Apache HTTP Server configuration.
- In our example we assume that JBoss EAP 6 is configured with the HTTP or HTTPS web connector. This is part of the Web subsystem configuration. Refer to [Section 15.1, “Configure the Web Subsystem”](#) for information about configuring the Web subsystem.

1. Enable the mod_proxy modules in the httpd

Look for the following lines in your **HTTPD_CONF/httpd.conf** file. If they are not present, add them to the bottom. If they are present but the lines begin with a comment (#) character, remove the character. Save the file afterward. Usually, the modules are already present and enabled.

```
LoadModule proxy_module modules/mod_proxy.so
LoadModule proxy_balancer_module modules/mod_proxy_balancer.so
LoadModule proxy_http_module modules/mod_proxy_http.so
# Uncomment these to proxy FTP or HTTPS
#LoadModule proxy_ftp_module modules/mod_proxy_ftp.so
#LoadModule proxy_connect_module modules/mod_proxy_connect.so
```

2. Add a non-load-balancing proxy.

Add the following configuration to your **HTTPD_CONF/httpd.conf** file, directly beneath any other **<VirtualHost>** directives you may have. Replace the values with ones appropriate to your setup.

This example uses a virtual host. See the next step to use the default httpd configuration.

```

<VirtualHost *:80>
# Your domain name
ServerName Domain_NAME_HERE

ProxyPreserveHost On

# The IP and port of JBoss EAP 6
# These represent the default values, if your httpd is on the same host
# as your JBoss EAP 6 managed domain or server

ProxyPass / http://localhost:8080/
ProxyPassReverse / http://localhost:8080/

# The location of the HTML files, and access control information
DocumentRoot /var/www
<Directory /var/www>
Options -Indexes
Order allow,deny
Allow from all
</Directory>
</VirtualHost>

```

After making your changes, save the file.

3. Add a load-balancing proxy.

To use **mod_proxy** as a load balancer, and send work to multiple JBoss EAP 6 instances, add the following configuration to your **HTTPD_CONF/httpd.conf** file. The example IP addresses are fictional. Replace them with the appropriate values for your environment.

```

<Proxy balancer://mycluster>

Order deny,allow
Allow from all

# Add each JBoss Enterprise Application Server by IP address and port.
# If the route values are unique like this, one node will not fail over to the other.
BalancerMember http://192.168.1.1:8080 route=node1
BalancerMember http://192.168.1.2:8180 route=node2
</Proxy>

<VirtualHost *:80>
# Your domain name
ServerName YOUR_DOMAIN_NAME

ProxyPreserveHost On
ProxyPass / balancer://mycluster/

# The location of the HTML files, and access control information
DocumentRoot /var/www
<Directory /var/www>
Options -Indexes
Order allow,deny
Allow from all
</Directory>

</VirtualHost>

```

The examples above all communicate using the HTTP protocol. You can use AJP or HTTPS protocols instead, if you load the appropriate **mod_proxy** modules. Refer to Apache's **mod_proxy** documentation http://httpd.apache.org/docs/2.2/mod/mod_proxy.html for more details.

4. Enable sticky sessions.

Sticky sessions mean that if a client request originally goes to a specific JBoss EAP 6 worker, all future requests will be sent to the same worker, unless it becomes unavailable. This is almost always the correct behavior.

To enable sticky sessions for **mod_proxy**, add the **stickysession** parameter to the **ProxyPass** statement. This example also shows some other parameters which you can use. See Apache's **mod_proxy** documentation at http://httpd.apache.org/docs/2.2/mod/mod_proxy.html for more information on them.

```
ProxyPass /MyApp balancer://mycluster stickysession=JSESSIONID lbmethod=bytraffic  
nofailover=Off
```

5. Restart the Web Server.

Restart the web server for your changes to take effect.

Result

Your Apache HTTP server is configured to use **mod_proxy** to send client requests to JBoss EAP 6 instances, either in a standard or load-balancing configuration. To configure JBoss EAP 6 to respond to these requests, see [Section 17.4.8, "Configure JBoss EAP 6 to Accept Requests From External Web Servers"](#).

[Report a bug](#)

17.9. MICROSOFT ISAPI CONNECTOR

17.9.1. About the Internet Server API (ISAPI)

Internet Server API (ISAPI) is a set of APIs used to write OLE Server extensions and filters for Web servers such as Microsoft's Internet Information Services (IIS). **isapi_redirect.dll** is an extension of **mod_jk** adjusted to IIS. **isapi_redirect.dll** enables you to configure JBoss EAP 6 instances as a worker nodes with an IIS as load balancer.

[Report a bug](#)

17.9.2. Download and Extract Webserver Connector Natives for Microsoft IIS

1. In a web browser, navigate to the Red Hat Customer Support portal at <https://access.redhat.com>.
2. Navigate to **Downloads**, then **Red Hat JBoss Middleware Download Software**, then select **Enterprise Application Platform** from the **Product** drop-down list.
3. Select the appropriate version from the **Version** drop-down list.
4. Choose the **Download** option of either **Red Hat JBoss Enterprise Application Platform <VERSION> Webserver Connector Natives for Windows Server 2008 x86_64** or **Red Hat JBoss Enterprise Application Platform <VERSION> Webserver Connector Natives for Windows Server 2008 i686** depending on the architecture of the server.

5. Open the Zip file and copy the contents of the **jboss-eap-*<VERSION>*/modules/system/layers/base/native/sbin** directory to a location on your server. It is assumed the contents were copied to **C:\connectors**.

[Report a bug](#)

17.9.3. Configure Microsoft IIS to Use the ISAPI Connector

Prerequisites:

- [Section 17.9.2, “Download and Extract Webserver Connector Natives for Microsoft IIS”](#)



NOTE

See <https://access.redhat.com/articles/111663> for a list of supported configurations of Microsoft Windows Server and IIS.

Procedure 17.22. Configure the IIS Redirector Using the IIS Manager (IIS 7)

1. Open the IIS manager by clicking **Start** → **Run**, and typing **inetmgr**.
2. In the tree view pane at the left, expand **IIS 7**.
3. Double-click **ISAPI and CGI Registrations** to open it in a new window.
4. In the **Actions** pane, click **Add**. The **Add ISAPI or CGI Restriction** window opens.
5. Specify the following values:
 - **ISAPI or CGI Path:** **c:\connectors\isapi_redirect.dll**
 - **Description:** **jboss**
 - **Allow extension path to execute:** select the check box.
6. Click **OK** to close the **Add ISAPI or CGI Restriction** window.
7. **Define a JBoss Native virtual directory**
 - a. Right-click **Default Web Site**, and click **Add Virtual Directory**. The **Add Virtual Directory** window opens.
 - b. Specify the following values to add a virtual directory:
 - **Alias:** **jboss**
 - **Physical Path:** **C:\connectors**
 - c. Click **OK** to save the values and close the **Add Virtual Directory** window.
8. **Define a JBoss Native ISAPI Redirect Filter**
 - a. In the tree view pane, expand **Sites** → **Default Web Site**.
 - b. Double-click **ISAPI Filters**. The **ISAPI Filters Features** view appears.

- c. In the **Actions** pane, click **Add**. The **Add ISAPI Filter** window appears.
 - d. Specify the following values in the **Add ISAPI Filter** window:
 - **Filter name:** `jboss`
 - **Executable:** `C:\connectors\isapi_redirect.dll`
 - e. Click **OK** to save the values and close the **Add ISAPI Filters** window.
9. **Enable the ISAPI-dll handler**
- a. Double-click the **IIS 7** item in the tree view pane. The **IIS 7 Home Features View** opens.
 - b. Double-click **Handler Mappings**. The **Handler Mappings Features View** appears.
 - c. In the **Group by** combo box, select **State**. The **Handler Mappings** are displayed in **Enabled and Disabled Groups**.
 - d. Find **ISAPI-dll**. If it is in the **Disabled** group, right-click it and select **Edit Feature Permissions**.
 - e. Enable the following permissions:
 - Read
 - Script
 - Execute
 - f. Click **OK** to save the values, and close the **Edit Feature Permissions** window.

Result

Microsoft IIS is now configured to use the ISAPI Connector. Next, [Section 17.4.8, "Configure JBoss EAP 6 to Accept Requests From External Web Servers"](#), then [Section 17.9.4, "Configure the ISAPI Connector to Send Client Requests to JBoss EAP 6"](#) or [Section 17.9.5, "Configure the ISAPI Connector to Balance Client Requests Across Multiple JBoss EAP 6 Servers"](#).

[Report a bug](#)

17.9.4. Configure the ISAPI Connector to Send Client Requests to JBoss EAP 6

Overview

This task configures a group of JBoss EAP 6 servers to accept requests from the ISAPI connector. It does not include configuration for load-balancing or high-availability failover. If you need these capabilities, refer to [Section 17.9.5, "Configure the ISAPI Connector to Balance Client Requests Across Multiple JBoss EAP 6 Servers"](#).

This configuration is done on the IIS server, and assumes that JBoss EAP 6 is already configured as per [Section 17.4.8, "Configure JBoss EAP 6 to Accept Requests From External Web Servers"](#).

Prerequisites

- You need full administrator access to the IIS server
- [Section 17.4.8, "Configure JBoss EAP 6 to Accept Requests From External Web Servers"](#)

- [Section 17.9.3, “Configure Microsoft IIS to Use the ISAPI Connector”](#)

Procedure 17.23. Edit Property Files and Setup Redirection

1. **Create a directory to store logs, property files, and lock files.**

The rest of this procedure assumes that you are using the directory **C:\connectors** for this purpose. If you use a different directory, modify the instructions accordingly.

2. **Create the `isapi_redirect.properties` file.**

Create a new file called **C:\connectors\isapi_redirect.properties**. Copy the following contents into the file.

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

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

# Log level (debug, info, warn, error or trace)
log_level=info

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

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

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

If you do not want to use a **rewrite.properties** file, comment out the last line by placing a **#** character at the beginning of the line. See [Step 5](#) for more information.

3. **Create the `uriworkermap.properties` file**

The **uriworkermap.properties** file contains mappings between deployed application URLs and which worker handles requests to them. The following example file shows the syntax of the file. Place your **uriworkermap.properties** file into **C:\connectors**.

```
# images and css files for path /status are provided by worker01
/status=worker01
/images/*=worker01
/css/*=worker01

# Path /web-console is provided by worker02
# IIS (customized) error page is used for http errors with number greater or equal to 400
# css files are provided by worker01
/web-console/*=worker02;use_server_errors=400
/web-console/css/*=worker01

# Example of exclusion from mapping, logo.gif won't be displayed
# /web-console/images/logo.gif=*

# Requests to /app-01 or /app-01/something will be routed to worker01
/app-01/*=worker01
```

```
# Requests to /app-02 or /app-02/something will be routed to worker02
/app-02/*=worker02
```

4. Create the **workers.properties** file.

The **workers.properties** file contains mapping definitions between worker labels and server instances. The following example file shows the syntax of the file. Place this file into the **C:\connectors** directory.

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

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

# First JBoss EAP 6 server definition, port 8009 is standard port for AJP in EAP
worker.worker01.host=127.0.0.1
worker.worker01.port=8009
worker.worker01.type=ajp13

# Second JBoss EAP 6 server definition
worker.worker02.host=127.0.0.100
worker.worker02.port=8009
worker.worker02.type=ajp13
```

5. Create the **rewrite.properties** file.

The **rewrite.properties** file contains simple URL rewriting rules for specific applications. The rewritten path is specified using name-value pairs, as shown in the example below. Place this file into the **C:\connectors** directory.

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

6. Restart the IIS server.

Restart your IIS server by using the **net stop** and **net start** commands.

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

Result

The IIS server is configured to send client requests to the specific JBoss EAP 6 servers you have configured, on an application-specific basis.

[Report a bug](#)

17.9.5. Configure the ISAPI Connector to Balance Client Requests Across Multiple JBoss EAP 6 Servers

Overview

This configuration balances client requests across the JBoss EAP 6 servers you specify. If you prefer to send client requests to specific JBoss EAP 6 servers on a per-deployment basis, refer to [Section 17.9.4, "Configure the ISAPI Connector to Send Client Requests to JBoss EAP 6"](#) instead.

This configuration is done on the IIS server, and assumes that JBoss EAP 6 is already configured as per [Section 17.4.8, "Configure JBoss EAP 6 to Accept Requests From External Web Servers"](#).

Prerequisites

- Full administrator access on the IIS server.
- [Section 17.4.8, "Configure JBoss EAP 6 to Accept Requests From External Web Servers"](#)
- [Section 17.9.3, "Configure Microsoft IIS to Use the ISAPI Connector"](#)

Procedure 17.24. Balance Client Requests Across Multiple Servers

1. Create a directory to store logs, property files, and lock files.

The rest of this procedure assumes that you are using the directory **C:\connectors** for this purpose. If you use a different directory, modify the instructions accordingly.

2. Create the **isapi_redirect.properties** file.

Create a new file called **C:\connectors\isapi_redirect.properties**. Copy the following contents into the file.

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

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

# Log level (debug, info, warn, error or trace)
log_level=info

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

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

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

If you do not want to use a **rewrite.properties** file, comment out the last line by placing a **#** character at the beginning of the line. See [Step 5](#) for more information.

3. Create the **uriworkermap.properties** file.

The **uriworkermap.properties** file contains mappings between deployed application URLs and which worker handles requests to them. The following example file shows the syntax of the file, with a load-balanced configuration. The wildcard (*) character sends all requests for various URL sub-directories to the load-balancer called **router**. The configuration of the load-balancer is covered in [Step 4](#).

Place your **uriworkermap.properties** file into **C:\connectors**.

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

```

/images/*=router
/status=router
/web-console/*=router

# Example of exclusion from mapping, logo.gif won't be displayed
# /web-console/images/logo.gif=*

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

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

```

4. Create the **workers.properties** file.

The **workers.properties** file contains mapping definitions between worker labels and server instances. The following example file shows the syntax of the file. The load balancer is configured near the end of the file, to comprise workers **worker01** and **worker02**. The **workers.properties** file follows the syntax of the same file used for Apache mod_jk configuration. For more information about the syntax of the **workers.properties** file, refer to [Section 17.7.5, "Configuration Reference for Apache mod_jk Workers"](#).

Place this file into the **C:\connectors** directory.

```

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

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

worker.worker01.port=8009
worker.worker01.host=127.0.0.1
worker.worker01.type=ajp13
worker.worker01.ping_mode=A
worker.worker01.socket_timeout=10
worker.worker01.lbfactor=3

# Second EAP server definition
worker.worker02.port=8009
worker.worker02.host=127.0.0.100
worker.worker02.type=ajp13
worker.worker02.ping_mode=A
worker.worker02.socket_timeout=10
worker.worker02.lbfactor=1

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

```

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

5. Create the `rewrite.properties` file.

The **rewrite.properties** file contains simple URL rewriting rules for specific applications. The rewritten path is specified using name-value pairs, as shown in the example below. Place this file into the **C:\connectors** directory.

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

6. Restart the IIS server.

Restart your IIS server by using the **net stop** and **net start** commands.

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

Result

The IIS server is configured to send client requests to the JBoss EAP 6 servers referenced in the **workers.properties** file, spreading the load across the servers in a 1:3 ratio. This ratio is derived from the load balancing factor (**lbfactor**) assigned to each server.

[Report a bug](#)

17.10. ORACLE NSAPI CONNECTOR

17.10.1. About the Netscape Server API (NSAPI)

Netscape Server API (NSAPI) is an API provided by Oracle iPlanet Web Server (formerly Netscape Web Server) for implementing extensions to the server. These extensions are known as server plugins. This API is used in **nsapi_redirector.so** provided by JBoss EAP 6 in **Native utilities** packages. To configure this connector, refer to [Section 17.10.4, "Configure the NSAPI Connector to Balance Client Requests Across Multiple JBoss EAP 6 Servers"](#).

[Report a bug](#)

17.10.2. Configure the NSAPI Connector on Oracle Solaris

Summary

The NSAPI connector is a module that runs within Oracle iPlanet Web Server.

Prerequisites

- Your server is running Oracle Solaris 10 or greater, on either an Intel 32-bit, an Intel 64-bit, or a SPARC64 architecture.
- Oracle iPlanet Web Server 7.0.15 or later for Intel architectures, or 7.0.14 or later for SPARC architectures, is installed and configured, aside from the NSAPI connector.

- JBoss EAP 6 is installed and configured on each server which will serve as a worker. Refer to [Section 17.4.8, "Configure JBoss EAP 6 to Accept Requests From External Web Servers"](#) .
- The JBoss Native Components ZIP package is downloaded from the Customer Service Portal at <https://access.redhat.com>.

Procedure 17.25. Extract and Setup the NSAPI Connector

1. Extract the JBoss Native Components package.

The rest of this procedure assumes that the Native Components package is extracted to the `EAP_HOME` directory. For the rest of this procedure, the directory `/opt/oracle/webserver7/config/` is referred to as `IPLANET_CONFIG`. If your Oracle iPlanet configuration directory is different, modify the procedure accordingly.

2. Disable servlet mappings.

Open the `IPLANET_CONFIG/default.web.xml` file and locate the section with the heading **Built In Server Mappings**. Disable the mappings to the following three servlets, by wrapping them in XML comment characters (`<!--` and `-->`).

- default
- invoker
- jsp

The following example configuration shows the disabled mappings.

```
<!-- ===== Built In Servlet Mappings ===== -->
<!-- The servlet mappings for the built in servlets defined above. -->
<!-- The mapping for the default servlet -->
<!--servlet-mapping>
  <servlet-name>default</servlet-name>
  <url-pattern>/</url-pattern>
</servlet-mapping-->
<!-- The mapping for the invoker servlet -->
<!--servlet-mapping>
  <servlet-name>invoker</servlet-name>
  <url-pattern>/servlet/*</url-pattern>
</servlet-mapping-->
<!-- The mapping for the JSP servlet -->
<!--servlet-mapping>
  <servlet-name>jsp</servlet-name>
  <url-pattern>*.jsp</url-pattern>
</servlet-mapping-->
```

Save and exit the file.

3. Configure the iPlanet Web Server to load the NSAPI connector module.

Add the following lines to the end of the `IPLANET_CONFIG/magnus.conf` file, modifying file paths to suit your configuration. These lines define the location of the `nsapi_redirector.so` module, as well as the `workers.properties` file, which lists the workers and their properties.

```
Init fn="load-modules" funcs="jk_init,jk_service"
shlib="EAP_HOME/modules/system/layers/base/native/lib/nsapi_redirector.so" shlib_flags="
(global|now)"
```

```
Init fn="jk_init" worker_file="IPLANET_CONFIG/connectors/workers.properties"
log_level="info" log_file="IPLANET_CONFIG/connectors/nsapi.log"
shm_file="IPLANET_CONFIG/connectors/tmp/jk_shm"
```

The configuration above is for a 32-bit architecture. If you use 64-bit Solaris, change the string **lib/nsapi_redirector.so** to **lib64/nsapi_redirector.so**.

Save and exit the file.

4. Configure the NSAPI connector.

You can configure the NSAPI connector for a basic configuration, with no load balancing, or a load-balancing configuration. Choose one of the following options, after which your configuration will be complete.

- [Section 17.10.3, "Configure the NSAPI Connector to Send Client Requests to JBoss EAP 6"](#)
- [Section 17.10.4, "Configure the NSAPI Connector to Balance Client Requests Across Multiple JBoss EAP 6 Servers"](#)

[Report a bug](#)

17.10.3. Configure the NSAPI Connector to Send Client Requests to JBoss EAP 6

Overview

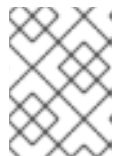
This task configures the NSAPI connector to redirect client requests to JBoss EAP 6 servers with no load-balancing or fail-over. The redirection is done on a per-deployment (and hence per-URL) basis. For a load-balancing configuration, refer to [Section 17.10.4, "Configure the NSAPI Connector to Balance Client Requests Across Multiple JBoss EAP 6 Servers"](#) instead.

Prerequisites

- You must complete [Section 17.10.2, "Configure the NSAPI Connector on Oracle Solaris"](#) before continuing with the current task.

Procedure 17.26. Setup the Basic HTTP Connector

1. Define the URL paths to redirect to the JBoss EAP 6 servers.



NOTE

In **IPLANET_CONFIG/obj.conf**, spaces are not allowed at the beginning of a line, except when the line is a continuation of the previous line.

Edit the **IPLANET_CONFIG/obj.conf** file. Locate the section which starts with **<Object name="default">**, and add each URL pattern to match, in the format shown by the example file below. The string **jknsapi** refers to the HTTP connector which will be defined in the next step. The example shows the use of wildcards for pattern matching.

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

```
NameTrans fn="assign-name" from="/nc(/*)" name="jknsapi"
NameTrans fn="assign-name" from="/jmx-console(/*)" name="jknsapi"
</Object>
```

2. Define the worker which serves each path.

Continue editing the **IPLANET_CONFIG/obj.conf** file. Add the following directly after the closing tag of the section you have just finished editing: **</Object>**.

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

The example above redirects requests to the URL path **/status** to the worker called **worker01**, and all URL paths beneath **/nc/** to the worker called **worker02**. The third line indicates that all URLs assigned to the **jknsapi** object which are not matched by the previous lines are served to **worker01**.

Save and exit the file.

3. Define the workers and their attributes.

Create a file called **workers.properties** in the **IPLANET_CONFIG/connectors/** directory. Paste the following contents into the file, and modify them to suit your environment.

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

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

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

The **workers.properties** file uses the same syntax as Apache `mod_jk`. For information about which options are available, refer to [Section 17.7.5, "Configuration Reference for Apache mod_jk Workers"](#).

Save and exit the file.

4. Restart the iPlanet Web Server.

Issue the following command to restart the iPlanet Web Server.

```
IPLANET_CONFIG/./bin/stopserv
IPLANET_CONFIG/./bin/startserv
```

Result

iPlanet Web Server now sends client requests to the URLs you have configured to deployments on JBoss EAP 6.

[Report a bug](#)

17.10.4. Configure the NSAPI Connector to Balance Client Requests Across Multiple JBoss EAP 6 Servers

Overview

This task configures the NSAPI connector to send client requests to JBoss EAP 6 servers in a load-balancing configuration. To use NSAPI connector as a simple HTTP connector with no load-balancing, see [Section 17.10.3, "Configure the NSAPI Connector to Send Client Requests to JBoss EAP 6"](#) .

Prerequisites

- [Section 17.10.2, "Configure the NSAPI Connector on Oracle Solaris"](#)

Procedure 17.27. Configure the Connector for Load-Balancing

1. Define the URL paths to redirect to the JBoss EAP 6 servers.



NOTE

In *IPLANET_CONFIG/obj.conf*, spaces are not allowed at the beginning of a line, except when the line is a continuation of the previous line.

Edit the *IPLANET_CONFIG/obj.conf* file. Locate the section which starts with **<Object name="default">**, and add each URL pattern to match, in the format shown by the example file below. The string **jknsapi** refers to the HTTP connector which will be defined in the next step. The example shows the use of wildcards for pattern matching.

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

2. Define the worker that serves each path.

Continue editing the *IPLANET_CONFIG/obj.conf* file. Directly after the closing tag for the section you modified in the previous step (**</Object>**), add the following new section and modify it to your needs:

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

This **jknsapi** object defines the worker nodes used to serve each path that was mapped to the **name="jknsapi"** mapping in the **default** object. Everything except for URLs matching **/jkmanager/*** is redirected to the worker called **router**.

3. Define the workers and their attributes.

Create a file called **workers.properties** in **IPLANET_CONFIG/connector/**. Paste the following contents into the file, and modify them to suit your environment.

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

# First JBoss EAP server
# (worker node) definition.
# Port 8009 is the standard port for AJP
#

worker.worker01.port=8009
worker.worker01.host=127.0.0.1
worker.worker01.type=ajp13
worker.worker01.ping_mode=A
worker.worker01.socket_timeout=10
worker.worker01.lbfactor=3

# Second JBoss EAP server
worker.worker02.port=8009
worker.worker02.host=127.0.0.100
worker.worker02.type=ajp13
worker.worker02.ping_mode=A
worker.worker02.socket_timeout=10
worker.worker02.lbfactor=1

# Define the load-balancer called "router"
worker.router.type=lb
worker.router.balance_workers=worker01,worker02

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

The **workers.properties** file uses the same syntax as Apache `mod_jk`. For information about which options are available, see [Section 17.7.5, "Configuration Reference for Apache mod_jk Workers"](#).

Save and exit the file.

4. Restart the iPlanet Web Server 7.0.

```
IPLANET_CONFIG/./bin/stopserv
IPLANET_CONFIG/./bin/startserv
```

Result

The iPlanet Web Server redirects the URL patterns you have configured to your JBoss EAP 6 servers in a load-balancing configuration.

[Report a bug](#)

CHAPTER 18. MESSAGING

18.1. INTRODUCTION

18.1.1. HornetQ

HornetQ is a multi-protocol, asynchronous messaging system developed by Red Hat. HornetQ provides high availability (HA) with automatic client failover to guarantee message reliability in the event of a server failure. HornetQ also supports flexible clustering solutions with load-balanced messages.

HornetQ is the Java Message Service (JMS) provider for JBoss EAP 6 and is configured as the **Messaging Subsystem**

[Report a bug](#)

18.1.2. Handling Slow HornetQ Consumers

A slow consumer with a server-side queue (e.g. JMS topic subscriber) can pose a significant problem for broker performance. If messages build up in the consumer's server-side queue then memory will begin filling up and the broker may enter paging mode which would impact performance negatively. However, criteria can be set so that consumers which don't acknowledge messages quickly enough can potentially be disconnected from the broker which in the case of a non-durable JMS subscriber would allow the broker to remove the subscription and all of its messages freeing up valuable server resources.

[Report a bug](#)

18.1.3. Handling Blocking Calls During fail-over

If the client code is in a blocking call to the server, waiting for a response to continue its execution, when fail-over occurs, the new session will not have any knowledge of the call that was in progress. This call might otherwise hang for ever, waiting for a response that will never come.

To prevent this, HornetQ will unblock any blocking calls that were in progress at the time of fail-over by making them throw a **javax.jms.JMSEXception** (if using JMS), or a **HornetQException** with error code **HornetQException.UNBLOCKED**. It is up to the client code to catch this exception and retry any operations if desired.

If the method being unblocked is a call to `commit()`, or `prepare()`, then the transaction will be automatically rolled back and HornetQ will throw a **javax.jms.TransactionRolledBackException** (if using JMS), or a **HornetQException** with error code **HornetQException.TRANSACTION_ROLLED_BACK** if using the core API.

[Report a bug](#)

18.1.4. Handling fail-over With Transactions

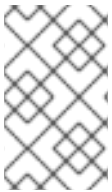
If the session is transactional and messages have already been sent or acknowledged in the current transaction, then the server cannot be sure that messages sent or acknowledgments have not been lost during the fail-over.

Consequently the transaction will be marked as rollback-only, and any subsequent attempt to commit will throw a **javax.jms.TransactionRolledBackException** (if using JMS), or a **HornetQException** with error code **HornetQException.TRANSACTION_ROLLED_BACK** if using the core API.

It is up to the user to catch the exception, and perform any client side local rollback code as necessary. There is no need to manually rollback the session - it is already rolled back. The user can then retry the transactional operations again on the same session.

If fail-over occurs when a commit call is being executed, the server, as previously described, will unblock the call to prevent a hang, since no response will come back. In this case it is not easy for the client to determine whether the transaction commit was actually processed on the live server before failure occurred.

To remedy this, the client can enable duplicate detection in the transaction, and retry the transaction operations again after the call is unblocked. If the transaction had indeed been committed on the live server successfully before fail-over, then when the transaction is retried, duplicate detection will ensure that any durable messages resent in the transaction will be ignored on the server to prevent them getting sent more than once.



NOTE

By catching the rollback exceptions and retrying, catching unblocked calls and enabling duplicate detection, once and only once delivery guarantees for messages can be provided in the case of failure, guaranteeing 100% no loss or duplication of messages.

[Report a bug](#)

18.1.5. Handling fail-over With Non Transactional Sessions

If the session is non transactional, messages or acknowledgments can be lost in the event of fail-over.

To provide once and only once delivery guarantees for non transacted sessions too, enabled duplicate detection, and catch unblock exceptions.

[Report a bug](#)

18.1.6. Getting Notified of Connection Failure

JMS provides a standard mechanism for getting notified asynchronously of connection failure: **java.jms.ExceptionListener**. For more information about ExceptionListener, refer [Oracle javax.jms Javadoc](#).

The HornetQ core API also provides a similar feature in the form of the class **org.hornet.core.client.SessionFailureListener**.

Any JMS **ExceptionListener** or Core **SessionFailureListener** instance will always be called by HornetQ in the event of connection failure, irrespective of whether the connection was successfully failed over, reconnected or reattached.

[Report a bug](#)

18.1.7. About Java Messaging Service (JMS)

Messaging systems allow you to loosely couple heterogeneous systems together with added reliability. Java Messaging Service (JMS) providers use a system of transactions, to commit or roll back changes atomically. Unlike systems based on a Remote Procedure Call (RPC) pattern, messaging systems primarily use an asynchronous message passing pattern with no tight relationship between requests and responses. Most messaging systems also support a request-response mode but this is not a primary feature of messaging systems.

Messaging systems decouple the senders of messages from the consumers of messages. The senders and consumers of messages are completely independent and know nothing of each other. This allows you to create flexible, loosely coupled systems. Often, large enterprises use a messaging system to implement a message bus which loosely couples heterogeneous systems together. Message buses often form the core of an Enterprise Service Bus (ESB). Using a message bus to decouple disparate systems can allow the system to grow and adapt more easily. It also allows more flexibility to add new systems or retire old ones since they don't have brittle dependencies on each other.

[Report a bug](#)

18.1.8. Supported Messaging Styles

HornetQ supports the following messaging styles:

Message Queue pattern

The Message Queue pattern involves sending a message to a queue. Once in the queue, the message is usually made persistent to guarantee delivery. Once the message has moved through the queue, the messaging system delivers it to a message consumer. The message consumer acknowledges the delivery of the message once it is processed.

When used with point-to-point messaging, the Message Queue pattern allows multiple consumers for a queue, but each message can only be received by a single consumer.

Publish-Subscribe pattern

The Publish-Subscribe pattern allows multiple senders to send messages to a single entity on the server. This entity is often known as a "topic". Each topic can be attended by multiple consumers, known as "subscriptions".

Each subscription receives a copy of every message sent to the topic. This differs from the Message Queue pattern, where each message is only consumed by a single consumer.

Subscriptions that are durable retain copies of each message sent to the topic until the subscriber consumes them. These copies are retained even in the event of a server restart. Non-durable subscriptions last only as long as the connection that created them.

[Report a bug](#)

18.2. CONFIGURATION OF TRANSPORTS

18.2.1. About Acceptors and Connectors

HornetQ uses the concept of connectors and acceptors as a key part of the messaging system.

Acceptors and Connectors

Acceptor

An acceptor defines which types of connections are accepted by the HornetQ server.

Connector

A connector defines how to connect to a HornetQ server, and is used by the HornetQ client.

There are two types of connectors and acceptors, relating to whether the matched connector and acceptor pair occur within same JVM or not.

Invm and Netty

Invm

Invm is short for Intra Virtual Machine. It can be used when both the client and the server are running in the same JVM.

Netty

The name of a JBoss project. It must be used when the client and server are running in different JVMs.

A HornetQ client must use a connector that is compatible with one of the server's acceptors. Only an Invm connector can connect to an Invm acceptor, and only a netty connector can connect to a netty acceptor. The connectors and acceptors are both configured on the server in a **standalone.xml** and **domain.xml**. You can use either the Management Console or the Management CLI to define them.

[Report a bug](#)

18.2.2. Configuring Netty TCP

Netty TCP is a simple unencrypted TCP sockets based transport. Netty TCP can be configured to use old blocking Java IO or non blocking Java NIO. Java NIO is recommended on the server side for better scalability with many concurrent connections. If the number of concurrent connections is less Java old IO can give better latency than NIO.

Netty TCP is not recommended for running connections across an untrusted network as it is unencrypted. With the Netty TCP transport all connections are initiated from the client side.

Example 18.1. Example of Netty TCP Configuration from Default EAP Configuration

```
<connectors>
  <netty-connector name="netty" socket-binding="messaging"/>
  <netty-connector name="netty-throughput" socket-binding="messaging-throughput">
    <param key="batch-delay" value="50"/>
  </netty-connector>
  <in-vm-connector name="in-vm" server-id="0"/>
</connectors>
<acceptors>
  <netty-acceptor name="netty" socket-binding="messaging"/>
  <netty-acceptor name="netty-throughput" socket-binding="messaging-throughput">
    <param key="batch-delay" value="50"/>
    <param key="direct-deliver" value="false"/>
  </netty-acceptor>
  <in-vm-acceptor name="in-vm" server-id="0"/>
</acceptors>
```

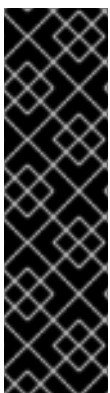
The example configuration also shows how the JBoss EAP 6 implementation of HornetQ uses socket bindings in the acceptor and connector configuration. This differs from the standalone version of HornetQ, which requires you to declare the specific hosts and ports.

The following table describes Netty TCP configuration properties:

Table 18.1. Netty TCP Configuration Properties

Property	Default	Description
batch-delay	0 milliseconds	Before writing packets to the transport, HornetQ can be configured to batch up writes for a maximum of batch-delay milliseconds. This increases the overall throughput for very small messages by increasing average latency for message transfer
direct-deliver	true	When a message arrives on the server and is delivered to waiting consumers, by default, the delivery is done on the same thread on which the message arrived. This gives good latency in environments with relatively small messages and a small number of consumers but reduces the throughput and latency. For highest throughput you can set this property as "false"
local-address	[local address available]	For a netty connector, this is used to specify the local address which the client will use when connecting to the remote address. If a local address is not specified then the connector will use any available local address
local-port	0	For a netty connector, this is used to specify which local port the client will use when connecting to the remote address. If the local-port default is used (0) then the connector will let the system pick up an ephemeral port. valid ports are 0 to 65535
nio-remoting-threads	-1	If configured to use NIO, HornetQ will, by default, use a number of threads equal to three times the number of cores (or hyper-threads) as reported by <code>Runtime.getRuntime().availableProcessors()</code> for processing incoming packets. To override this value, you can set a custom value for the number of threads

Property	Default	Description
tcp-no-delay	true	If this is true then Nagle's algorithm will be enabled. This algorithm helps improve the efficiency of TCP/IP networks by reducing the number of packets sent over a network
tcp-send-buffer-size	32768 bytes	This parameter determines the size of the TCP send buffer in bytes
tcp-receive-buffer-size	32768 bytes	This parameter determines the size of the TCP receive buffer in bytes
use-nio	false	If this is true then Java non blocking NIO will be used. If set to false then old blocking Java IO will be used. If you need the server to handle many concurrent connections use non blocking Java NIO otherwise go for old (blocking) IO
use-nio-global-worker-pool	false	This parameter will ensure all JMS connections share a single pool of Java threads (rather than each connection having its own pool). This serves to avoid exhausting the maximum number of processes on the operating system.



IMPORTANT

If you have **use-nio** set to true, use the **use-nio-global-worker-pool** parameter to minimize the risk that a machine may create a large number of connections, which could lead to an **OutOfMemory** error.

```
<netty-connector name="netty" socket-binding="messaging">
  <param key="use-nio" value="true"/>
  <param key="use-nio-global-worker-pool" value="true"/>
</netty-connector>
```



NOTE

Netty TCP properties are valid for all types of transport (Netty SSL, Netty HTTP and Netty Servlet).

[Report a bug](#)

18.2.3. Configuring Netty Secure Sockets Layer (SSL)

Netty TCP is a simple unencrypted TCP sockets based transport. Netty SSL is similar to Netty TCP but it provides enhanced security by encrypting TCP connections using the Secure Sockets Layer (SSL).



WARNING

Use of SSLv3 protocol in SSL connectors/acceptors was disallowed because of "Poodle" vulnerability. JMS clients connecting by this protocol will be refused.

The following example shows Netty configuration for one way SSL:



NOTE

Most of the following parameters can be used with acceptors as well as connectors. However some parameters work only with acceptors. The parameter description explains the difference between using these parameters in connectors and acceptors.

```
<acceptors>
  <netty-acceptor name="netty" socket-binding="messaging"/>
    <param key="ssl-enabled" value="true"/>
    <param key="key-store-password" value="[keystore password]"/>
    <param key="key-store-path" value="[path to keystore file]"/>
  </netty-acceptor>
</acceptors>
```

Table 18.2. Netty SSL Configuration Properties

Property Name	Default	Description
ssl-enabled	true	This enables SSL
key-store-password	[keystore password]	When used on an acceptor this is the password for the server side keystore. When used on a connector this is the password for the client-side keystore. This is only relevant for a connector if you are using two way SSL (mutual authentication). This value can be configured on the server, but it is downloaded and used by the client.

Property Name	Default	Description
key-store-path	[path to keystore file]	<p>When used on an acceptor this is the path to the SSL key store on the server which holds the server's certificates (whether self-signed or signed by an authority).</p> <p>When used on a connector this is the path to the client-side SSL key store which holds the client certificates. This is only relevant for a connector if you are using 2-way SSL (i.e. mutual authentication). Although this value is configured on the server, it is downloaded and used by the client.</p>

If you are configuring Netty for two way SSL (mutual authentication between server and client), there are three additional parameters in addition to the ones described in the above example for one way SSL:

- **need-client-auth:** This specifies the need for two way (mutual authentication) for client connections.
- **trust-store-password:** When used on an acceptor this is the password for the server side trust store. When used on a connector this is the password for the client side trust store. This is relevant for a connector for both one way and two way SSL. This value can be configured on the server, but it is downloaded and used by the client
- **trust-store-path:** When used on an acceptor this is the path to the server side SSL trust store that holds the keys of all the clients that the server trusts. When used on a connector this is the path to the client side SSL key store which holds the public keys of all the servers that the client trusts. This is relevant for a connector for both one way and two way SSL. This path can be configured on the server, but it is downloaded and used by the client.

[Report a bug](#)

18.2.4. Configuring Netty HTTP

Netty HTTP tunnels packets over the HTTP protocol. It can be useful in scenarios where firewalls allow only HTTP traffic to pass. Netty HTTP uses the same properties as Netty TCP along with some following additional properties:



NOTE

The following parameters can be used with acceptors as well as connectors. Netty HTTP transport does not allow the reuse of standard HTTP port (8080 by default). The use of standard HTTP port results in an exception. You can use [Section 18.2.5, "Configuring Netty Servlet"](#) (Netty Servlet Transport) for tunneling HornetQ connections through standard HTTP port.

```
<socket-binding name="messaging-http" port="7080" />
```

```

<acceptors>
  <netty-acceptor name="netty" socket-binding="messaging-http">
    <param key="http-enabled" value="false"/>
    <param key="http-client-idle-time" value="500"/>
    <param key="http-client-idle-scan-period" value="500"/>
    <param key="http-response-time" value="10000"/>
    <param key="http-server-scan-period" value="5000"/>
    <param key="http-requires-session-id" value="false"/>
  </netty-acceptor>
</acceptors>

```

The following table describes the additional properties for configuring Netty HTTP:

Table 18.3. Netty HTTP Configuration Properties

Property Name	Default	Description
http-enabled	false	If this is true HTTP is enabled
http-client-idle-time	500 milliseconds	How long a client can be idle before sending an empty HTTP request to keep the connection alive
http-client-idle-scan-period	500 milliseconds	How often (milliseconds) to scan for idle clients
http-response-time	10000 milliseconds	The time period for which the server can wait before sending an empty HTTP response to keep the connection alive
http-server-scan-period	5000 milliseconds	How often, in milliseconds, to scan for clients needing responses
http-requires-session-id	false	If this is true then client will wait after the first call to receive a session ID



WARNING

Automatic client failover is not supported for clients connecting through Netty HTTP transport.

[Report a bug](#)

18.2.5. Configuring Netty Servlet

The servlet transport allows HornetQ traffic to be tunneled over HTTP to a servlet running in a servlet engine which then redirects it to an in-VM HornetQ server. Netty HTTP transport acts as a web server listening for HTTP traffic on specific ports. With the servlet transport HornetQ traffic is proxied through a servlet engine which may already be serving web site or other applications.

In order to configure a servlet engine to work the Netty Servlet transport you need to follow these steps:

- Deploy the servlet: The following example describes a web application that uses the servlet:

```
<web-app>
  <servlet>
    <servlet-name>HornetQServlet</servlet-name>
    <servlet-class>org.jboss.netty.channel.socket.http.HttpTunnelingServlet</servlet-class>
    <init-param>
      <param-name>endpoint</param-name>
      <param-value>local:org.hornetq</param-value>
    </init-param>
    <load-on-startup>1</load-on-startup>
  </servlet>

  <servlet-mapping>
    <servlet-name>HornetQServlet</servlet-name>
    <url-pattern>/HornetQServlet</url-pattern>
  </servlet-mapping>
</web-app>
```

The init parameter **endpoint** specifies the host attribute of the Netty acceptor that the servlet will forward its packets to

- Insert the Netty servlet acceptor on the server side configuration: The following example shows the definition of an acceptor in server configuration files (**standalone.xml** and **domain.xml**):

```
<acceptors>
  <acceptor name="netty-servlet">
    <factory-class>
      org.hornetq.core.remoting.impl.netty.NettyAcceptorFactory
    </factory-class>
    <param key="use-servlet" value="true"/>
    <param key="host" value="org.hornetq"/>
  </acceptor>
</acceptors>
```

- The last step is to define a connector for the client in server configuration files (**standalone.xml** and **domain.xml**):

```
<netty-connector name="netty-servlet" socket-binding="http">
  <param key="use-servlet" value="true"/>
  <param key="servlet-path" value="/messaging/HornetQServlet"/>
</netty-connector>
```

- It is also possible to use the servlet transport over SSL by adding the following configuration to the connector:

```
<netty-connector name="netty-servlet" socket-binding="https">
```

```

<param key="use-servlet" value="true"/>
<param key="servlet-path" value="/messaging/HornetQServlet"/>
<param key="ssl-enabled" value="true"/>
<param key="key-store-path" value="path to a key-store"/>
<param key="key-store-password" value="key-store password"/>
</netty-connector>

```



WARNING

Automatic client failover is not supported for clients connecting through HTTP tunneling servlet.



NOTE

Netty servlet cannot be used to configure EAP 6 servers in order to set up a HornetQ cluster.

[Report a bug](#)

18.3. DEAD CONNECTION DETECTION

18.3.1. Closing Dead Connection Resources on the Server

A HornetQ core or JMS client application must close its resources before it exits. You can configure your application to automatically close its resources by using the **finally** block in the application's code.

The following example shows a core client application which closes its session and session factory in a **finally** block:

```

ServerLocator locator = null;
ClientSessionFactory sf = null;
ClientSession session = null;

try
{
    locator = HornetQClient.createServerLocatorWithoutHA(..);

    sf = locator.createClientSessionFactory();

    session = sf.createSession(...);

    ... do some operations with the session...
}

finally
{
    if (session != null)
    {
        session.close();
    }
}

```

```

    }

    if (sf != null)
    {
        sf.close();
    }

    if(locator != null)
    {
        locator.close();
    }
}

```

The following example shows a JMS client application which closes its session and session factory in a **finally** block:

```

Connection jmsConnection = null;

try
{
    ConnectionFactory jmsConnectionFactory =
HornetQJMSClient.createConnectionFactoryWithoutHA(...);

    jmsConnection = jmsConnectionFactory.createConnection();

    ... do some operations with the connection...
}
finally
{
    if (connection != null)
    {
        connection.close();
    }
}

```

Using Connection Time to Live (TTL) Parameter

The **connection-ttl** parameter determines the time period for which the server keeps the connection alive when it does not receive data or ping packets from the client. This parameter ensures that dead server resources like old sessions are sustained longer thereby allowing clients to reconnect when a failed network connection recovers.

You can define connection TTL for JMS clients by specifying **connection-ttl** parameter in **HornetQConnectionFactory** instance. If you are deploying JMS connection factory instances direct into JNDI; you can define **connection-ttl** parameter in **standalone.xml** and **domain.xml** server configuration files.

The default value of **connection-ttl** parameter is 60000 milliseconds. If you do not need clients to specify their own connection TTL; you can define the **connection-ttl-override** parameter in server configuration files to override all values. The **connection-ttl-override** parameter is disabled by default and has a value of -1.

Garbage Collection

HornetQ uses garbage collection to detect and close the sessions which are not explicitly closed in a **finally** block. HornetQ server logs a warning similar to the warning shown below before closing the sessions:

```
[Finalizer] 20:14:43,244 WARNING [org.hornetq.core.client.impl.DelegatingSession] I'm closing a
ClientSession you left open. Please make sure you close all ClientSessions explicitly before let
ting them go out of scope!
[Finalizer] 20:14:43,244 WARNING [org.hornetq.core.client.impl.DelegatingSession] The session you
didn't close was created here:
java.lang.Exception
  at org.hornetq.core.client.impl.DelegatingSession.<init>(DelegatingSession.java:83)
  at org.acme.yourproject.YourClass (YourClass.java:666)
```

The log message contains information about the code part where a JMS connection or user session was created and not closed later.

[Report a bug](#)

18.3.2. Detecting Client Side Failure

The client application automatically sends ping packets to the server to prevent the client from shutting down. In a similar way, the client application considers the connection alive as long as it receives data from the server.

If the client does not receive data packets from the server for a time period specified by ***client-failure-check-period*** parameter then the client considers that the connection has failed. The client then initiates a failover or calls **FailureListener** instances.

For JMS clients, client failure check period is configured using ***ClientFailureCheckPeriod*** attribute on **HornetQConnectionFactory** instance. If you are deploying JMS connection factory instances directly into JNDI on the server side, you can specify ***client-failure-check-period*** parameter in **standalone.xml** and **domain.xml** server configuration files.

The default value for client failure check period is 30000 milliseconds. A value of -1 means that the client will never close the connection if no data is received from the server.

Configuring Asynchronous Connection Execution

By default, packets received on the server side are executed on the remoting thread. It is possible to free up the remoting thread by processing operations asynchronously on any thread from the thread pool. You can configure asynchronous connection execution using ***async-connection-execution-enabled*** parameter in **standalone.xml** and **domain.xml** server configuration files. The default value of this parameter is "true".



NOTE

If you process operations asynchronously on any thread from the thread pool, it adds a little latency. Short running operations are always handled on the remoting thread for performance reasons.

[Report a bug](#)

18.4. WORK WITH LARGE MESSAGES

18.4.1. Work with Large Messages

HornetQ supports the use of large messages even when either the client or server has limited amounts of memory. Large messages can be streamed as they are, or compressed further for more efficient

transferral. A user can send a large message by setting an **InputStream** in the body of the message. When the message is sent HornetQ reads this **InputStream** and transmits data to the server in fragments.

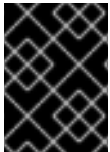
The client or the server never store the complete body of a large message in memory. The consumer initially receives a large message with an empty body and thereafter sets an **OutputStream** on the message to stream it in fragments to a disk file.

[Report a bug](#)

18.4.2. Configuring HornetQ Large Messages

Configuring the Server

In Standalone mode large messages are stored in **EAP_HOME/standalone/data/largemessages** directory. In Domain mode large messages are stored in **EAP_HOME/domain/servers/SERVERNAME/data/largemessages** directory. The configuration property **large-messages-directory** indicates the location where large messages are stored.



IMPORTANT

To achieve best performance, we recommend storing the large messages directory on a different physical volume to the message journal or the paging directory

[Report a bug](#)

18.4.3. Configuring Parameters

You can configure HornetQ large messages by setting various parameters:

Using HornetQ Core API on Client Side

If you are using HornetQ Core API on client side you need to set **ServerLocator.setMinLargeMessageSize** parameter to specify minimum size of large messages. The minimum size of large messages(**min-large-message-size**) is set to 100KiB by default.

```
ServerLocator locator = HornetQClient.createServerLocatorWithoutHA(new
TransportConfiguration(NettyConnectorFactory.class.getName()))

locator.setMinLargeMessageSize(25 * 1024);

ClientSessionFactory factory = HornetQClient.createClientSessionFactory();
```

Configuring server for Java Messaging Service (JMS) clients

If you using Java Messaging Service (JMS) you need to specify the minimum size of large messages in the attribute **min-large-message-size** of your server configuration files (**standalone.xml** and **domain.xml**). The minimum size of large messages(**min-large-message-size**) is set to 100KiB by default.



NOTE

The value of the attribute **min-large-message-size** should be in bytes

You may choose to compress large messages for fast and efficient transfer. All compression/de-compression operations are handled on client side. If the compressed message is smaller than **min-large-message-size**, it is sent to the server as a regular message. Using Java Messaging Service (JMS) you can compress large messages by setting the boolean property **compress-large-messages** "true" on the server locator or ConnectionFactory.

```
<connection-factory name="ConnectionFactory">
  <connectors>
    <connector-ref connector-name="netty"/>
  </connectors>
  ...
  <min-large-message-size>204800</min-large-message-size>
  <compress-large-messages>true</compress-large-messages>
</connection-factory>
```

[Report a bug](#)

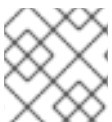
18.5. PAGING

18.5.1. About Paging

HornetQ supports many message queues with each queue containing millions of messages. The HornetQ server runs with limited memory thereby making it difficult to store all message queues in memory at one time.

Paging is a mechanism used by the HornetQ server to transparently page messages in and out of memory on need basis in order to accommodate large message queues in a limited memory.

HornetQ starts paging messages to disk, when the size of messages in memory for a particular address exceeds the maximum configured message size.



NOTE

HornetQ paging is enabled by default.

[Report a bug](#)

18.5.2. Page Files

There is an individual folder for each address on the file system which stores messages in multiple files. These files which store the messages are called page files. Each file contains messages up to the maximum configured message size (**page-size-bytes**).

The system navigates the page files as needed and removes the page files as soon as all messages in the page were received by client.

**NOTE**

If a consumer has a message selector to read messages from queue, only the messages in memory that match the selector are delivered to the consumer. When the consumer acknowledges delivery of these messages, new messages are depaged and loaded into memory. For performance reasons, HornetQ does not scan paged messages to verify if they match the consumer's message selector and there is no confirmation that new depaged messages match the consumer's message selector. There may be messages that match consumer's selector on disk in page files but HornetQ does not load them into memory until another consumer reads the messages in memory and provides free space. If the free space is not available, the consumer with selector may not receive any new messages.

[Report a bug](#)

18.5.3. Configuration of Paging Folder

Global paging parameters are specified in server configuration files (standalone.xml and domain.xml). You can configure the location of the paging directory/folder by using the ***paging-directory*** parameter:

```
<hornetq-server>
...
<paging-directory>/location/paging-directory</paging-directory>
...
</hornetq-server>
```

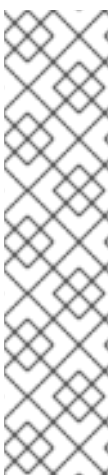
The ***paging-directory*** parameter is used to specify a location/folder to store the page files. HornetQ creates one folder for each paging address in this paging directory. The page files are stored in these folders.

The default paging directory is **EAP_HOME/standalone/data/messagingpaging** (standalone mode) and **EAP_HOME/domain/servers/SERVERNAME/data/messagingpaging** (domain mode).

[Report a bug](#)

18.5.4. Paging Mode

When messages delivered to an address exceed the configured size, that address goes into "page/paging mode".

**NOTE**

Paging is done individually per address. If you configure a ***max-size-bytes*** for an address, it means each matching address will have a maximum size that you specified. However it does not mean that the total overall size of all matching addresses is limited to ***max-size-bytes***.

Even with ***page*** mode, the server may crash due to an out-of-memory error. HornetQ keeps a reference to each page file on the disk. In a situation with millions of page files, HornetQ can face memory exhaustion. To minimize this risk, it is important to set the attribute ***page-size-bytes*** to a suitable value. You must configure the memory for your JBoss EAP 6 server higher than ***(number of destinations)*(max-size-bytes)***, otherwise an out-of-memory error can occur.

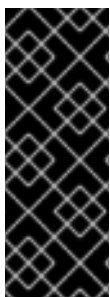
You can configure the maximum size in bytes (*max-size-bytes*) for an address in server configuration files (*standalone.xml* and *domain.xml*):

```
<address-settings>
  <address-setting match="jms.someaddress">
    <max-size-bytes>104857600</max-size-bytes>
    <page-size-bytes>10485760</page-size-bytes>
    <address-full-policy>PAGE</address-full-policy>
  </address-setting>
</address-settings>
```

The following table describes the parameters on the address settings:

Table 18.4. Paging Address Settings

Element	Default Value	Description
max-size-bytes	10485760	This is used to specify the maximum memory size the address can have before entering into paging mode
page-size-bytes	2097152	This is used to specify the size of each page file used on the paging system.
address-full-policy	PAGE	This value of this attribute is used for paging decisions. You can set either of these values for this attribute: PAGE : To enable paging and page messages beyond the set limit to disk, DROP : To silently drop messages which exceed the set limit, FAIL : To drop messages and send an exception to client message producers, BLOCK : To block client message producers when they send messages beyond the set limit
page-max-cache-size	5	The system will keep page files up to <i>page-max-cache-size</i> in memory to optimize Input/Output during paging navigation



IMPORTANT

If you don't want to page messages when the maximum size is reached, you may choose to configure an address in order to simply drop messages, drop messages with an exception on client side or block producers from sending further messages, by setting the *address-full-policy* to **DROP**, **FAIL** and **BLOCK** respectively. In the default configuration, all addresses are configured to page messages after an address reaches *max-size-bytes*.

Addresses with Multiple Queues

When a message is routed to an address that has multiple queues bound to it, there is only a single copy of the message in memory. Each queue only handles a reference to this original copy of the message. Thus the memory is freed up only when all the queues referencing the original message, have delivered the message.

**NOTE**

A single lazy queue/subscription can reduce the Input/Output performance of the entire address as all the queues will have messages being sent through an extra storage on the paging system.

[Report a bug](#)

18.6. DIVERTS

Diverts are objects configured in HornetQ; which help in diverting messages from one address (to which the message is routed) to some other address. Diverts can be configured in server configuration files (**standalone.xml** and **domain.xml**).

Diverts can be classified into the following types:

- Exclusive Divert: A message is only diverted to a new address and not sent to the old address at all
- Non-exclusive Divert: A message continues to go the old address, and a copy of it is also sent to the new address. Non-exclusive diverts can be used for splitting the flow of messages

Diverts can be configured to apply a **Transformer** and an optional message filter. An optional message filter helps only divert messages which match the specified filter. A transformer is used for transforming messages to another form. When a transformer is specified; all diverted messages are transformed by the **Transformer**.

A divert only diverts a message to an address within the same server. If you need to divert a message to an address on a different server, you can follow the pattern described below:

- Divert messages to a local store and forward queue. Setup a bridge which consumes from that queue and directs messages to an address on a different server

You can combine diverts with bridges to create various routings.

[Report a bug](#)

18.6.1. Exclusive Divert

An exclusive divert; diverts all messages from an old address to a new address. Matching messages are not routed to the old address at all. You can enable exclusive divert by setting **exclusive** attribute as **true** in **standalone.xml** and **domain.xml** server configuration files.

The following example shows an exclusive divert configured in server configuration file(s):

```
<divert name="prices-divert">
  <address>jms.topic.priceUpdates</address>
  <forwarding-address>jms.queue.priceForwarding</forwarding-address>
  <filter string="office='New York'"/>
  <transformer-class-name>
    org.hornetq.jms.example.AddForwardingTimeTransformer
  </transformer-class-name>
  <exclusive>true</exclusive>
</divert>
```

The following list describes the attributes used in the above example:

- **address**: Messages sent to this address are diverted to another address
- **forwarding-address**: Messages are diverted to this address from the old address
- **filter-string**: Messages which match the **filter-string** value are diverted. All other messages are routed to the normal address
- **transformer-class-name**: If you specify this parameter; it executes transformation for each matching message. This allows you to change a message's body or property before it is diverted
- **exclusive**: Used to enable or disable exclusive divert

[Report a bug](#)

18.6.2. Non-exclusive Divert

Non-exclusive diverts forward a copy of the original message to the new address. The original message continues to arrive at the old address. You can configure non-exclusive diverts by setting **exclusive** property as false in **standalone.xml** and **domain.xml** server configuration files.

The following example shows a non-exclusive divert:

```
<divert name="order-divert">
  <address>jms.queue.orders</address>
  <forwarding-address>jms.topic.spyTopic</forwarding-address>
  <exclusive>>false</exclusive>
</divert>
```

The above example makes a copy of every message sent to **jms.queue.orders** address and sends it to **jms.topic.spyTopic** address.

[Report a bug](#)

18.7. THE CLIENT CLASSPATH

HornetQ requires several jars on the Client Classpath depending on whether the client uses HornetQ Core API, JMS, or JNDI.



WARNING

All the jars mentioned here can be found in the **EAP_HOME/bin/client** directory of the HornetQ distribution. Be sure you only use the jars from the correct version of the release, you must not mix and match versions of jars from different HornetQ versions. Mixing and matching different jar versions may cause subtle errors and failures to occur.

To set the client classpath, include **EAP_HOME/bin/client/jboss-client.jar**. You can also use Maven dependency settings as described in the **EAP_HOME/bin/client/README-EJB-JMS.txt**.

[Report a bug](#)

18.8. CONFIGURATION

18.8.1. Configure the JMS Server

To configure the JMS Server for HornetQ, edit the server configuration file. The server configuration is contained in the **EAP_HOME/domain/configuration/domain.xml** file for domain servers, or in the **EAP_HOME/standalone/configuration/standalone-full.xml** file for standalone servers.

The **<subsystem xmlns="urn:jboss:domain:messaging:1.4">** element in the server configuration file contains all JMS configuration. Add any JMS **ConnectionFactory**, **Queue**, or **Topic** instances required for the JNDI.

1. **Enable the JMS subsystem in JBoss EAP 6.**

Within the **<extensions>** element, verify that the following line is present and is not commented out:

```
<extension module="org.jboss.as.messaging"/>
```

2. **Add the basic JMS subsystem.**

If the Messaging subsystem is not present in your configuration file, add it.

- a. Look for the **<profile>** which corresponds to the profile you use, and locate its **<subsystems>** tag.
- b. Paste the following XML immediately following the **<profile>** tag.

```
<subsystem xmlns="urn:jboss:domain:messaging:1.4">
  <hornetq-server>
    <!-- ALL XML CONFIGURATION IS ADDED HERE -->
  </hornetq-server>
</subsystem>
```

All further configuration will be added to the empty line above.

3. **Add basic configuration for JMS.**

Add the following XML in the blank line after the **<subsystem xmlns="urn:jboss:domain:messaging:1.4"><hornetq-server>** tag:

```
<journal-min-files>2</journal-min-files>
<journal-type>NIO</journal-type>
<persistence-enabled>true</persistence-enabled>
```

Customize the values above to meet your needs.

**WARNING**

The value of **journal-file-size** must be higher than or equal to **min-large-message-size** (100KiB by default), or the server won't be able to store the message.

4. Add connection factory instances to HornetQ

The client uses a JMS **ConnectionFactory** object to make connections to the server. To add a JMS connection factory object to HornetQ, include a single **<jms-connection-factories>** tag and **<connection-factory>** element for each connection factory as follows:

```
<jms-connection-factories>
  <connection-factory name="InVmConnectionFactory">
    <connectors>
      <connector-ref connector-name="in-vm"/>
    </connectors>
    <entries>
      <entry name="java:/ConnectionFactory"/>
    </entries>
  </connection-factory>
  <connection-factory name="RemoteConnectionFactory">
    <connectors>
      <connector-ref connector-name="netty"/>
    </connectors>
    <entries>
      <entry name="java:jboss/exported/jms/RemoteConnectionFactory"/>
    </entries>
  </connection-factory>
  <pooled-connection-factory name="hornetq-ra">
    <transaction mode="xa"/>
    <connectors>
      <connector-ref connector-name="in-vm"/>
    </connectors>
    <entries>
      <entry name="java:/JmsXA"/>
    </entries>
  </pooled-connection-factory>
</jms-connection-factories>
```

5. Configure the netty connectors and acceptors

This JMS connection factory uses **netty** acceptors and connectors. These are references to connector and acceptor objects deployed in the server configuration file. The connector object defines the transport and parameters used to connect to the HornetQ server. The acceptor object identifies the type of connections accepted by the HornetQ server.

To configure the **netty** connectors, include the following settings:

```
<connectors>
  <netty-connector name="netty" socket-binding="messaging"/>
  <netty-connector name="netty-throughput" socket-binding="messaging-throughput">
```

```

    <param key="batch-delay" value="50"/>
  </netty-connector>
  <in-vm-connector name="in-vm" server-id="0"/>
</connectors>

```

To configure the **netty** acceptors, include the following settings:

```

<acceptors>
  <netty-acceptor name="netty" socket-binding="messaging"/>
  <netty-acceptor name="netty-throughput" socket-binding="messaging-throughput">
    <param key="batch-delay" value="50"/>
    <param key="direct-deliver" value="false"/>
  </netty-acceptor>
  <in-vm-acceptor name="in-vm" server-id="0"/>
</acceptors>

```

6. Review the configuration

If you have followed the previous steps, your messaging subsystem should look like the following:

```

<subsystem xmlns="urn:jboss:domain:messaging:1.4">
  <hornetq-server>
    <journal-min-files>2</journal-min-files>
    <journal-type>NIO</journal-type>
    <persistence-enabled>true</persistence-enabled>
    <jms-connection-factories>
      <connection-factory name="InVmConnectionFactory">
        <connectors>
          <connector-ref connector-name="in-vm"/>
        </connectors>
        <entries>
          <entry name="java:/ConnectionFactory"/>
        </entries>
      </connection-factory>
      <connection-factory name="RemoteConnectionFactory">
        <connectors>
          <connector-ref connector-name="netty"/>
        </connectors>
        <entries>
          <entry name="java:jboss/exported/jms/RemoteConnectionFactory"/>
        </entries>
      </connection-factory>
      <pooled-connection-factory name="hornetq-ra">
        <transaction mode="xa"/>
        <connectors>
          <connector-ref connector-name="in-vm"/>
        </connectors>
        <entries>
          <entry name="java:/JmsXA"/>
        </entries>
      </pooled-connection-factory>
    </jms-connection-factories>
    <connectors>
      <netty-connector name="netty" socket-binding="messaging"/>
      <netty-connector name="netty-throughput" socket-binding="messaging-throughput">

```



```

        <param key="batch-delay" value="50"/>
    </netty-connector>
    <in-vm-connector name="in-vm" server-id="0"/>
</connectors>
<acceptors>
    <netty-acceptor name="netty" socket-binding="messaging"/>
    <netty-acceptor name="netty-throughput" socket-binding="messaging-throughput">
        <param key="batch-delay" value="50"/>
        <param key="direct-deliver" value="false"/>
    </netty-acceptor>
    <in-vm-acceptor name="in-vm" server-id="0"/>
</acceptors>
</hornetq-server>
</subsystem>

```

7. Configure the socket binding groups

The netty connectors reference the **messaging** and **messaging-throughput** socket bindings. The **messaging** socket binding uses port 5445, and the **messaging-throughput** socket binding uses port 5455. The **<socket-binding-group>** tag is in a separate section of the server configuration file. Ensure the following socket bindings are present in the **<socket-binding-groups>** element:

```

<socket-binding-group name="standard-sockets" default-interface="public" port-
offset="${jboss.socket.binding.port-offset:0}">
    ...
    <socket-binding name="messaging" port="5445"/>
    <socket-binding name="messaging-throughput" port="5455"/>
    ...
</socket-binding-group>

```

8. Add queue instances to HornetQ

There are 4 ways to setup the queue instances (or JMS destinations) for HornetQ.

- Use the Management Console

To use the Management Console, the server must have been started in the **Message-Enabled** mode. You can do this by using the **-c** option and forcing the use of the **standalone-full.xml** (for standalone servers) configuration file. For example, in the standalone mode, the following will start the server in a message enabled mode

```
./standalone.sh -c standalone-full.xml
```

Once the server has started, logon to the Management Console and select the **Configuration** tab. Expand the **Subsystems** menu, then expand the **Messaging** menu and click **Destinations**. Next to **Default** on the JMS Messaging Provider table, click **View**, and then click **Add** to enter details of the JMS destination.

- Use the Management CLI:

First, connect to the Management CLI:

```
bin/jboss-cli.sh --connect
```

Next, change into the messaging subsystem:

```
■
```

```
cd /subsystem=messaging/hornetq-server=default
```

Finally, execute an add operation, replacing the examples values given below with your own:

```
./jms-queue=testQueue:add(durable=false,entries=
["java:jboss/exported/jms/queue/test"])
```

- Create a JMS configuration file and add it to the deployments folder

Start by creating a JMS configuration file: *example-jms.xml*. Add the following entries to it, replacing the values with your own:

```
<?xml version="1.0" encoding="UTF-8"?>      <messaging-deployment
xmlns="urn:jboss:messaging-deployment:1.0">
  <hornetq-server>
    <jms-destinations>
      <jms-queue name="testQueue">
        <entry name="queue/test"/>
        <entry name="java:jboss/exported/jms/queue/test"/>
      </jms-queue>
      <jms-topic name="testTopic">
        <entry name="topic/test"/>
        <entry name="java:jboss/exported/jms/topic/test"/>
      </jms-topic>
    </jms-destinations>
  </hornetq-server>
</messaging-deployment>
```

Save this file in the deployments folder and do a deployment.

- Add entries in the JBoss EAP 6 configuration file.

Queue attributes can be set in one of two ways.

- Configuration at a JMS level

The following shows a queue predefined in the **standalone.xml** or **domain.xml** configuration file.

```
<jms-queue name="selectorQueue">
  <entry name="/queue/selectorQueue"/>
  <selector string="color='red'"/>
  <durable>true</durable>
</jms-queue>
```

This name attribute of queue defines the name of the queue. When we do this at a jms level we follow a naming convention so the actual name of the core queue will be **jms.queue.selectorQueue**.

The entry element configures the name that will be used to bind the queue to JNDI. This is a mandatory element and the queue can contain multiple of these to bind the same queue to different names.

The selector element defines what JMS message selector the predefined queue will have. Only messages that match the selector will be added to the queue. This is an optional element with a default of null when omitted.

The durable element specifies whether the queue will be persisted. This again is optional and defaults to true if omitted.

- Configuration at a core level

A queue can be predefined at a core level in the **standalone.xml** or **domain.xml** file. For example:

```
<core-queues>
  <queue name="jms.queue.selectorQueue">
    <address>jms.queue.selectorQueue</address>
    <filter string="color='red'"/>
    <durable>true</durable>
  </queue>
</core-queues>
```

9. Perform additional configuration

If you need additional settings, review the DTD in **EAP_HOME/docs/schema/jboss-as-messaging_1_4.xsd**.

[Report a bug](#)

18.8.2. Configure JMS Address Settings

The JMS subsystem has several configurable options which control aspects of how and when a message is delivered, how many attempts should be made, and when the message expires. These configuration options all exist within the **<address-settings>** configuration element.

A common feature of address configurations is the syntax for matching multiple addresses, also known as wild cards.

Wildcard Syntax

Address wildcards can be used to match multiple similar addresses with a single statement, similar to how many systems use the asterisk (*****) character to match multiple files or strings with a single search. The following characters have special significance in a wildcard statement.

Table 18.5. JMS Wildcard Syntax

Character	Description
. (a single period)	Denotes the space between words in a wildcard expression.
# (a pound or hash symbol)	Matches any sequence of zero or more words.
* (an asterisk)	Matches a single word.

Table 18.6. JMS Wildcard Examples

Example	Description
news.europe.#	Matches news.europe , news.europe.sport , news.europe.politic , but not news.usa or europe .
news.*	Matches news.europe but not news.europe.sport .
news.*.sport	Matches news.europe.sport and news.usa.sport , but not news.europe.politics .

Example 18.2. Default Address Setting Configuration

The values in this example are used to illustrate the rest of this topic.

```
<address-settings>
  <!--default for catch all-->
  <address-setting match="#">
    <dead-letter-address>jms.queue.DLQ</dead-letter-address>
    <expiry-address>jms.queue.ExpiryQueue</expiry-address>
    <redelivery-delay>0</redelivery-delay>
    <max-size-bytes>10485760</max-size-bytes>
    <address-full-policy>BLOCK</address-full-policy>
    <message-counter-history-day-limit>10</message-counter-history-day-limit>
  </address-setting>
</address-settings>
```

Table 18.7. Description of JMS Address Settings

Element	Description	Default Value	Type
address-full-policy	Determines what happens when an address where max-size-bytes is specified becomes full.	PAGE	STRING
dead-letter-address	If a dead letter address is specified, messages are moved to the dead letter address if max-delivery-attempts delivery attempts have failed. Otherwise, these undelivered messages are discarded. Wildcards are allowed.	jms.queue.DLQ	STRING

Element	Description	Default Value	Type
expiry-address	If the expiry address is present, expired messages are sent to the address or addresses matched by it, instead of being discarded. Wildcards are allowed.	jms.queue.ExpiryQueue	STRING
last-value-queue	Defines whether a queue only uses last values or not.	false	BOOLEAN
max-delivery-attempts	The maximum number of times to attempt to re-deliver a message before it is sent to dead-letter-address or discarded.	10	INT
max-size-bytes	The maximum bytes size.	10485760L	LONG
message-counter-history-day-limit	Day limit for the message counter history.	10	INT
page-max-cache-size	The number of page files to keep in memory to optimize IO during paging navigation.	5	INT
page-size-bytes	The paging size.	5	INT
redelivery-delay	Time to delay between re-delivery attempts of messages, expressed in milliseconds. If set to 0 , re-delivery attempts occur indefinitely.	0L	LONG
redistribution-delay	Defines how long to wait when the last consumer is closed on a queue before redistributing any messages.	-1L	LONG
send-to-dla-on-no-route	A parameter for an address that sets the condition of a message not routed to any queues to instead be sent to the dead letter address (DLA) indicated for that address.	false	BOOLEAN

Element	Description	Default Value	Type
slow-consumer-threshold	The minimum rate of message consumption allowed before a consumer is considered "slow." Measured in messages-per-second.	-1	INT
slow-consumer-policy	What should happen when a slow consumer is detected. KILL will kill the consumer's connection (which will obviously impact any other client threads using that same connection). NOTIFY will send a CONSUMER_SLOW management notification which an application could receive and take action with.		STRING
slow-consumer-check-period	How often to check for slow consumers on a particular queue. Measured in seconds.	5	INT

- **Configure Address Setting and Pattern Attributes**

Choose either the Management CLI or the Management Console to configure your pattern attributes as required.

- **Configure the Address Settings Using the Management CLI**

Use the Management CLI to configure address settings.

- a. **Add a New Pattern**

Use the **add** operation to create a new address setting if required. You can run this command from the root of the Management CLI session, which in the following examples creates a new pattern titled *patternname*, with a **max-delivery-attempts** attribute declared as 5. The examples for both Standalone Server and a Managed Domain editing on the **full** profile are shown.

```
[standalone@localhost:9999 /] /subsystem=messaging/hornetq-server=default/address-setting=patternname/:add(max-delivery-attempts=5)
```

```
[domain@localhost:9999 /] /profile=full/subsystem=messaging/hornetq-server=default/address-setting=patternname/:add(max-delivery-attempts=5)
```

- b. **Edit Pattern Attributes**

Use the **write** operation to write a new value to an attribute. You can use tab completion to help complete the command string as you type, as well as to expose the available attributes. The following example updates the **max-delivery-attempts** value to *10*

```
[standalone@localhost:9999 /] /subsystem=messaging/hornetq-
server=default/address-setting=patternname/:write-attribute(name=max-delivery-
attempts,value=10)
```

```
[domain@localhost:9999 /] /profile=full/subsystem=messaging/hornetq-
server=default/address-setting=patternname/:write-attribute(name=max-delivery-
attempts,value=10)
```

c. **Confirm Pattern Attributes**

Confirm the values are changed by running the **read-resource** operation with the **include-runtime=true** parameter to expose all current values active in the server model.

```
[standalone@localhost:9999 /] /subsystem=messaging/hornetq-
server=default/address-setting=patternname/:read-resource
```

```
[domain@localhost:9999 /] /profile=full/subsystem=messaging/hornetq-
server=default/address-setting=patternname/:read-resource
```

o **Configure the Address Settings Using the Management Console**

Use the Management Console to configure address settings.

- a. Log into the Management Console of your Managed Domain or Standalone Server.
- b. Select the **Configuration** tab at the top of the screen. For Domain mode, select a profile from the **Profile** menu at the top left. Only the **full** and **full-ha** profiles have the **messaging** subsystem enabled.
- c. Expand the **Messaging** menu, and select **Destinations**.
- d. A list of JMS Providers is shown. In the default configuration, only one provider, called **default**, is shown. Click **View** to view the detailed settings for this provider.
- e. Click the **Address Settings** tab. Either add a new pattern by clicking **Add**, or select an existing pattern and click **Edit** to update the settings.
- f. If you are adding a new pattern, the **Pattern** field refers to the **match** parameter of the **address-setting** element. You can also edit the **Dead Letter Address, Expiry Address, Redelivery Delay**, and **Max Delivery Attempts**. Other options need to be configured using the Management CLI.

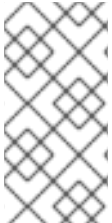
[Report a bug](#)

18.8.3. Temporary Queues and Runtime Queues

While designing request-reply scenarios that involve a client sending a request and waiting for a reply, an addressable issue is whether each runtime instance of the client has a dedicated queue for its replies or whether the runtime instances access a shared queue, selecting their specific reply message based on an appropriate attribute. If multiple queues are required, then we need the ability to create a queue dynamically for use by the client, and JMS provides this facility using the concept of temporary queues. The **TemporaryQueue** is created on request by the **QueueSession** and exists until it is deleted or for

the life of the **QueueConnection** (i.e. until the **QueueConnection** is closed). This means that although the **TemporaryQueue** is created by a specific **QueueSession**, it can be reused by any other **QueueSessions** created from the same **QueueConnection** to create a **QueueReceiver**.

The tradeoff between having a shared queue for replies or having individual temporary queues is influenced by the potential number of active client instances. With a shared-queue approach, at some provider-specific threshold, contention for access to the queue can become a concern. This has to be contrasted against the additional overhead associated with the provider creating queue storage at runtime and the impact on machine memory of having to host a potentially large number of temporary queues.



NOTE

The creation of temporary queues is accomplished with the **createTemporaryQueue** method. Similarly, the creation of temporary topics is accomplished with the **createTemporaryTopic** method. Both of these methods are for creating the physical queue and physical topic.

If there are messages that have been received but not acknowledged when a **QueueSession** terminates, these messages will be retained and redelivered when a consumer next accesses the queue. A **QueueSession** is used for creating Point-to-Point specific objects. In general, use the Session object. The **QueueSession** is used to support existing code. Using the Session object simplifies the programming model, and allows transactions to be used across the two messaging domains.

A **TopicSession** is used for creating Pub/Sub specific objects. In general, use the Session object, and use **TopicSession** only to support existing code. Using the Session object simplifies the programming model, and allows transactions to be used across the two messaging domains.

[Report a bug](#)

18.8.4. Last-Value Queues

Last-Value queues are special queues which discard any messages when a newer message with the same value for a well-defined Last-Value property is put in the queue. In other words, a Last-Value queue only retains the last value. A typical example for Last-Value queue is for stock prices, where you are only interested by the latest value for a particular stock.

Configuring Last-Value Queues

Last-value queues are defined in the address-setting configuration:

```
<address-setting match="jms.queue.lastValueQueue">
  <last-value-queue>true</last-value-queue>
</address-setting>
```

Using Last-Value Property

The property name used to identify the last value is **"_HQ_LVQ_NAME"** (or the constant **Message.HDR_LAST_VALUE_NAME** from the Core API). For example, if two messages with the same value for the Last-Value property are sent to a Last-Value queue, only the latest message will be kept in the queue:

Example 18.3. Send 1st message with Last-Value property set to *STOCK_NAME*

```
TextMessage message = session.createTextMessage("1st message with Last-Value property
```



```
set");
message.setStringProperty("_HQ_LVQ_NAME", "STOCK_NAME");
producer.send(message);
```

Example 18.4. Send 2nd message with Last-Value property set to `STOCK_NAME`

```
message = session.createTextMessage("2nd message with Last-Value property set");
message.setStringProperty("_HQ_LVQ_NAME", "STOCK_NAME");
producer.send(message);
```

Example 18.5. Only the 2nd message will be received; it is the latest with the Last-Value property set:

```
TextMessage messageReceived = (TextMessage)messageConsumer.receive(5000);
System.out.format("Received message: %s\n", messageReceived.getText());
```

[Report a bug](#)

18.8.5. Core and JMS Destinations

HornetQ core does not have any concept of a JMS topic. A JMS topic is implemented in core as an address (the topic name) with zero or more queues bound to it. Each queue bound to that address represents a topic subscription. Likewise, a JMS queue is implemented as an address (the JMS queue name) but with one single queue bound to it which represents the JMS queue.

A JMS topic is the channel through which users subscribe to receive specific messages from a producer in the publish-and-subscribe model of JMS messaging.

A JMS queue is a channel through which users "pull" messages they want to receive using the Point-to-point (p2p) model, instead of automatically receiving messages on a particular topic. The producer submits messages to the queue, and recipients can browse the queue and decide which messages they wish to receive. In the p2p model, users can see the contents of the messages held in the queue before deciding whether or not to accept their delivery.

If you want to configure settings for a JMS Queue with the name `orders.europe`, you need to configure the corresponding core queue `jms.queue.orders.europe`:

```
<!-- expired messages in JMS Queue "orders.europe" will be sent to the JMS Queue "expiry.europe" -
->
<address-setting match="jms.queue.orders.europe">
  <expiry-address>jms.queue.expiry.europe</expiry-address>
  ...
</address-setting>
```

[Report a bug](#)

18.8.6. JMS Message Selectors

If your messaging application needs to filter the messages it receives, you can use a JMS API message selector, which allows a message consumer to specify the messages it is interested in. Message selectors assign the work of filtering messages to the JMS provider rather than to the application.

You can define your message selector as follows:

```
@MessageDriven(name = "MDBMessageSelectorExample", activationConfig =
{
    @ActivationConfigProperty(propertyName = "destinationType", propertyValue = "javax.jms.Queue"),
    @ActivationConfigProperty(propertyName = "destination", propertyValue = "queue/testQueue"),
    @ActivationConfigProperty(propertyName = "messageSelector", propertyValue = "color = 'RED'")
})
@TransactionManagement(value= TransactionManagementType.CONTAINER)
@TransactionAttribute(value= TransactionAttributeType.REQUIRED)
public class MDBMessageSelectorExample implements MessageListener
{
    public void onMessage(Message message)....
}
```

[Report a bug](#)

18.8.7. Configure Messaging with HornetQ

The recommended method of configuring messaging in JBoss EAP 6 is in either the Management Console or Management CLI. You can make persistent changes with either of these management tools without needing to manually edit the **standalone.xml** or **domain.xml** configuration files. It is useful however to familiarize yourself with the messaging components of the default configuration files, where documentation examples using management tools give configuration file snippets for reference.

[Report a bug](#)

18.8.8. Enable Logging for HornetQ

You can enable logging for HornetQ in EAP 6.x using any of the following approaches:

- Editing server configuration files (**standalone-full.xml** and **standalone-full-ha.xml**) manually
- Editing server configuration files using the CLI

Procedure 18.1. Set HornetQ logging by editing server configuration files manually

1. Open the server configuration file(s) for editing. For example **standalone-full.xml** and **standalone-full-ha.xml**
2. Navigate to logging subsystem configuration in the file(s). The default configuration looks like this:

```
<logger category="com.arjuna">
    <level name="TRACE"/>
</logger>
...
<logger category="org.apache.tomcat.util.modeler">
    <level name="WARN"/>
</logger>
....
```

3. Add the **org.hornetq** logger category along with the desired logging level as shown in the following example:

```

<logger category="com.arjuna">
  <level name="TRACE"/>
</logger>
...
<logger category="org.hornetq">
  <level name="INFO"/>
</logger>
....

```

Result

HornetQ logging is enabled and log messages are processed based on the configured log level.

Set HornetQ logging by editing server configuration files using the CLI

You can also use CLI to add the **org.hornetq** logger category along with the desired logging level to server configuration file(s). For more information see: [Section 12.3.2, "Configure a Log Category in the CLI"](#)

[Report a bug](#)

18.8.9. Configuring HornetQ Core Bridge

Example 18.6. Example configuration for HornetQ Core Bridge:

The values in this example are used to illustrate the rest of this topic.

```

<bridges>
  <bridge name="myBridge">
    <queue-name>jms.queue.InQueue</queue-name>
    <forwarding-address>jms.queue.OutQueue</forwarding-address>
  <ha>true</ha>
  <reconnect-attempts>-1</reconnect-attempts>
  <use-duplicate-detection>true</use-duplicate-detection>
  <static-connectors>
    <connector-ref>
      bridge-connector
    </connector-ref>
  </static-connectors>
</bridge>
</bridges>

```

Table 18.8. HornetQ Core Bridge Attributes

Attribute	Description
name	All bridges must have a unique name on the server.

Attribute	Description
queue-name	This mandatory parameter is the unique name of the local queue that the bridge consumes from. The queue must already exist by the time the bridge is instantiated at start-up.
forwarding-address	This is the address on the target server that the message will be forwarded to. If a forwarding address is not specified, then the original address of the message will be retained.
ha	This optional parameter determines whether or not this bridge should support high availability. true means it will connect to any available server in a cluster and support failover. The default value is false.
reconnect-attempts	This optional parameter determines the total number of reconnect attempts the bridge should make before giving up and shutting down. A value of -1 signifies an unlimited number of attempts. The default value is -1.
use-duplicate-detection	This optional parameter determines whether the bridge will automatically insert a duplicate id property into each message that it forwards.
static-connectors	The static-connectors is a list of connector-ref elements pointing to connector elements defined elsewhere. A connector encapsulates knowledge of what transport to use (TCP, SSL, HTTP etc) as well as the server connection parameters (host, port etc).

[Report a bug](#)

18.8.10. Configuring JMS Bridge

HornetQ includes a fully functional JMS message bridge. The function of this bridge is to consume messages from a source queue or topic, and send them to a target queue or topic, typically on a different server.

The source and target servers do not have to be in the same cluster, which makes bridging suitable for reliably sending messages from one cluster to another, for instance across a WAN, and where the connection is unreliable.

A bridge can be deployed as a standalone application, with HornetQ standalone server or inside a JBoss AS instance. The source and the target can be located in the same virtual machine or another one.

Example 18.7. Example configuration for JMS Bridge:

The values in this example are used to illustrate the rest of this topic.

```
<subsystem>
  <subsystem xmlns="urn:jboss:domain:messaging:1.3">
    <hornetq-server>
```

```

...
</hornetq-server>

<jms-bridge name="myBridge">
  <source>
    <connection-factory name="ConnectionFactory"/>
    <destination name="jms/queue/InQueue"/>
  </source>
  <target>
    <connection-factory name="jms/RemoteConnectionFactory"/>
    <destination name="jms/queue/OutQueue"/>
    <context>
      <property key="java.naming.factory.initial"
value="org.jboss.naming.remote.client.InitialContextFactory"/>
      <property key="java.naming.provider.url" value="remote://192.168.40.1:4447"/>
    </context>
  </target>
  <quality-of-service>AT_MOST_ONCE</quality-of-service>
  <failure-retry-interval>1000</failure-retry-interval>
  <max-retries>-1</max-retries>
  <max-batch-size>10</max-batch-size>
  <max-batch-time>100</max-batch-time>
  <add-messageID-in-header>true</add-messageID-in-header>
</jms-bridge>
...
</subsystem>

```



WARNING

It is recommended to use a connection factory that does not set the **reconnect-attempts** parameter (or sets it to **0**), as JMS Bridge has its own **max-retries** parameter to handle reconnection. This is to avoid a potential collision that may result in longer reconnection times.

Table 18.9. HornetQ Core JMS Attributes

Attribute	Description
name	All bridges must have a unique name on the server.
source connection-factory	This injects the SourceCFF bean (also defined in the beans file). This bean creates the source ConnectionFactory.
source destination name	This injects the SourceDestinationFactory bean (also defined in the beans file). This bean creates the source Destination.

Attribute	Description
target connection-factory	This injects the TargetCFF bean (also defined in the beans file). This bean creates the target ConnectionFactory.
target destination name	This injects the TargetDestinationFactory bean (also defined in the beans file). This bean creates the target Destination.
quality-of-service	This parameter represents the required quality of service mode. The possible values are: AT_MOST_ONCE, DUPLICATES_OK, ONCE_AND_ONLY_ONCE
failure-retry-interval	This represents the amount of time in milliseconds to wait between trying to recreate connections to the source or target servers when the bridge has detected they have failed.
max-retries	This represents the number of attempts to recreate connections to the source or target servers when the bridge has detected they have failed. The bridge will give up after trying this number of times. -1 represents 'try forever'.
max-batch-size	This represents the maximum number of messages to consume from the source destination before sending them in a batch to the target destination. Its value must ≥ 1 .
max-batch-time	This represents the maximum number of milliseconds to wait before sending a batch to target, even if the number of messages consumed has not reached MaxBatchSize. Its value must be -1 to represent 'wait forever', or ≥ 1 to specify an actual time.
add-messageID-in-header	<p>If true, then the original message's message id will be appended in the message sent to the destination in the header HORNETQ_BRIDGE_MSG_ID_LIST. If the message is bridged more than once, each message id will be appended. This enables a distributed request-response pattern to be used.</p> <p>When you receive the message you can send a response using the correlation id of the first message id, so when the original sender receives the message, it is easy to correlate.</p>



NOTE

When shutting down a server that has a deployed JMS bridge with *quality-of-service* attribute set to **ONCE_AND_ONLY_ONCE**, shut the server down with the JMS bridge first to avoid unexpected exceptions.

For more complete instructions, see [Section 18.13.2, "Create a JMS Bridge"](#) .

[Report a bug](#)

18.8.11. Configure Delayed Redelivery

Introduction

Delayed redelivery is defined in the `<redelivery-delay>` element, which is a child element of the `<address-setting>` configuration element in the Java Messaging Service (JMS) subsystem configuration.

```
<!-- delay redelivery of messages for 5s -->
<address-setting match="jms.queue.exampleQueue">
  <redelivery-delay>5000</redelivery-delay>
</address-setting>
```

If a redelivery delay is specified, the JMS system waits for the duration of this delay before redelivering the messages. If `<redelivery-delay>` is set to `0`, there is no redelivery delay. Address wildcards can be used on the `match` attribute of `<address-match>` element to configure the redelivery delay for addresses that match the wildcard.

[Report a bug](#)

18.8.12. Configure Dead Letter Addresses

Introduction

A dead letter address is defined in the `<address-setting>` element of the Java Messaging Service (JMS) subsystem configuration.

```
<!-- undelivered messages in exampleQueue will be sent to the dead letter address
deadLetterQueue after 3 unsuccessful delivery attempts -->
-->
<address-setting match="jms.queue.exampleQueue">
  <dead-letter-address>jms.queue.deadLetterQueue</dead-letter-address>
  <max-delivery-attempts>3</max-delivery-attempts>
</address-setting>
```

If a `<dead-letter-address>` is not specified, messages are removed after trying to deliver `<max-delivery-attempts>` times. By default, messages delivery is attempted 10 times. Setting `<max-delivery-attempts>` to `-1` allows infinite redelivery attempts. For example, a dead letter can be set globally for a set of matching addresses and you can set `<max-delivery-attempts>` to `-1` for a specific address setting to allow infinite redelivery attempts only for this address. Address wildcards can also be used to configure dead letter settings for a set of addresses.

[Report a bug](#)

18.8.13. Configure Message Expiry Addresses

Introduction

Message expiry addresses are defined in the address-setting configuration of the Java Messaging Service (JMS). For example:

```
<!-- expired messages in exampleQueue will be sent to the expiry address expiryQueue -->
<address-setting match="jms.queue.exampleQueue">
```

```
<expiry-address>jms.queue.expiryQueue</expiry-address>
</address-setting>
```

If messages are expired and no expiry address is specified, messages are simply removed from the queue and dropped. *Address wildcards* can also be used to configure specific ranges of an expiry address for a set of addresses. See [Section 18.8.2, "Configure JMS Address Settings"](#) for the JMX wildcard syntax and examples.

[Report a bug](#)

18.8.14. Flow Control

Flow control is used to limit the flow of data between a client and server, or a server and another server, in order to prevent the client or the server being overloaded with data.

- Consumer Flow Control - controls the flow of data between the server and the client as the client consumes messages. For performance reasons clients normally buffer messages before delivering to the consumer via the **receive()** method or asynchronously via a message listener.

Rate-limited flow control: It is possible to control the rate at which a consumer can consume messages. This is a form of throttling and can be used to ensure that a consumer never consumes messages at a rate faster than the rate specified. The rate must be a positive integer to enable this functionality and is the maximum desired message consumption rate specified in units of messages per second. Setting this to **-1** disables rate limited flow control. The default value is **-1**.

Example 18.8. Rate limited flow control using JMS

If JNDI is used to look up the connection factory, the max rate can be configured in **standalone.xml** or **domain.xml**:

```
<connection-factory name="ConnectionFactory">
  <connectors>
    <connector-ref connector-name="netty-connector"/>
  </connectors>
  <entries>
    <entry name="ConnectionFactory"/>
  </entries>
  <!-- We limit consumers created on this connection factory to consume messages at a
  maximum rate of 10 messages per sec -->
  <consumer-max-rate>10</consumer-max-rate>
</connection-factory>
```

If the connection factory is directly instantiated, the max rate size can be set via the **HornetQConnectionFactory.setConsumerMaxRate(int consumerMaxRate)** method.

- Producer flow control - limits the amount of data sent from a client to a server to prevent the server being overwhelmed.

Window based flow control: HornetQ producers, by default, can only send messages to an address as long as they have sufficient credits to do so. The amount of credits required to send a message is given by the size of the message. As producers run low on credits they request more from the server, when the server sends them more credits they can send more messages. The amount of credits a producer requests in one go is known as the window size.

Example 18.9. Producer window size flow control using JMS

If JNDI is used to look up the connection factory, the producer window size can be configured in **standalone.xml** or **domain.xml**:

```
<connection-factory name="ConnectionFactory">
  <connectors>
    <connector-ref connector-name="netty-connector"/>
  </connectors>
  <entries>
    <entry name="ConnectionFactory"/>
  </entries>
  <producer-window-size>10</producer-window-size>
</connection-factory>
```

If the connection factory is directly instantiated, the producer window size can be set via the **HornetQConnectionFactory.setProducerWindowSize(int producerWindowSize)** method.

[Report a bug](#)

18.8.15. Reference for HornetQ Configuration Attributes

The JBoss EAP 6 implementation of HornetQ exposes the following attributes for configuration. You can use the Management CLI to show the configurable or viewable attributes with the **read-resource** operation.

Example 18.10. Use read-resource to show attributes

```
[standalone@localhost:9999 /] /subsystem=messaging/hornetq-server=default:read-resource
```

Table 18.10. HornetQ Attributes

Attribute	Default Value	Type	Description
allow-failback	true	BOOLEAN	Whether this server will automatically shutdown if the original live server comes back up
async-connection-execution-enabled	true	BOOLEAN	Whether incoming packets on the server must be handed off to a thread from the thread pool for processing
address-setting			An address setting defines some attributes that are defined against an address wildcard rather than a specific queue

Attribute	Default Value	Type	Description
acceptor			An acceptor defines a way in which connections can be made to the HornetQ server
backup-group-name		STRING	The name of a set of live/backups that must replicate with each other
backup	false	BOOLEAN	Whether this server is a backup server
check-for-live-server	false	BOOLEAN	Whether a replicated live server must check the current cluster to see if there is already a live server with the same node ID
clustered	false	BOOLEAN	[Deprecated] Whether the server is clustered
cluster-password	CHANGE ME!!	STRING	The password used by cluster connections to communicate between the clustered nodes
cluster-user	HORNETQ. CLUSTER.A DMIN.USER	STRING	The user used by cluster connections to communicate between the clustered nodes
cluster-connection			Cluster connections group servers into clusters so that messages can be load balanced between the nodes of the cluster
create-bindings-dir	true	BOOLEAN	Whether the server must create the bindings directory on start up
create-journal-dir	true	BOOLEAN	Whether the server must create the journal directory on start up
connection-ttl-override	-1L	LONG	If set, this will override how long (in ms) to keep a connection alive without receiving a ping

Attribute	Default Value	Type	Description
connection-factory			Defines a connection factory
connector			A connector can be used by a client to define how it connects to a server
connector-service			
divert			A messaging resource that allows you to transparently divert messages routed to one address to some other address, without making any changes to any client application logic
discovery-group			Multicast group to listen to receive broadcast from other servers announcing their connectors
failback-delay	5000	LONG	How long to wait before failback occurs on live server restart
failover-on-shutdown	false	BOOLEAN	Whether this backup server (if it is a backup server) must come live on a normal server shutdown
grouping-handler			Makes decisions about which node in a cluster must handle a message with a group id assigned
id-cache-size	20000	INT	The size of the cache for pre-creating message IDs
in-vm-acceptor			Defines a way in which in-VM connections can be made to the HornetQ server
in-vm-connector			Used by an in-VM client to define how it connects to a server

Attribute	Default Value	Type	Description
jmx-domain	org.hornetq	STRING	The JMX domain used to register internal HornetQ MBeans in the MBeanServer
jmx-management-enabled	false	BOOLEAN	Whether HornetQ must expose its internal management API via JMX. This is not recommended, as accessing these MBeans can lead to inconsistent configuration
journal-buffer-size	501760 (490KiB)	LONG	The size of the internal buffer on the journal
journal-buffer-timeout	500000 (0.5 milliseconds)) for ASYNCIO journal and 3333333 (3.33 milliseconds)) for NIO journal	LONG	The timeout (in nanoseconds) used to flush internal buffers on the journal
journal-compact-min-files	10	INT	The minimal number of journal data files before we can start compacting
journal-compact-percentage	30	INT	The percentage of live data on which we consider compacting the journal
journal-file-size	10485760	LONG	The size (in bytes) of each journal file
journal-max-io	1	INT	The maximum number of write requests that can be in the AIO queue at any one time. The default value changes to 500 when ASYNCIO journal is used
journal-min-files	2	INT	How many journal files to pre-create

Attribute	Default Value	Type	Description
journal-sync-non-transactional	true	BOOLEAN	Whether to wait for non transaction data to be synced to the journal before returning a response to the client
journal-sync-transactional	true	BOOLEAN	Whether to wait for transaction data to be synchronized to the journal before returning a response to the client
journal-type	ASYNCIO	String	The type of journal to use. This attribute can take the values "ASYNCIO" or "NIO"
jms-topic			Defines a JMS topic
live-connector-ref	reference	STRING	[Deprecated] The name of the connector used to connect to the live connector. If this server is not a backup that uses shared nothing HA, it's value is "undefined"
log-journal-write-rate	false	BOOLEAN	Whether to periodically log the journal's write rate and flush rate
mask-password	true	BOOLEAN	
management-address	jms.queue.hornetq.management	STRING	Address to send management messages to
management-notification-address	hornetq.notifications	STRING	The name of the address that consumers bind to in order to receive management notifications
max-saved-replicated-journal-size	2	INT	The maximum number of backup journals to keep after failback occurs
memory-measure-interval	-1	LONG	Frequency to sample JVM memory in ms (or -1 to disable memory sampling)

Attribute	Default Value	Type	Description
memory-warning-threshold	25	INT	Percentage of available memory which if exceeded results in a warning log
message-counter-enabled	false	BOOLEAN	Whether message counters are enabled
message-counter-max-day-history	10	INT	How many days to keep message counter history
message-counter-sample-period	10000	LONG	The sample period (in ms) to use for message counters
message-expiry-scan-period	30000	LONG	How often (in ms) to scan for expired messages
message-expiry-thread-priority	3	INT	The priority of the thread expiring messages
page-max-concurrent-io	5	INT	The maximum number of concurrent reads allowed on paging
perf-blast-pages	-1	INT	
persist-delivery-count-before-delivery	false	BOOLEAN	Whether the delivery count is persisted before delivery. False means that this only happens after a message has been canceled
persist-id-cache	true	BOOLEAN	Whether IDs are persisted to the journal
persistence-enabled	true	BOOLEAN	Whether the server will use the file based journal for persistence
pooled-connection-factory			Defines a managed connection factory
remoting-interceptors	undefined	LIST	[Deprecated] The list of interceptor classes used by this server

Attribute	Default Value	Type	Description
remoting-incoming-interceptors	undefined	LIST	The list of incoming interceptor classes used by this server
remoting-outgoing-interceptors	undefined	LIST	The list of outgoing interceptor classes used by this server
run-sync-speed-test	false	BOOLEAN	Whether to perform a diagnostic test on how fast your disk can sync on startup. Useful when determining performance issues
replication-clustername		STRING	The name of the cluster connection to replicate from if more than one cluster connection is configured
runtime-queue			A runtime queue
remote-connector			Used by a remote client to define how it connects to a server
remote-acceptor			Defines a way in which remote connections can be made to the HornetQ server
scheduled-thread-pool-max-size	5	INT	The number of threads that the main scheduled thread pool has
security-domain	other	STRING	The security domain to use in order to verify user and role information
security-enabled	true	BOOLEAN	Whether security is enabled
security-setting			A security setting allows sets of permissions to be defined against queues based on their address
security-invalidation-interval	10000	LONG	How long (in ms) to wait before invalidating the security cache

Attribute	Default Value	Type	Description
server-dump-interval	-1	LONG	How often to dump basic runtime information to the server log. A value less than 1 disables this feature
shared store	true	BOOLEAN	Whether this server is using a shared store for failover
thread-pool-max-size	30	INT	The number of threads that the main thread pool has. -1 means no limit
transaction-timeout	300000	LONG	How long (in ms) before a transaction can be removed from the resource manager after create time
transaction-timeout-scan-period	1000	LONG	How often (in ms) to scan for timeout transactions
wild-card-routing-enabled	true	BOOLEAN	Whether the server supports wild card routing



WARNING

The value of **journal-file-size** must be higher than the size of message sent to server, or the server will not be able to store the message.

[Report a bug](#)

18.8.16. Set Message Expiry

Introduction

Sent messages can be set to expire on server if they're not delivered to consumer after specified amount of time (milliseconds). Using Java Messaging Service (JMS) or HornetQ Core API, the expiration time can be set directly on the message. For example:

```
// message will expire in 5000ms from now
message.setExpiration(System.currentTimeMillis() + 5000);
```

JMS **MessageProducer** includes a **TimeToLive** parameter which controls message expiry for the messages it sends:

■


```
// messages sent by this producer will be retained for 5s (5000ms) before expiration
producer.setTimeToLive(5000);
```

Expired messages which are consumed from an expiry address have the following properties:

- `_HQ_ORIG_ADDRESS`

A string property containing the original address of the expired message.

- `_HQ_ACTUAL_EXPIRY`

A long property containing the actual expiration time of the expired message.

Besides setting the time-to-live parameter on the JMS producer, you can also set it on a per-message basis. You can achieve this by adding TTL parameter to producer's send method when sending the message.

```
producer.send(message, DeliveryMode.PERSISTENT, 0, 5000)
```

Where, the last parameter is message specific TTL.

Configuring Expiry Addresses

Expiry address are defined in the address-setting configuration:

```
<!-- expired messages in exampleQueue will be sent to the expiry address expiryQueue -->
<address-setting match="jms.queue.exampleQueue">
  <expiry-address>jms.queue.expiryQueue</expiry-address>
</address-setting>
```

If the messages are expired and no expiry address is specified, the messages are removed from the queue and dropped.

Configuring Expiry Reaper Thread

A reaper thread periodically inspects the queues to validate if messages have expired.

- `message-expiry-scan-period`

How often the queues will be scanned to detect expired messages (in milliseconds, default is 30000ms, set to -1 to disable the reaper thread).

- `message-expiry-thread-priority`

The reaper thread priority. It must be between 0 and 9, 9 being the highest priority, default is 3.

[Report a bug](#)

18.9. PRE_ACKNOWLEDGE MODE

JMS specifies three acknowledgement modes:

- `AUTO_ACKNOWLEDGE`
- `CLIENT_ACKNOWLEDGE`

- DUPS_OK_ACKNOWLEDGE

HornetQ supports two additional modes: PRE_ACKNOWLEDGE and INDIVIDUAL_ACKNOWLEDGE.

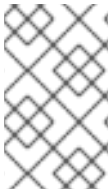
In some scenarios you could afford to lose messages in event of failure, so it would help to acknowledge the message on the server before delivering it to the client.

This extra mode is supported by HornetQ and is called as pre-acknowledge mode.

The disadvantage of acknowledging the messages on the server before delivery is that the message is lost if the system crashes after acknowledging the message on the server but before it is delivered to the client. In that case, the message is lost and will not be recovered when the system restart.

Depending on the messaging case, pre-acknowledgement mode can avoid extra network traffic and CPU at the cost of coping with message loss.

An example use case for pre-acknowledgement is related to stock price update messages. With these messages it might be reasonable to lose a message in event of crash, since the next price update message will arrive soon, overriding the previous price.



NOTE

If you use pre-acknowledge mode, then you will lose transactional semantics for messages being consumed, since they are being acknowledged first on the server, not when you commit the transaction.

[Report a bug](#)

18.9.1. Using PRE_ACKNOWLEDGE

The pre-acknowledgement mode can be configured in the **standalone.xml** or **domain.xml** file on the connection factory.

```
<connection-factory name="ConnectionFactory">
  <connectors>
    <connector-ref connector-name="netty-connector"/>
  </connectors>
  <entries>
    <entry name="ConnectionFactory"/>
  </entries>
  <pre-acknowledge>true</pre-acknowledge>
</connection-factory>
```

Alternatively, to use pre-acknowledgement mode using the JMS API, create a JMS Session with the **HornetQSession.PRE_ACKNOWLEDGE** constant.

```
// messages will be acknowledge on the server *before* being delivered to the client
Session session = connection.createSession(false, HornetQJMSConstants.PRE_ACKNOWLEDGE);
```

Alternatively, you can set pre-acknowledge directly on the **HornetQConnectionFactory** instance using the setter method.

To use pre-acknowledgement mode using the core API you can set it directly on the **ClientSessionFactory** instance by using the setter method.

[Report a bug](#)

18.9.2. Individual Acknowledge

A valid use-case for individual acknowledgement is when you need to have your own scheduling and you do not know when your message processing will be finished. You prefer having one consumer per thread worker but this is not possible in some circumstances depending on how complex is your processing. For that you can use the individual acknowledgement.

You must setup **Individual Acknowledge** by creating a session with the acknowledge mode with **HornetQJMSConstants.INDIVIDUAL_ACKNOWLEDGE**. Individual Acknowledge inherits all the semantics from Client Acknowledge, with the exception the message is individually acknowledged.



NOTE

To avoid confusion on MDB processing, Individual ACKNOWLEDGE is not supported through MDBs (or the inbound resource adapter). This is because you have to finish the process of your message inside the MDB

[Report a bug](#)

18.10. THREAD MANAGEMENT

18.10.1. Server-Side Thread Management

Each HornetQ Server maintains a single thread pool for general use, and scheduled thread pool for scheduled use. A Java scheduled thread pool cannot be configured to use a standard thread pool, otherwise we could use a single thread pool for both scheduled and non scheduled activity.

When using old (blocking) IO, a separate thread pool is used to service connections. Since old IO requires a thread per connection, it is not recommended to get the threads from the standard pool as the pool will get exhausted if too many connections are made. This results in the server hanging, since it has no remaining threads to do anything else. If you require the server to handle many concurrent connections you must use NIO, not old IO.

When using new IO (NIO), by default, HornetQ uses a number of threads equal to three times the number of cores (or hyper-threads) as reported by `.getRuntime().availableProcessors()` for processing incoming packets. To override this value, you can set the number of threads by specifying the ***nio-remoting-threads*** parameter in the transport configuration.

[Report a bug](#)

18.10.1.1. Server Scheduled Thread Pool

The server scheduled thread pool is used for most activities on the server side that require running periodically or with delays. It maps internally to a **`java.util.concurrent.ScheduledThreadPoolExecutor`** instance.

The maximum number of thread used by this pool is configured in **`standalone.xml`** or **`domain.xml`** using the ***scheduled-thread-pool-max-size*** parameter. The default value is 5 threads. A small number of threads is usually sufficient for this pool.

[Report a bug](#)

18.10.1.2. General Purpose Server Thread Pool

The general purpose thread pool is used for most asynchronous actions on the server side. It maps internally to a **java.util.concurrent.ThreadPoolExecutor** instance.

The maximum number of thread used by this pool is configured in **standalone.xml** or **domain.xml** using the **thread-pool-max-size** parameter.

If a value of -1 is used, the thread pool has no upper bound and new threads are created on demand if there are not enough threads available to fulfill a request. If activity later subsides then threads are timed-out and closed.

If a value of *n*, where *n* is a positive integer greater than zero, is used then the thread pool is bounded. If more requests come in and there are no free threads are available in the pool and the pool is full then requests will block until a thread becomes available. It is recommended that a bounded thread pool is used with caution since it can lead to dead-lock situations if the upper bound is chosen to be too low.

The default value for **thread-pool-max-size** is 30.

[Report a bug](#)

18.10.1.3. Expiry Reaper Thread

A single thread is also used on the server side to scan for expired messages in queues. We cannot use either of the thread pools for this since this thread needs to run at its own configurable priority.

[Report a bug](#)

18.10.1.4. Asynchronous IO

Asynchronous IO has a thread pool for receiving and dispatching events out of the native layer. It is on a thread dump with the prefix **HornetQ-AIO-poller-pool**. HornetQ uses one thread per opened file on the journal (there is usually one).

There is also a single thread used to invoke writes on **libaio**. It is done to avoid context switching on **libaio** that would cause performance issues. This thread is found on a thread dump with the prefix **HornetQ-AIO-writer-pool**.

[Report a bug](#)

18.10.2. Client-Side Thread Management

On the client side, HornetQ maintains a single static scheduled thread pool and a single static general thread pool for use by all clients using the same classloader in that JVM instance.

The static scheduled thread pool has a maximum size of 5 threads, and the general purpose thread pool has an unbounded maximum size.

If required HornetQ can also be configured so that each **ClientSessionFactory** instance does not use these static pools but instead maintains its own scheduled and general purpose pool. Any sessions created from that **ClientSessionFactory** will use those pools instead.

To configure a **ClientSessionFactory** instance to use its own pools, use the appropriate setter methods immediately after creation. For example:

```
ServerLocator locator = HornetQClient.createServerLocatorWithoutHA(...)
```

```
ClientSessionFactory myFactory = locator.createClientSessionFactory();
myFactory.setUseGlobalPools(false);
myFactory.setScheduledThreadPoolMaxSize(10);
myFactory.setThreadPoolMaxSize(-1);
```

If you are using the JMS API, you can set the same parameters on the **ClientSessionFactory** and use it to create the **ConnectionFactory** instance. For example:

```
ConnectionFactory myConnectionFactory = HornetQJMSClient.createConnectionFactory(myFactory);
```

If you are using JNDI to instantiate **HornetQConnectionFactory** instances, you can also set these parameters in the **standalone.xml** or **domain.xml** file where you describe your connection factory. For example:

```
<connection-factory name="ConnectionFactory">
  <connectors>
    <connector-ref connector-name="netty"/>
  </connectors>
  <entries>
    <entry name="ConnectionFactory"/>
    <entry name="XAConnectionFactory"/>
  </entries>
  <use-global-pools>false</use-global-pools>
  <scheduled-thread-pool-max-size>10</scheduled-thread-pool-max-size>
  <thread-pool-max-size>-1</thread-pool-max-size>
</connection-factory>
```

[Report a bug](#)

18.11. MESSAGE GROUPING

18.11.1. About Message Grouping

A message group is a set/group of messages which share certain characteristics:

- All messages in a message group are grouped under a common group id. This means that they can be identified with a common group property
- All messages in a message group are serially processed and consumed by the same consumer irrespective of the number of customers on the queue. This means that a specific message group with a unique group id is always processed by one consumer when the consumer opens it. If the consumer closes the message group the entire message group is directed to another consumer in the queue



IMPORTANT

Message groups are especially useful when there is a need for messages with a certain value of the property (group id) to be processed serially by a single consumer.

[Report a bug](#)

18.11.2. Using HornetQ Core API on Client Side

The property **_HQ_GROUP_ID** is used to identify a message group in HornetQ Core API on client side. To pick a random unique message group identifier you can also set the **auto-group** property as **true** on the `SessionFactory`.

[Report a bug](#)

18.11.3. Configuring Server for Java Messaging Service (JMS) Clients

The property **JMSXGroupID** is used to identify a message group for Java Messaging Service (JMS) clients. If you wish to send a message group with different messages to one consumer you can set the same **JMSXGroupID** for different messages:

```
Message message = ...
message.setStringProperty("JMSXGroupID", "Group-0");
producer.send(message);

message = ...
message.setStringProperty("JMSXGroupID", "Group-0");
producer.send(message);
```

The second approach is to set the **auto-group** property to **true** on the **HornetQConnectionFactory**. The **HornetQConnectionFactory** will then pick up a random unique message group identifier. You can set the **auto-group** property in server configuration files (**standalone.xml** and **domain.xml**) as follows:

```
<connection-factory name="ConnectionFactory">
  <connectors>
    <connector-ref connector-name="netty-connector"/>
  </connectors>
  <entries>
    <entry name="ConnectionFactory"/>
  </entries>
  <auto-group>true</auto-group>
</connection-factory>
```

An alternative to the above methods is to set a specific message group identifier through the connection factory. This will in turn set the property **JMSXGroupID** to the specified value for all messages sent through this connection factory. To set a specific message group identifier on the connection factory, edit the **group-id** property in server configuration files (**standalone.xml** and **domain.xml**) as follows:

```
<connection-factory name="ConnectionFactory">
  <connectors>
    <connector-ref connector-name="netty-connector"/>
  </connectors>
  <entries>
    <entry name="ConnectionFactory"/>
  </entries>
  <group-id>Group-0</group-id>
</connection-factory>
```

[Report a bug](#)

18.11.4. Clustered Grouping



IMPORTANT

Clustered grouping is provided as [technology preview](#) only. It is not supported for use in a production environment and may be subject to significant future changes.

Clustered grouping follows a different approach relative to normal message grouping. In a cluster, message groups with specific group ids can arrive on any of the nodes. It is important for a node to determine which group ids are bound to which consumer on which node. Each node is responsible for routing message groups correctly to the node which has the consumer processing those group ids irrespective of where the message groups arrive by default. Once messages with a given group id are sent to a specific consumer connected to the given node in the cluster, then those messages are never sent to another node even if the consumer is disconnected.

This situation is addressed by a grouping handler. Each node has a grouping handler and this grouping handler (along with other handlers) is responsible for routing the message groups to the correct node. There are two types of grouping handlers namely **local** and **remote**.

The local handler is responsible for deciding the route which a message group should take. The remote handlers communicate with the local handler and work accordingly. Each cluster should choose a specific node to have a local grouping handler and all the other nodes should have remote handlers.



WARNING

If message grouping is used in cluster, it breaks if the server with configured remote grouping handler fails. Setting up backup for remote grouping handler also does not have affect.

You can configure "local" and "remote" grouping handlers in server configuration files (**standalone.xml** and **domain.xml**) as follows:

```
<grouping-handler name="my-grouping-handler">
  <type>LOCAL</type>
  <address>jms</address>
  <timeout>5000</timeout>
</grouping-handler>

<grouping-handler name="my-grouping-handler">
  <type>REMOTE</type>
  <address>jms</address>
  <timeout>5000</timeout>
</grouping-handler>
```

The "timeout" attribute ensures that a routing decision is made quickly within the specified time. If a decision is not made within this time an exception is thrown.

The node which initially receives a message group takes the routing decision based on regular cluster routing conditions (round-robin queue availability). The node proposes this decision to the respective grouping handler which then routes the messages to the proposed queue if it accepts the proposal.

If the grouping handler rejects the proposal, it proposes some other route and the routing takes place accordingly. The other nodes follow suite and forward the message groups to the chosen queue. After a message arrives on a queue it is pinned to a customer on that queue.

[Report a bug](#)

18.11.5. Best Practices for Clustered Grouping

Some best practices for clustered grouping are as follows:

- If you create and close consumers regularly make sure that your consumers are distributed evenly across the different nodes. Once a queue is pinned, messages are automatically transferred to that queue regardless of removing customers from it
- If you wish to remove a queue which has a message group bound to it, make sure the queue is deleted by the session that is sending the messages. Doing this will ensure that other nodes will not try to route messages to this queue after it is removed
- As a failover mechanism always replicate the node which has the local grouping handler

[Report a bug](#)

18.12. DUPLICATE MESSAGE DETECTION

18.12.1. About Duplicate Message Detection

Duplicate message detection allows filtering of duplicate messages without the need of coding the duplicate detection logic within the application. You can configure duplicate message detection in HornetQ.

When a sender(client/server) sends a message to another server there can be a situation where the target server(receiver) or the connection fails after sending the message but before sending a response to the sender indicating that the process was successful. In such situations, it is very difficult for the sender(client) to determine if the message was sent successfully to the intended receiver.

The message send may or may not be successful depending on when the target receiver or connection failed (before or after sending the message). If the sender (client/server) decides to resend the last message it can result in a duplicate message being sent to the address.

HornetQ provides duplicate message detection for messages sent to addresses.

[Report a bug](#)

18.12.2. Using Duplicate Message Detection for Sending Messages

To enable duplicate message detection for sent messages you need to set a special property on the message to a unique value. You can create this value the way you wish but this value must be unique.

When the target server receives this message, it checks if the special property is set. If the property is set then the target server checks its memory cache for a received message with that value of the header. If the server finds any message with the same value of the header it ignores the message sent by a client.

If you are sending messages in a transaction then you do not have to set the property for every message you send in that transaction; you only need to set it once in the transaction. If the server detects a duplicate message for any message in the transaction, then it will ignore the entire transaction.

The name of the property that you set is given by the value of ***org.hornetq.api.core.HDR_DUPLICATE_DETECTION_ID***, which is ***_HQ_DUPL_ID***. The value of this property can be of type **byte[]** or **SimpleString** for core API. For Java Messaging Service (JMS) clients, it must be of the type **String** with a unique value. An easy way of generating a unique id is by generating a UUID.

The following example shows how to set the property for core API:

```
...
ClientMessage message = session.createMessage(true);

SimpleString myUniqueID = "This is my unique id"; // Can use a UUID for this
message.setStringProperty(HDR_DUPLICATE_DETECTION_ID, myUniqueID);

...
```

The following example shows how to set the property for JMS clients:

```
...
Message jmsMessage = session.createMessage();

String myUniqueID = "This is my unique id"; // Could use a UUID for this
message.setStringProperty(HDR_DUPLICATE_DETECTION_ID.toString(), myUniqueID);

...
```

[Report a bug](#)

18.12.3. Configuring Duplicate ID Cache

The server maintains caches of received values of the ***org.hornetq.core.message.impl.HDR_DUPLICATE_DETECTION_ID*** property sent to each address. Each address maintains its own address cache.

The cache is fixed in terms of size. The maximum size of cache is configured using the parameter ***id-cache-size*** in server configuration files (**standalone.xml** and **domain.xml**). The default value of this parameter is 2000 elements. If the cache has a maximum size of *n* elements, then the (*n* + 1)th ID stored will overwrite the 0th element in the cache.

The caches can also be configured to persist to disk or not. This can be configured using the parameter ***persist-id-cache*** in server configuration files (**standalone.xml** and **domain.xml**). If this value is set "true" then each ID will be persisted to permanent storage as they are received. The default value for this parameter is true.

**NOTE**

Set the size of the duplicate ID cache to a large size in order to ensure that resending of messages does not overwrite the previously sent messages stored in the cache.

[Report a bug](#)

18.12.4. Using Duplicate Detection with Bridges and Cluster Connections

Core bridges can be configured to automatically add a unique duplicate ID value (if there isn't already one in the message) before forwarding the message to the target. To configure a core bridge for duplication message detection set the property ***use-duplicate-detection*** to "true" in server configuration files (**standalone.xml** and **domain.xml**). The default value of this parameter is "true".

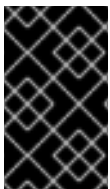
Cluster connections internally use core bridges to move messages between nodes of the cluster. To configure a cluster connection for duplicate message detection set the property ***use-duplicate-detection*** to "true" in server configuration files (**standalone.xml** and **domain.xml**). The default value of this parameter is "true".

[Report a bug](#)

18.13. JMS BRIDGES

18.13.1. About Bridges

The function of a bridge is to consume messages from a source queue, and forward them to a target address, typically on a different HornetQ server. Bridges cope with unreliable connections, automatically reconnecting when the connections become available again. HornetQ bridges can be configured with filter expressions to only forward certain messages.

**IMPORTANT**

JMS bridge cannot be deployed to EAP 6 server, which includes HornetQ configured as a dedicated backup. The reason is that Transaction Manager on a dedicated backup server is unable to recover transactions previously started on the HornetQ live server.

[Report a bug](#)

18.13.2. Create a JMS Bridge

Summary

A JMS bridge consumes messages from a source JMS queue or topic and sends them to a target JMS queue or topic, which is typically on a different server. It can be used to bridge messages between any JMS servers, as long as they are JMS 1.1 compliant. The source and destination JMS resources are looked up using JNDI and the client classes for the JNDI lookup must be bundled in a module. The module name is then declared in the JMS bridge configuration.

Procedure 18.2. Create a JMS Bridge

This procedure demonstrates how to configure a JMS bridge to migrate messages from a JBoss EAP 5.x server to a JBoss EAP 6 server.

1. Configure the Bridge On the Source JMS Messaging Server

Configure the JMS bridge on the source server using the instructions provided for that server type. For an example of how to configure a JMS Bridge for a JBoss EAP 5.x server, see the topic entitled *Create a JMS Bridge* in the *Migration Guide* for JBoss EAP 6.

2. Configure the Bridge on the Destination JBoss EAP 6 Server

In JBoss EAP 6.1 and later, the JMS bridge can be used to bridge messages from any JMS 1.1 compliant server. Because the source and target JMS resources are looked up using JNDI, the JNDI lookup classes of the source messaging provider, or message broker, must be bundled in a JBoss Module. The following steps use the fictitious 'MyCustomMQ' message broker as an example.

a. Create the JBoss module for the messaging provider.

- i. Create a directory structure under **EAP_HOME/modules/system/layers/base/** for the new module. The **main/** subdirectory will contain the client JARs and **module.xml** file. The following is an example of the directory structure created for the MyCustomMQ messaging provider:

EAP_HOME/modules/system/layers/base/org/mycustommq/main/

- ii. In the **main/** subdirectory, create a **module.xml** file containing the module definition for the messaging provider. The following is an example of the **module.xml** created for the MyCustomMQ messaging provider.

```
<?xml version="1.0" encoding="UTF-8"?>
<module xmlns="urn:jboss:module:1.1" name="org.mycustommq">
  <properties>
    <property name="jboss.api" value="private"/>
  </properties>

  <resources>
    <!-- Insert resources required to connect to the source or target -->
    <resource-root path="mycustommq-1.2.3.jar" />
    <resource-root path="mylogapi-0.0.1.jar" />
  </resources>

  <dependencies>
    <!-- Add the dependencies required by JMS Bridge code -->
    <module name="javax.api" />
    <module name="javax.jms.api" />
    <module name="javax.transaction.api"/>
    <!-- Add a dependency on the org.hornetq module since we send -->
    <!-- messages to the HornetQ server embedded in the local EAP instance -->
    <module name="org.hornetq" />
  </dependencies>
</module>
```

- iii. Copy the messaging provider JARs required for the JNDI lookup of the source resources to the module's **main/** subdirectory. The directory structure for the MyCustomMQ module should now look like the following.

```
modules/
  -- system
    -- layers
```

```

|-- base
  |-- org
    |-- mycustommq
      |-- main
        |-- mycustommq-1.2.3.jar
        |-- mylogapi-0.0.1.jar
        |-- module.xml

```

- b. Configure the JMS bridge in the **messaging** subsystem of the JBoss EAP 6 server.
 - i. Before you begin, stop the server and back up the current server configuration files. If you are running a standalone server, this is the ***EAP_HOME/standalone/configuration/standalone-full-ha.xml*** file. If you are running a managed domain, back up both the ***EAP_HOME/domain/configuration/domain.xml*** and the ***EAP_HOME/domain/configuration/host.xml*** files.
 - ii. Add the **jms-bridge** element to the **messaging** subsystem in the server configuration file. The **source** and **target** elements provide the names of the JMS resources used for JNDI lookups. If **user** and **password** credentials are specified, they are passed as arguments when JMS connection is created.

The following is an example of the **jms-bridge** element configured for the MyCustomMQ messaging provider:

```

<subsystem xmlns="urn:jboss:domain:messaging:1.3">
  ...
  <jms-bridge name="myBridge" module="org.mycustommq">
    <source>
      <connection-factory name="ConnectionFactory"/>
      <destination name="sourceQ"/>
      <user>user1</user>
      <password>pwd1</password>
      <context>
        <property key="java.naming.factory.initial"
value="org.mycustommq.jndi.MyCustomMQInitialContextFactory"/>
        <property key="java.naming.provider.url" value="tcp://127.0.0.1:9292"/>
      </context>
    </source>
    <target>
      <connection-factory name="java:/ConnectionFactory"/>
      <destination name="/jms/targetQ"/>
    </target>
    <quality-of-service>DUPLICATES_OK</quality-of-service>
    <failure-retry-interval>500</failure-retry-interval>
    <max-retries>1</max-retries>
    <max-batch-size>500</max-batch-size>
    <max-batch-time>500</max-batch-time>
    <add-messageID-in-header>true</add-messageID-in-header>
  </jms-bridge>
</subsystem>

```

In the above example, the JNDI properties are defined in the **context** element for the **source**. If the **context** element is omitted, as in the **target** example above, the JMS resources are looked up in the local instance.

18.14. PERSISTENCE

18.14.1. About Persistence in HornetQ

HornetQ handles its own persistence. It ships with a high-performance journal, which is optimized for messaging-specific use cases.

The HornetQ journal is append only with a configurable file size, which improves performance by enabling single write operations. It consists of a set of files on disk, which are initially pre-created to a fixed size and filled with padding. As server operations (add message, delete message, update message, etc.) are performed, records of the operations are appended to the journal until the journal file is full, at which point the next journal file is used.

A sophisticated garbage collection algorithm determines whether journal files can be reclaimed and re-used when all of their data has been deleted. A compaction algorithm removes dead space from journal files and compresses the data.

The journal also fully supports both local and XA transactions.

The majority of the journal is written in Java, but interaction with the file system has been abstracted to allow different pluggable implementations. The two implementations shipped with HornetQ are:

- *Java New I/O (NIO)*

Uses standard Java NIO to interface with the file system. This provides extremely good performance and runs on any platform with a Java 6 or later runtime.

- *Linux Asynchronous IO (AIO)*

Uses a native code wrapper to talk to the Linux asynchronous IO library (AIO). With AIO, HornetQ receives a message when data has been persisted. This removes the need for explicit syncs. AIO will typically provide better performance than Java NIO, but requires Linux kernel 2.6 or later and the libaio package.

AIO also requires ext2, ext3, ext4, jfs or xfs type file systems.

The standard HornetQ core server uses the following journal instances:

- *bindings journal*

Stores bindings-related data, including the set of queues deployed on the server and their attributes. It also stores data such as ID sequence counters. The bindings journal is always a NIO journal, as it typically has low throughput in comparison to the message journal.

The files on this journal are prefixed as hornetq-bindings. Each file has a bindings extension. File size is 1048576 bytes, and it is located in the bindings folder.

- *JMS journal*

Stores all JMS-related data, for example, any JMS queues, topics or connection factories and any JNDI bindings for these resources. Any JMS resources created with the management API are persisted to this journal. Any resources configured with configuration files are not. This journal is created only if JMS is in use.

- *message journal*

Stores all message-related data, including messages themselves and duplicate-id caches. By default, HornetQ uses AIO for this journal. If AIO is not available, it will automatically fall back to NIO.

Large messages are persisted outside the message journal. In low memory situations, configure HornetQ to page messages to disk. If persistence is not required, HornetQ can be configured not to persist any data.



NOTE

Running JBoss EAP server with installed HornetQ natives and journal type set to AsyncIO causes error when running on **tmpfs** filesystem.

[Report a bug](#)

18.14.2. Import or Export the Journal Data

You may want to inspect the existent records on each one of the journals used by HornetQ, and you can use the export/import tool for that purpose. The export/import are classes located in the **EAP_HOME/bin/client/jboss-client.jar**, you can export the journal as a text file by using this command: **java -cp jboss-client.jar org.hornetq.core.journal.impl.ExportJournal <JournalDirectory> <JournalPrefix> <FileExtension> <FileSize> <FileOutput>**

To import the file as binary data on the journal: **java -cp jboss-client.jar org.hornetq.core.journal.impl.ImportJournal <JournalDirectory> <JournalPrefix> <FileExtension> <FileSize> <FileInput>**

- JournalDirectory: Use the configured folder for your selected folder.
Example: *./hornetq/data/journal*
- JournalPrefix: Use the prefix for your selected journal, as discussed
- FileExtension: Use the extension for your selected journal, as discussed
- FileSize: Use the size for your selected journal, as discussed
- FileOutput: text file that will contain the exported data

[Report a bug](#)

18.15. HORNETQ CLUSTERING

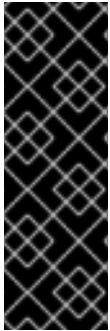
HornetQ clusters are used to create groups of HornetQ servers in order to share message processing load. Each active node in the cluster acts as an independent HornetQ server and manages its own messages and connections.

To form a cluster, each node (independent HornetQ server) declares cluster connections with another node with configuration parameters in server configuration files (**standalone.xml** and **domain.xml**).

In clustering, core bridges are used for bridging/routing messages from one cluster to another. Core bridges consume messages from a source queue and then forward these messages to a target HornetQ server (node) which may or may not be in the same cluster.

When a node forms a cluster connection with another node, it creates a core bridge internally. Each node creates an explicit core bridge and you do not need to declare it. These cluster connections allow transmission of messages between nodes in various clusters for balancing message processing load.

You can configure cluster nodes in server configuration files (**standalone.xml** and **domain.xml**).



IMPORTANT

You can configure a node through server configuration files (**standalone.xml** and **domain.xml**) and copy this configuration to other nodes to generate a symmetric cluster. However you must be careful when you are copying the server configuration files. You must not copy the HornetQ data (i.e. the bindings, journal, and large messages directories) from one node to another. When a node is started for the first time it persists a unique identifier to the journal directory which is needed for proper formation of clusters.

[Report a bug](#)

18.15.1. About Server Discovery

Servers use a mechanism called "server discovery" to:

- Forward their connection details to messaging clients: Messaging clients intend to connect to servers of a cluster without specific details on the servers which are up and running at a given point of time
- Connect to other servers: Servers in a cluster want to establish cluster connections with other servers without specific details on of all other servers in a cluster

Information about servers is sent to messaging clients via normal HornetQ connections and to other servers via cluster connections.

The initial first connection needs to be established and it can be established using dynamic server discovery techniques like UDP (User Datagram Protocol), JGroups or by providing a list of connectors.

[Report a bug](#)

18.15.2. Broadcast Groups

Connectors are used on the client to define how and in what ways it connects to the server. Servers use broadcast groups to broadcast connectors over the network. The broadcast group takes a set of connector pairs and broadcasts them on the network. Each connector pair contains connection settings for a live and backup server.

You can define broadcast groups in **broadcast-groups** element of server configuration files (**standalone.xml** and **domain.xml**). A single HornetQ server can have many broadcast groups. You can define either a User Datagram Protocol (UDP) or a JGroup broadcast group.

[Report a bug](#)

18.15.2.1. User Datagram Protocol (UDP) Broadcast Group

The example shown below defines a UDP broadcast group:

```
<broadcast-groups>
```

```

<broadcast-group name="my-broadcast-group">
  <local-bind-address>172.16.9.3</local-bind-address>
  <local-bind-port>5432</local-bind-port>
  <group-address>231.7.7.7</group-address>
  <group-port>9876</group-port>
  <broadcast-period>2000</broadcast-period>
  <connector-ref>netty</connector-ref>
</broadcast-group>
</broadcast-groups>

```

**NOTE**

In the configuration example shown above, the attributes "local-bind-address", "local-bind-port", "group-address" and "group-port" are deprecated. Instead of these attributes you can choose to use the attribute "socket-binding".

The example shown below defines a UDP broadcast group replacing all the deprecated attributes with the attribute "socket-binding":

```

<broadcast-groups>
  <broadcast-group name="my-broadcast-group">
    <socket-binding>messaging-group</socket-binding>
    <broadcast-period>2000</broadcast-period>
    <connector-ref>netty</connector-ref>
  </broadcast-group>
</broadcast-groups>

```

The table shown below describes all the important parameters used in the above examples and in general to define a UDP broadcast group:

Table 18.11. UDP Broadcast Group Parameters

Attribute	Description
name attribute	Denotes the name of each broadcast group in a server. Each broadcast group must have a unique name.
local-bind-address	[Deprecated] This is a UDP specific attribute and specifies the local bind address which the datagram packet binds to. You must set this property to define the interface which you wish to use for your broadcasts. If this property is not specified then the socket binds to a wildcard address (a random kernel generated address).
local-bind-port	[Deprecated] This is a UDP specific attribute and is used to specify a local port which the datagram socket binds to. A default value of "-1" specifies an anonymous port to be used.

Attribute	Description
group-address	[Deprecated] This is a multicast address specific to UDP where messages are broadcast. This IP address has a range of 224.0.0.0 to 239.255.255.255, inclusive. The IP address 224.0.0 is reserved and can not be used.
group-port	[Deprecated] This denotes the UDP port number for broadcasting.
socket-binding	This denotes the broadcast group socket binding
broadcast-period	This parameter specifies the time between two broadcasts (milliseconds). It is optional.
connector-ref	This refers to the connector which will be broadcasted.

[Report a bug](#)

18.15.2.2. JGroups Broadcast Group

You can use JGroups to broadcast by specifying two attributes namely *jgroups-stack* and *jgroups-channel*. The example shown below defines a JGroups broadcast group:

```
<broadcast-groups>
  <broadcast-group name="bg-group1">
    <jgroups-stack>udp</jgroups-stack>
    <jgroups-channel>udp</jgroups-channel>
    <broadcast-period>2000</broadcast-period>
    <connector-ref>netty</connector-ref>
  </broadcast-group>
</broadcast-groups>
```

The JGroups broadcast group definition uses two main attributes:

- *jgroups-stack* attribute: This denotes the name of a stack defined in the **org.jboss.as.clustering.jgroups** subsystem
- *jgroups-channel* attribute: This denotes the channel which JGroups channels connect to for broadcasting

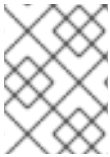
[Report a bug](#)

18.15.3. Discovery Groups

Broadcast groups are used for broadcasting connectors over a network. On the other hand discovery groups define how connector information is received from broadcast endpoints (UDP or JGroups broadcast group). A discovery group maintains a list of connector pair- one for each broadcast by a different server.

When a discovery group receives broadcasts on a broadcast endpoint for a specific server, it accordingly updates the connector pairs entry in the list for the specific server. If it does not receive a broadcast from a specific server for a long time, it removes the server's entry from the list altogether.

Discovery groups are mainly used by cluster connections and Java Messaging Service (JMS) clients to obtain initial connection information in order to download the required topology.



NOTE

You must configure each discovery group with an appropriate broadcast endpoint which matches its broadcast group counterpart (UDP or JGroups).

[Report a bug](#)

18.15.3.1. Configuring User Datagram Protocol (UDP) Discovery Group on the Server

The example shown below defines a UDP discovery group:

```
<discovery-groups>
  <discovery-group name="my-discovery-group">
    <local-bind-address>172.16.9.7</local-bind-address>
    <group-address>231.7.7.7</group-address>
    <group-port>9876</group-port>
    <refresh-timeout>10000</refresh-timeout>
  </discovery-group>
</discovery-groups>
```



NOTE

In the configuration example shown above, the attributes "local-bind-address", "group-address" and "group-port" are deprecated. Instead of these attributes you can choose to use the attribute "socket-binding".

The example shown below defines a UDP discovery group replacing all the deprecated attributes with the attribute "socket-binding":

```
<discovery-groups>
  <discovery-group name="my-discovery-group">
    <socket-binding>messaging-group</socket-binding>
    <refresh-timeout>10000</refresh-timeout>
  </discovery-group>
</discovery-groups>
```

The table shown below describes all the important parameters used in the above example and in general to define a discovery group:

Table 18.12. UDP Discovery Group Parameters

Attribute	Description
name attribute	This attribute denotes the name of your discovery group. Each discovery name must have a unique name per server.

Attribute	Description
local-bind-address	[Deprecated] This is an optional UDP specific attribute. It is used to configure a discovery group to listen on a specific interface when using multiple interfaces on the same machine.
group-address	[Deprecated] This is a compulsory UDP specific attribute. It is used to configure a discovery group to listen on the multicast IP address of a group. The value of this attribute must match the group-address attribute of the broadcast group that you wish to listen from.
group-port	[Deprecated] This is a compulsory UDP specific attribute. It is used to configure the UDP port of the multicast group. The value of this attribute must match the group-port attribute of the multicast group that you wish to listen from.
socket-binding	This denotes the discovery group socket binding
refresh-timeout	This is an optional UDP specific attribute. It is used to configure the time period (in milliseconds) for which the discovery group waits before removing a server's connector pair entry from the list after receiving the last broadcast from that server. The value of refresh-timeout must be set significantly higher than the value of broadcast-period attribute on the broadcast group to prevent quick removal servers from the list when the broadcast process is still on. The default value of this attribute is 10,000 milliseconds.

[Report a bug](#)

18.15.3.2. Configuring JGroups Discovery Group on the Server

The example shown below defines a JGroups discovery group:

```
<discovery-groups>
  <discovery-group name="dg-group1">
    <jgroups-stack>udp</jgroups-stack>
    <jgroups-channel>udp</jgroups-channel>
    <refresh-timeout>10000</refresh-timeout>
  </discovery-group>
</discovery-groups>
```

The JGroups discovery group definition uses two main attributes:

- **jgroups-stack** attribute: This denotes the name of a stack defined in the **org.jboss.as.clustering.jgroups** subsystem

- ***jgroups-channel*** attribute: This attribute denotes the channel which JGroups channels connect to for receiving broadcasts



NOTE

JGroup attributes and UDP specific attributes are exclusive. You can use either JGroup or UDP set of attributes in the configuration of a discovery group or a broadcast group

[Report a bug](#)

18.15.3.3. Configuring Discovery Groups for Java Messaging Service (JMS) Clients

Discovery groups can be configured for JMS and core clients. You can specify the discovery group to be used for a JMS connection factory in server configuration files (**standalone.xml** and **domain.xml**):

```
<connection-factory name="ConnectionFactory">
  <discovery-group-ref discovery-group-name="my-discovery-group"/>
  <entries>
    <entry name="ConnectionFactory"/>
  </entries>
</connection-factory>
```

The element ***discovery-group-ref*** is used to specify the name of a discovery group. When a client application downloads this connection factory from Java Naming and Directory Interface (JNDI) and creates JMS connections, these connections are load balanced across all the servers which the discovery group maintains by listening on the multicast address specified in the discovery group configuration.

If you are using JMS but not JNDI to lookup for a connection factory then you can specify the discovery group parameters directly when creating the JMS connection factory:

```
final String groupAddress = "231.7.7.7";
final int groupPort = 9876;
ConnectionFactory jmsConnectionFactory = HornetQJMSClient.createConnectionFactory(new
DiscoveryGroupConfiguration(groupAddress, groupPort, new
UDPBroadcastGroupConfiguration(groupAddress, groupPort, null, -1)), JMSFactoryType.CF);
Connection jmsConnection1 = jmsConnectionFactory.createConnection();
Connection jmsConnection2 = jmsConnectionFactory.createConnection();
```

The default value of ***refresh-timeout*** attribute can be set on `DiscoveryGroupConfiguration` by using the setter method **`setDiscoveryRefreshTimeout()`**. For the connection factory to wait for a specific amount of time before creating the first connection, you can use the setter method **`setDiscoveryInitialWaitTimeout()`** on `DiscoveryGroupConfiguration`.

Doing this ensures that the connection factory has enough time to receive broadcasts from all the nodes in the cluster. The default value for this parameter is 10000 milliseconds.

[Report a bug](#)

18.15.3.4. Configuring discovery for Core API

If you are using the core API to directly instantiate ***ClientSessionFactory*** instances, then you can specify the discovery group parameters directly when creating the session factory:

```

final String groupAddress = "231.7.7.7";
final int groupPort = 9876;
ServerLocator factory = HornetQClient.createServerLocatorWithHA(new
DiscoveryGroupConfiguration(groupAddress, groupPort, new
UDPBroadcastGroupConfiguration(groupAddress, groupPort, null, -1)));
ClientSessionFactory factory = locator.createSessionFactory();
ClientSession session1 = factory.createSession();
ClientSession session2 = factory.createSession();

```

The default value of *refresh-timeout* attribute can be set on `DiscoveryGroupConfiguration` by using the setter method `setDiscoveryRefreshTimeout()`. You can use `setDiscoveryInitialWaitTimeout()` on `DiscoveryGroupConfiguration` for the session factory to wait for a specific amount of time before creating a session.

[Report a bug](#)

18.15.4. Server Side Load Balancing

There is one important cluster topology:

- Symmetric Cluster: In a symmetric cluster every cluster node is connected directly to every other node in the cluster. To create a symmetric cluster every node in the cluster defines a cluster connection with the attribute *max-hops* set to 1.



NOTE

In a symmetric cluster each node knows about all the queues that exist on all the other nodes and what consumers they have. With this knowledge it can determine how to load balance and redistribute messages around the nodes.

[Report a bug](#)

18.15.4.1. Configuring Cluster Connections

Cluster connections are configured in server configuration files (`standalone.xml` and `domain.xml`) in the element *cluster-connection*. There can be zero or more cluster connections defined per HornetQ server.

```

<cluster-connections>
  <cluster-connection name="my-cluster">
    <address>jms</address>
    <connector-ref>netty-connector</connector-ref>
    <check-period>1000</check-period>
    <connection-ttl>5000</connection-ttl>
    <min-large-message-size>50000</min-large-message-size>
    <call-timeout>5000</call-timeout>
    <retry-interval>500</retry-interval>
    <retry-interval-multiplier>1.0</retry-interval-multiplier>
    <max-retry-interval>5000</max-retry-interval>
    <reconnect-attempts>1</reconnect-attempts>
    <use-duplicate-detection>true</use-duplicate-detection>
    <forward-when-no-consumers>>false</forward-when-no-consumers>
    <max-hops>1</max-hops>
    <confirmation-window-size>32000</confirmation-window-size>
  </cluster-connection>
</cluster-connections>

```

```

<call-failover-timeout>30000</call-failover-timeout>
<notification-interval>1000</notification-interval>
<notification-attempts>2</notification-attempts>
  <discovery-group-ref discovery-group-name="my-discovery-group"/>
</cluster-connection>
</cluster-connections>

```

The following table defines the configurable attributes:

Table 18.13. Cluster Connections Configurable Attributes

Attribute	Description	Default
<i>address</i>	Each cluster connection only applies to messages sent to an address that starts with this value. The address can be any value and you can have many cluster connections with different values of addresses, simultaneously balancing messages for those addresses, potentially to different clusters of servers. This does not use wild card matching.	
<i>connector-ref</i>	This is a compulsory attribute which refers to the connector sent to other nodes in the cluster so that they have the correct cluster topology	
<i>check-period</i>	This refers to the time period (in milliseconds) which is used to verify if a cluster connection has failed to receive pings from another server	30,000 milliseconds
<i>connection-ttl</i>	This specifies how long a cluster connection must stay alive if it stops receiving messages from a specific node in the cluster	60,000 milliseconds
<i>min-large-message-size</i>	If the message size (in bytes) is larger than this value then it will be split into multiple segments when sent over the network to other cluster members	102400 bytes
<i>call-timeout</i>	This specifies the time period (milliseconds) for which a packet sent over a cluster connection waits (for a reply) before throwing an exception	30,000 milliseconds

Attribute	Description	Default
<i>retry-interval</i>	If the cluster connection is created between nodes of a cluster and the target node has not been started, or is being rebooted, then the cluster connections from other nodes will retry connecting to the target until it comes back up. The parameter <i>retry-interval</i> defines the interval (milliseconds) between retry attempts	500 milliseconds
<i>retry-interval-multiplier</i>	This is used to increment the <i>retry-interval</i> after each retry attempt	1
<i>max-retry-interval</i>	This refers to the maximum delay (in milliseconds) for retries	2000 milliseconds
<i>reconnect-attempts</i>	This defines the number of times the system will try to connect a node on the cluster	-1 (infinite retries)
<i>use-duplicate-detection</i>	Cluster connections use bridges to link the nodes, and bridges can be configured to add a duplicate id property in each message that is forwarded. If the target node of the bridge crashes and then recovers, messages might be resent from the source node. By enabling duplicate detection any duplicate messages will be filtered out and ignored on receipt at the target node.	True
<i>forward-when-no-consumers</i>	This parameter determines whether or not messages will be distributed in a round robin fashion between other nodes of the cluster regardless of whether there are matching or indeed any consumers on other nodes	False
<i>max-hops</i>	This determines how messages are load balanced to other HornetQ servers which are connected to this server	-1

Attribute	Description	Default
<i>confirmation-window-size</i>	The size (in bytes) of the window used for sending confirmations from the server connected to	1048576
<i>call-failover-timeout</i>	This is used when a call is made during a failover attempt	-1 (no timeout)
<i>notification-interval</i>	This determines how often (in milliseconds) the cluster connection must broadcast itself when attaching to the cluster	1000 milliseconds
<i>notification-attempts</i>	This defines as to how many times the cluster connection must broadcast itself when connecting to the cluster	2
<i>discovery-group-ref</i>	This parameter determines which discovery group is used to obtain the list of other servers in the cluster which the current cluster connection will make connections to	

When creating connections between nodes of a cluster to form a cluster connection, HornetQ uses a cluster user and cluster password which is defined in server configuration files (**standalone.xml** and **domain.xml**):

```
<cluster-user>HORNETQ.CLUSTER.ADMIN.USER</cluster-user>
<cluster-password>NEW USER</cluster-password>
```



WARNING

It is important to change the default values of these credentials to prevent remote clients from making connections to the server using the default values.

[Report a bug](#)

18.16. HIGH AVAILABILITY

18.16.1. High Availability Introduction

HornetQ supports the ability to continue functioning after failure of one or more of the servers. Part of this is achieved through failover support where client connections migrate from the live server to a

backup server in the event of the live server failing. To keep the backup server current, messages are replicated from the live server to the backup server continuously through two strategies: shared store and replication.

There are two types of high-availability Topologies:

- **Dedicated Topology:** This topology comprises of two EAP servers. In the first server HornetQ is configured as a live server. In the second server HornetQ is configured as a backup server. The EAP server which has HornetQ configured as a backup server, acts only as a container for HornetQ. This server is inactive and can not host deployments like EJBs, MDBs or Servlets.
- **Collocated Topology:** This topology contains two EAP servers. Each EAP server contains two HornetQ servers (a live server and a backup server). The HornetQ live server on first EAP server and the HornetQ backup server on the second EAP server form a live backup pair. Whereas the HornetQ live server on the second EAP server and the HornetQ backup server on the first EAP server form another live backup pair.

In collocated topology, as soon as a live HornetQ server (part of live-backup pair) fails, the backup HornetQ server takes up and becomes active. When the backup HornetQ server shuts down in case of failback then destinations and connection factories configured in the backup server are unbound from JNDI (Java Naming and Directory Interface).

Java Naming and Directory Interface is shared with the other live HornetQ server (part of the other live-backup pair). Therefore unbounding of destinations and connection factories from JNDI also unbounds destinations and connection factories for this live HornetQ server.



IMPORTANT

Configuration of collocated backup servers cannot contain configuration of destinations or connection factories.



NOTE

The following information references **standalone-full-ha.xml**. The configuration changes can be applied to **standalone-full-ha.xml**, or any configuration files derived from it.

[Report a bug](#)

18.16.2. About HornetQ Shared Stores

When using a shared store, both the live and backup servers share the same, entire data directory, using a shared file system. This includes the paging directory, journal directory, large messages, and the binding journal. When failover occurs and the backup server takes over, it will load the persistent storage from the shared file system. Clients can then connect to it.

This form of high-availability differs from data replication, as it requires a shared file system accessible by both the live and backup nodes. This will usually be a high performance Storage Area Network (SAN) of some kind.

The advantage of shared store high-availability is that no replication occurs between the live and backup nodes. This means it does not suffer any performance penalties due to the overhead of replication during normal operation.

The disadvantage of shared store replication is that it requires a shared file system, and when the backup server activates it must load the journal from the shared store. This can take some time, depending on the amount of data in the store.

If the highest performance during normal operation is required, there is access to a fast SAN, and a slightly slower failover rate is acceptable (depending on the amount of data), shared store high-availability is recommended.



NOTE

HornetQ's data replication mechanism would replicate JMS data and not replicate bindings.

[Report a bug](#)

18.16.3. About HornetQ Storage Configurations

HornetQ supports shared storage when using the Red Hat Enterprise Linux version of NFSv4, either ASYNCIO or NIO journal type, for shared storage. The Red Hat Enterprise Linux NFS implementation supports both direct I/O (opening files with the O_DIRECT flag set), and kernel based asynchronous I/O. When configuring NFS for shared storage, it is recommended to use a highly-available NFS configuration.



IMPORTANT

When using the Red Hat Enterprise Linux NFSv4 as a shared storage option, the client cache must be disabled.

[Report a bug](#)

18.16.4. About HornetQ Journal Types

Two journal types are available for HornetQ:

- ASYNCIO
- NIO

The ASYNCIO journal type, also known as AIO, is a thin native code wrapper around the Linux asynchronous IO library (AIO). Using native functionality can provide better performance than NIO. This journal type is only supported on Red Hat Enterprise Linux and requires that **libaio** and the Native Components package are installed where JBoss EAP 6 is running. See the *Installation Guide* for instructions on installing the Native Components package.



IMPORTANT

Check the server log after JBoss EAP 6 is started, to ensure that the native library successfully loaded, and that the ASYNCIO journal type is being used. If the native library fails to load, HornetQ will revert to the NIO journal type, and this will be stated in the server log.

The NIO journal type uses standard Java NIO to interface with the file system. It provides very good performance and runs on all supported platforms.

To specify the HornetQ journal type, set the parameter `<journal-type>` in the **Messaging** subsystem.

[Report a bug](#)

18.16.5. Configuring HornetQ for Dedicated Topology with Shared Store

To configure the live and backup servers for shared store in dedicated topology, configure the **standalone-X.xml** files on each server to have the following:

```
<shared-store>true</shared-store>
<paging-directory path="${shared.directory}/paging"/>
<bindings-directory path="${shared.directory}/bindings"/>
<journal-directory path="${shared.directory}/journal"/>
<large-messages-directory path="${shared.directory}/large-messages"/>
.
.
.
<cluster-connections>
  <cluster-connection name="my-cluster">
    ...
  </cluster-connection>
</cluster-connections>
```

Table 18.14. HornetQ Servers Setup Attributes (for both live and backup servers)

Attribute	Description
shared-store	Whether this server is using shared store or not. Default is false
paging-directory path	This indicates the path to the paging directory. This path is the same for both live and backup servers as they share this directory
bindings-directory path	This indicates the path to the binding journal. This path is the same for both live and backup servers as they share this journal
journal-directory path	This indicates the path to the journal directory. This path is the same for both live and backup servers as they share this directory
large-messages-directory path	This indicates the path to the large messages directory. This path is the same for both live and backup servers as they share this directory
failover-on-shutdown	Whether this server becomes active when live or currently active backup server shuts down

The backup server must also be flagged explicitly as a backup.

```
<backup>true</backup>
```

The setup attribute exclusively for HornetQ backup server is: **allow-failback**. This specifies whether the backup server will automatically shutdown if the original live server comes back up.

[Report a bug](#)

18.16.6. HornetQ Message Replication



WARNING

Only persistent messages are replicated. Any non-persistent messages do not survive failover.

Message replication between a live and a backup server is achieved via network traffic as the live and backup servers do not share the same data stores. All the journals are replicated between the two servers as long as the two servers are within the same cluster and have the same cluster username and password. All persistent data traffic received by the live server gets replicated to the backup server.

When the backup server comes online, it looks for and connects to a live server to attempt synchronization. While it is synchronizing, it is unavailable as a backup server. Synchronization can take a long time depending on the amount of data to be synchronized and the network speed. If the backup server comes online and no live server is available, the backup server will wait until the live server is available in the cluster.

To enable servers to replicate data, a link must be defined between them in the **standalone-full-ha.xml** file. A backup server will only replicate with a live server with the same group name. The group name must be defined in the **backup-group-name** parameter in the **standalone-full-ha.xml** file on each server.

In the event of a live server failing, the correctly configured and fully synchronized backup server takes over its duties. The backup server will activate only if the live server has failed and the backup server is able to connect to more than half of the servers in the cluster. If more than half of the other servers in the cluster also fail to respond it would indicate a general network failure and the backup server will wait to retry the connection to the live server.

To get to the original state after failover, it is necessary to start the live server and wait until it is fully synchronized with the backup server. When this has been achieved, you can shutdown the backup server for the original live server to activate again. This happens automatically if the **allow-failback** attribute is set to true.

[Report a bug](#)

18.16.7. Configuring the HornetQ Servers for Replication

To configure the live and backup servers to be a replicating pair, configure the **standalone-full-ha.xml** files on each server to have the following settings:

```
<shared-store>>false</shared-store>
<backup-group-name>NameOfLiveBackupPair</backup-group-name>
<check-for-live-server>>true</check-for-live-server>
```

```

.
.
<cluster-connections>
  <cluster-connection name="my-cluster">
    ...
  </cluster-connection>
</cluster-connections>

```

**WARNING**

Administrators must take care not to mix settings for shared store and replicated configurations. For example, the **backup-group-name** attribute, which is used for replication, should not be set when **shared-store** is set to **true**, which indicates a shared store.

Table 18.15. HornetQ Replicating Setup Attributes

Attribute	Description
shared-store	Whether this server is using shared store or not. This value should be set to false for a replicated configuration. Default is false.
backup-group-name	This is the unique name which identifies a live/backup pair that should replicate with each other
check-for-live-server	If a replicated live server should check the current cluster to see if there is already a live server with the same node id. Default is false.
failover-on-shutdown	Whether this backup server (if it is a backup server) becomes the live server on a normal server shutdown. Default is false.

The backup server must also be flagged explicitly as a backup.

```
<backup>true</backup>
```

Table 18.16. HornetQ Backup Server Setup Attributes

Attribute	Description
allow-failback	Whether this server will automatically shutdown if the original live server comes back up. Default is true.

Attribute	Description
max-saved-replicated-journal-size	The maximum number of backup journals to keep after failback occurs. Specifying this attribute is only necessary if allow-failback is true. Default value is 2, which means that after 2 failbacks the backup server must be restarted in order to be able to replicate journal from live server and become backup again.

[Report a bug](#)

18.16.8. About High-availability (HA) Failover

High-availability failover is available with either automatic client failover, or application-level failover, through a live-backup structure. Each live server has a backup server. Only one backup per live server is supported.

The backup server only takes over if the live server crashes and there is a failover. After the live server has been restarted, and if the **allow-failback** attribute is set to true, it becomes the live server again. When the original live server takes over, the backup server reverts to being backup for the live server.



IMPORTANT

Clustering should be enabled even if you are not using the clustering capabilities. This is because each node of the HA cluster must have a cluster-connection to all of the other nodes, in order to negotiate roles with the other servers.

High availability cluster topology is achieved by the live and backup server as they send information about their connection details using IP multicasts. If IP multicasts can not be used, it is also possible to use a static configuration of the initial connections. After the initial connection, the client is informed about the topology. If the current connection is stale, the client establishes a new connection to another node.

After a live server has failed and a backup server has taken over, you will need to restart the live server and have clients fail back. To do this, restart the original live server and kill the new live server. You can do this by killing the process itself or wait for the server to crash on its own. You can also cause failover to occur on normal server shutdown, to enable this set the **failover-on-shutdown** property to true in the **standalone.xml** configuration file:

```
<failover-on-shutdown>true</failover-on-shutdown>
```

By default, the **failover-on-shutdown** property is set to false.

You can also force the new live server to shutdown when the old live server comes back up allowing the original live server to take over automatically by setting the **allow-failback** property to true in the **standalone.xml** configuration file:

```
<allow-failback>true</allow-failback>
```

In replication HA mode, to force the new live server to shutdown when the old live server comes back, set the **check-for-live-server** property to true in **standalone.xml** configuration file:

-

```
<check-for-live-server>true</check-for-live-server>
```

[Report a bug](#)

18.16.9. Deployments on HornetQ Backup Servers

In a dedicated HA environment, a JBoss EAP 6 server with HornetQ configured as a backup must not be used to host any deployments which use or connect to the HornetQ backup on that server. This includes deployments such as Enterprise Java Beans (Stateless Session Beans, Message Driven Beans), or servlets.

If a JBoss EAP 6 server has a HornetQ collocated backup configuration (where in the messaging subsystem there is a HornetQ server configured as 'live' and another HornetQ server configured as backup), then the JBoss EAP 6 server can host deployments as long as they are configured to connect to the 'live' HornetQ server.

[Report a bug](#)

18.16.10. HornetQ Failover Modes

HornetQ defines two types of client failover:

- Automatic client failover
- Application-level client failover

HornetQ provides transparent automatic reattachment of connections to the same server, for example, in case of transient network problems. This is similar to failover, except it is reconnecting to the same server.

During failover, if the client has consumers on any non persistent or temporary queues, those queues are automatically recreated during failover on the backup node, since the backup node does not have any information about non persistent queues.

[Report a bug](#)

18.16.11. Automatic Client Failover

HornetQ clients can be configured to receive information about live and backup servers, this information helps in event of client connection failure - live server connection, the client detects failover and reconnects to the backup server. The backup server automatically recreates any sessions and consumers that existed on each connection before failover, thus saving the user from having to hand-code manual reconnection logic.

HornetQ clients detect connection failure if packets are not received from the server within the time specified in **client-failure-check-period**. If the client does not receive data in time, the client assumes the connection has failed and attempts failover. If the socket is closed by the operating system, the server process is killed rather than the machine itself crashing, then the client immediately initiates failover.

HornetQ clients can be configured in different ways to discover the list of live-backup server groups. The client can be configured explicitly or use server discovery for the client to automatically discover the list. Alternatively, the clients can explicitly connect to a specific server and download the current servers and backups.

To enable automatic client failover, the client must be configured to allow non-zero reconnection attempts.

By default, failover only occurs after at least one connection has been made to the live server. The client retries connecting to the live server as specified in the **reconnect-attempts** property and fails after the specified number of attempts.

[Report a bug](#)

18.16.12. Application-Level Failover

In some cases, as per your requirement, you could handle any connection failure manually by specifying reconnection logic in a custom failure handler. You can define this as application-level failover, since the failover is handled at the user application level.

To implement application-level failover, if you are using JMS, you need to set an `ExceptionListener` class on the JMS connection. If a connection failure is detected, the `ExceptionListener` class is called by HornetQ. In your `ExceptionListener`, close the old JMS connections, look up for new connection factory instances from JNDI and create new connections.

If you are using the core API, then the procedure is very similar: set a `FailureListener` on the core `ClientSession` instances.

[Report a bug](#)

18.17. PERFORMANCE TUNING

18.17.1. Tuning Persistence

- Put the message journal on its own physical volume. If the disk is shared with other processes, for example transaction co-ordinator, database or other journals, which are also reading and writing from it, then this may greatly reduce performance since the disk head may be skipping between the different files. One of the advantages of an append only journal is that disk head movement is minimized. This advantage is lost if the disk is shared. If you are using paging or large messages, make sure they are put on separate volumes too.
- Minimum number of journal files. Set ***journal-min-files*** parameter to a number of files that would fit your average sustainable rate. If you see new files being created on the journal data directory too often, that is, lots of data is being persisted, you need to increase the minimal number of files, this way the journal would reuse more files instead of creating new data files.
- Journal file size. The journal file size must be aligned to the capacity of a cylinder on the disk. The default value of 10MiB should be enough on most systems.
- Use AIO journal. For Linux operating system, keep your journal type as **AIO**. **AIO** will scale better than Java NIO.
- Tune ***journal-buffer-timeout***. The timeout can be increased to increase throughput at the expense of latency.
- If you are running AIO you might be able to get improved performance by increasing ***journal-max-io*** parameter value. Do not change this parameter if you are running NIO.

[Report a bug](#)

18.17.2. Tuning JMS

There are a few areas where some tweaks can be done if you are using the JMS API.

- Disable message ID. Use the **setDisableMessageID()** method on the **MessageProducer** class to disable message IDs if you do not need them. This decreases the size of the message and also avoids the overhead of creating a unique ID.
- Disable message timestamp. Use the **setDisableMessageTimeStamp()** method on the **MessageProducer** class to disable message timestamps if you do not need them.
- Avoid **ObjectMessage**. **ObjectMessage** is convenient but it comes at a cost. The body of a **ObjectMessage** uses Java serialization to serialize it to bytes. The Java serialized form of even small objects is very verbose so takes up a lot of space on the wire, also Java serialization is slow compared to custom marshalling techniques. Only use **ObjectMessage** if you really cannot use one of the other message types, that is if you do not know the type of the payload until run-time.
- Avoid **AUTO_ACKNOWLEDGE**. **AUTO_ACKNOWLEDGE** mode requires an acknowledgement to be sent from the server for each message received on the client, this means more traffic on the network. If you can, use **DUPS_OK_ACKNOWLEDGE** or use **CLIENT_ACKNOWLEDGE** or a transacted session and batch up many acknowledgements with one acknowledge/commit.
- Avoid durable messages. By default, JMS messages are durable. If you do not need durable messages then set them to be **non-durable**. Durable messages incur a lot more overhead in persisting them to storage.
- Batch many sends or acknowledgements in a single transaction. HornetQ will only require a network round trip on the commit, not on every send or acknowledgement.

[Report a bug](#)

18.17.3. Other Tunings

There are various places in HornetQ where we can perform some tuning:

- Use Asynchronous Send Acknowledgements. If you need to send durable messages non transactional and you need a guarantee that they have reached the server by the time the call to **send()** returns, do not set durable messages to be sent blocking, instead use asynchronous send acknowledgements to get your acknowledgements of send back in a separate stream.
- Use **pre-acknowledge** mode. With **pre-acknowledge** mode, messages are acknowledged before they are sent to the client. This reduces the amount of acknowledgement traffic on the wire.
- Disable security. There is a small performance boost when you disable security by setting the **security-enabled** parameter to false in **standalone.xml** or **domain.xml**.
- Disable persistence. You can turn off message persistence altogether by setting **persistence-enabled** to false in **standalone.xml** or **domain.xml**.
- Sync transactions lazily. Setting **journal-sync-transactional** to false in **standalone.xml** or **domain.xml** gives better transactional persistent performance at the expense of some possibility of loss of transactions on failure.

- Sync non transactional lazily. Setting ***journal-sync-non-transactional*** to false in **standalone.xml** or **domain.xml** gives better non-transactional persistent performance at the expense of some possibility of loss of durable messages on failure
- Send messages non blocking. Setting ***block-on-durable-send*** and ***block-on-non-durable-send*** to false in **standalone.xml** or **domain.xml** (if you are using JMS and JNDI) or directly on the `ServerLocator`. This means you do not have to wait a whole network round trip for every message sent.
- If you have very fast consumers, you can increase ***consumer-window-size***. This effectively disables consumer flow control.
- Socket NIO vs Socket Old IO. By default HornetQ uses old (blocking) on the server and the client side. NIO is much more scalable but can give some latency hit compared to old blocking IO. To service many thousands of connections on the server, then you must use NIO on the server. However, if there are no thousands of connections on the server you can keep the server acceptors using old IO, and you may get a small performance advantage.
- Use the core API not JMS. Using the JMS API you will have slightly lower performance than using the core API, since all JMS operations need to be translated into core operations before the server can handle them. If using the core API try to use methods that take ***SimpleString*** as much as possible. ***SimpleString***, unlike **`java.lang.String`** does not require copying before it is written to the wire, so if you re-use ***SimpleString*** instances between calls then you can avoid some unnecessary copying.

[Report a bug](#)

18.17.4. Tuning Transport Settings

- TCP buffer sizes. If you have a fast network and fast machines you may get a performance boost by increasing the TCP send and receive buffer sizes.



NOTE

Some operating systems like later versions of Linux include TCP auto-tuning and setting TCP buffer sizes manually can prevent auto-tune from working and actually give you worse performance.

- Increase limit on file handles on the server. If you expect a lot of concurrent connections on your servers, or if clients are rapidly opening and closing connections, you must make sure the user running the server has permission to create sufficient file handles.

This varies from operating system to operating system. On Linux systems you can increase the number of allowable open file handles in the file **`/etc/security/limits.conf`**. For example, add the lines

```
serveruser soft nofile 20000
serveruser hard nofile 20000
```

This would allow up to 20000 file handles to be open by the user ***serveruser***.

- Use ***batch-delay*** and set ***direct-deliver*** to false for the best throughput for very small messages. HornetQ comes with a preconfigured connector/acceptor pair (***netty-throughput***) in **standalone.xml** or **domain.xml** and JMS connection factory

(*ThroughputConnectionFactory*) in `standalone.xml` or `domain.xml` which can be used to give the very best throughput, especially for small messages.

[Report a bug](#)

18.17.5. Tuning the VM

It is highly recommend to use the latest Java JVM for the best performance. Internal testing is done using the Sun JVM, so some of these tunings may not apply to JDKs from other providers like IBM or JRockit

- Garbage collection. For smooth server operation, it is recommended to use a parallel garbage collection algorithm. For example, using the JVM argument **-XX:+UseParallelGC** on Sun JDKs.
- Memory settings. Give as much memory as you can to the server. HornetQ can run in low memory by using paging but if it can run with all queues in RAM this will improve performance. The amount of memory you require will depend on the size and number of your queues and the size and number of your messages. Use the JVM arguments **-Xms** and **-Xmx** to set server available RAM. We recommend setting them to the same high value.
- Aggressive options. Different JVMs provide different sets of JVM tuning parameters. It is recommended at least using **-XX:+AggressiveOpts** and **-XX:+UseFastAccessorMethods**. You may get some mileage with the other tuning parameters depending on your operating system platform and application usage patterns.

[Report a bug](#)

18.17.6. Avoiding Anti-Patterns

- Re-use connections / sessions / consumers / producers. Probably the most common messaging anti-pattern we see is users who create a new connection/session/producer for every message they send or every message they consume. This is a poor use of resources. These objects take time to create and may involve several network round trips. Always re-use them.



NOTE

Some popular libraries such as the Spring JMS Template use these anti-patterns. If you are using Spring JMS Template, you may get poor performance. The Spring JMS Template can only safely be used in an application server which caches JMS sessions, example, using JCA), and only then for sending messages. It cannot be safely be used for synchronously consuming messages, even in an application server.

- Avoid fat messages. Verbose formats such as XML take up a lot of space on the wire and performance will suffer as result. Avoid XML in message bodies if you can.
- Do not create temporary queues for each request. This common anti-pattern involves the temporary queue request-response pattern. With the temporary queue request-response pattern a message is sent to a target and a reply-to header is set with the address of a local temporary queue. When the recipient receives the message they process it then send back a response to the address specified in the reply-to. A common mistake made with this pattern is to create a new temporary queue on each message sent. This drastically reduces performance. Instead the temporary queue should be re-used for many requests.

- Do not use Message-Driven Beans for the sake of it. As soon as you start using MDBs you are greatly increasing the codepath for each message received compared to a straightforward message consumer, since a lot of extra application server code is executed.

[Report a bug](#)

CHAPTER 19. TRANSACTION SUBSYSTEM

19.1. TRANSACTION SUBSYSTEM CONFIGURATION

19.1.1. Transactions Configuration Overview

Introduction

The following procedures show you how to configure the transactions subsystem of JBoss EAP 6.

- [Section 19.1.3, "Configure Your Datasource to Use JTA Transaction API"](#)
- [Section 19.1.4, "Configure an XA Datasource"](#)
- [Section 19.1.2, "Configure the Transaction Manager"](#)
- [Section 19.1.6, "Configure Logging for the Transaction Subsystem"](#)

[Report a bug](#)

19.1.2. Configure the Transaction Manager

You can configure the Transaction Manager (TM) using the web-based Management Console or the command-line Management CLI. For each command or option given, the assumption is made that you are running JBoss EAP 6 as a Managed Domain. If you use a Standalone Server or you want to modify a different profile than **default**, you may need to modify the steps and commands in the following ways.

Notes about the Example Commands

- For the Management Console, the **default** profile is the one which is selected when you first log into the console. If you need to modify the Transaction Manager's configuration in a different profile, select your profile instead of **default**, in each instruction.

Similarly, substitute your profile for the **default** profile in the example CLI commands.

- If you use a Standalone Server, only one profile exists. Ignore any instructions to choose a specific profile. In CLI commands, remove the **/profile=default** portion of the sample commands.



NOTE

In order for the TM options to be visible in the Management Console or Management CLI, the **transactions** subsystem must be enabled. It is enabled by default, and required for many other subsystems to function properly, so it is very unlikely that it would be disabled.

Configure the TM Using the Management Console

To configure the TM using the web-based Management Console, select the **Configuration** tab from the top of the screen. If you use a managed domain, choose the correct profile from the **Profile** selection box at the top left. Expand the **Container** menu and select **Transactions**.

Most options are shown in the Transaction Manager configuration page. The **Recovery** options are hidden by default. Click the **Recovery** tab to see the recovery options. Click **Edit** to edit any of the options. Changes take effect immediately.

Click the **Need Help?** label to display in-line help text.

Configure the TM using the Management CLI

In the Management CLI, you can configure the TM using a series of commands. The commands all begin with **/profile=default/subsystem=transactions/** for a managed domain with profile **default**, or **/subsystem=transactions** for a Standalone Server.



IMPORTANT

If transaction subsystem is configured to use hornetq journal as storage type for transaction logs, then two instances of JBoss EAP is not permitted to use the same directory for storing the journal. Application server instances can't share the same location and each has to configure unique location for it.

Table 19.1. TM Configuration Options

Option	Description	CLI Command
Enable Statistics	Whether to enable transaction statistics. These statistics can be viewed in the Management Console in the Subsystem Metrics section of the Runtime tab.	/profile=default/subsystem=transactions/:write-attribute(name=enable-statistics,value=true)
Enable TSM Status	Whether to enable the transaction status manager (TSM) service, which is used for out-of-process recovery. Running an out of process recovery manager to contact the ActionStatusService from different process is not supported (it is normally contacted in memory).	This configuration option is unsupported.
Default Timeout	The default transaction timeout. This defaults to 300 seconds. You can override this programmatically, on a per-transaction basis.	/profile=default/subsystem=transactions/:write-attribute(name=default-timeout,value=300)
Object Store Path	A relative or absolute filesystem path where the TM object store stores data. By default relative to the object-store-relative-to parameter's value.	/profile=default/subsystem=transactions/:write-attribute(name=object-store-path,value=tx-object-store)

Option	Description	CLI Command
Object Store Path Relative To	References a global path configuration in the domain model. The default value is the data directory for JBoss EAP 6, which is the value of the property jboss.server.data.dir , and defaults to EAP_HOME/domain/data/ for a Managed Domain, or EAP_HOME/standalone/data/ for a Standalone Server instance. The value of the object store object-store-path TM attribute is relative to this path.	/profile=default/subsystem=transactions/:write-attribute(name=object-store-relative-to,value=jboss.server.data.dir)
Socket Binding	Specifies the name of the socket binding used by the Transaction Manager for recovery and generating transaction identifiers, when the socket-based mechanism is used. Refer to process-id-socket-max-ports for more information on unique identifier generation. Socket bindings are specified per server group in the Server tab of the Management Console.	/profile=default/subsystem=transactions/:write-attribute(name=socket-binding,value=txn-recovery-environment)
Status Socket Binding	Specifies the socket binding to use for the Transaction Status manager.	This configuration option is unsupported.
Recovery Listener	Whether or not the Transaction Recovery process should listen on a network socket. Defaults to false .	/profile=default/subsystem=transactions/:write-attribute(name=recovery-listener,value=false)

The following options are for advanced use and can only be modified using the Management CLI. Be cautious when changing them from the default configuration. Contact [Red Hat Global Support Services](#) for more information.

Table 19.2. Advanced TM Configuration Options

Option	Description	CLI Command
jts	Whether to use Java Transaction Service (JTS) transactions. Defaults to false , which uses JTA transactions only.	/profile=default/subsystem=transactions/:write-attribute(name=jts,value=false)

Option	Description	CLI Command
node-identifier	<p>The node identifier for the Transaction Manager. This option is required in the following situations:</p> <ul style="list-style-type: none"> ● For JTS to JTS communications ● When two Transaction Managers access shared resource managers ● When two Transaction Managers access shared object stores <p>The node-identifier must be unique for each Transaction Manager as it is required to enforce data integrity during recovery. The node-identifier must also be unique for JTA because multiple nodes may interact with the same resource manager or share a transaction object store.</p>	<pre>/profile=default/subsystem=transactions/:write-attribute(name=node-identifier,value=1)</pre>
process-id-socket-max-ports	<p>The Transaction Manager creates a unique identifier for each transaction log. Two different mechanisms are provided for generating unique identifiers: a socket-based mechanism and a mechanism based on the process identifier of the process.</p> <p>In the case of the socket-based identifier, a socket is opened and its port number is used for the identifier. If the port is already in use, the next port is probed, until a free one is found. The process-id-socket-max-ports represents the maximum number of sockets the TM will try before failing. The default value is 10.</p>	<pre>/profile=default/subsystem=transactions/:write-attribute(name=process-id-socket-max-ports,value=10)</pre>
process-id-uuid	<p>Set to true to use the process identifier to create a unique identifier for each transaction. Otherwise, the socket-based mechanism is used. Defaults to true. Refer to process-id-socket-max-ports for more information. To enable process-id-socket-binding, set process-id-uuid to false.</p>	<pre>/profile=default/subsystem=transactions/:write-attribute(name=process-id-uuid,value=true)</pre>

Option	Description	CLI Command
process-id-socket-binding	The name of the socket binding configuration to use if the transaction manager should use a socket-based process id. Will be undefined if process-id-uuid is true ; otherwise must be set.	/profile=default/subsystem=transactions/:write-attribute(name=process-id-socket-binding,value=true)
use-hornetq-store	Use HornetQ's journaled storage mechanisms instead of file-based storage, for the transaction logs. This is disabled by default, but can improve I/O performance. It is not recommended for JTS transactions on separate Transaction Managers. When changing this option, the server has to be restarted using the shutdown command for the change to take effect.	/profile=default/subsystem=transactions/:write-attribute(name=use-hornetq-store,value=false)

[Report a bug](#)

19.1.3. Configure Your Datasource to Use JTA Transaction API

Summary

This task shows you how to enable Java Transaction API (JTA) on your datasource.

Prerequisites

You must meet the following conditions before continuing with this task:

- Your database or other resource must support Java Transaction API. If in doubt, consult the documentation for your database or other resource.
- Create a datasource. Refer to [Section 6.3.1, "Create a Non-XA Datasource with the Management Interfaces"](#).
- Stop JBoss EAP 6.
- Have access to edit the configuration files directly, in a text editor.

Procedure 19.1. Configure the Datasource to use Java Transaction API

1. **Open the configuration file in a text editor.**

Depending on whether you run JBoss EAP 6 in a managed domain or standalone server, your configuration file will be in a different location.

- **Managed domain**

The default configuration file for a managed domain is in

EAP_HOME/domain/configuration/domain.xml for Red Hat Enterprise Linux, and ***EAP_HOME/domain/configuration/domain.xml*** for Microsoft Windows Server.

- **Standalone server**

The default configuration file for a standalone server is in **`EAP_HOME/standalone/configuration/standalone.xml`** for Red Hat Enterprise Linux, and **`EAP_HOME\standalone\configuration\standalone.xml`** for Microsoft Windows Server.

2. **Locate the `<datasource>` tag that corresponds to your datasource.**

The datasource will have the **`jndi-name`** attribute set to the one you specified when you created it. For example, the ExampleDS datasource looks like this:

```
<datasource jndi-name="java:jboss/datasources/ExampleDS" pool-name="H2DS"
enabled="true" jta="true" use-java-context="true" use-ccm="true">
```

3. **Set the `jta` attribute to `true`.**

Add the following to the contents of your **`<datasource>`** tag, as they appear in the previous step: **`jta="true"`**

Unless you have a specific use case (such as defining a read only datasource) Red Hat discourages overriding the default value of **`jta=true`**. This setting indicates that the datasource will honor the Java Transaction API and allows better tracking of connections by the JCA implementation.

4. **Save the configuration file.**

Save the configuration file and exit the text editor.

5. **Start JBoss EAP 6.**

Relaunch the JBoss EAP 6 server.

Result:

JBoss EAP 6 starts, and your datasource is configured to use Java Transaction API.

[Report a bug](#)

19.1.4. Configure an XA Datasource

Prerequisites

Log into the Management Console.

1. **Add a new datasource.**

Add a new datasource to JBoss EAP 6. Click the **XA Datasource** tab at the top.



NOTE

Refer to *Create an XA Datasource with the Management Interfaces* section of the *Administration and Configuration Guide* on the Red Hat Customer Portal for information on how to add a new datasource to JBoss EAP 6.

2. **Configure additional properties as appropriate.**

All datasource parameters are listed in [Section 6.7.1, "Datasource Parameters"](#).

Result

Your XA Datasource is configured and ready to use.

[Report a bug](#)

19.1.5. About Transaction Log Messages

To track transaction status while keeping the log files readable, use the **DEBUG** log level for the transaction logger. For detailed debugging, use the **TRACE** log level. Refer to [Section 19.1.6, "Configure Logging for the Transaction Subsystem"](#) for information on configuring the transaction logger.

The transaction manager can generate a lot of logging information when configured to log in the **TRACE** log level. Following are some of the most commonly-seen messages. This list is not comprehensive, so you may see other messages than these.

Table 19.3. Transaction State Change

Transaction Begin	<p>When a transaction begins, the following code is executed:</p> <pre>com.arjuna.ats.arjuna.coordinator.BasicAction: :Begin:1342 tsLogger.logger.trace("BasicAction::Begin() for action-id "+ get_uid());</pre>
Transaction Commit	<p>When a transaction commits, the following code is executed:</p> <pre>com.arjuna.ats.arjuna.coordinator.BasicAction: :End:1342 tsLogger.logger.trace("BasicAction::End() for action-id "+ get_uid());</pre>
Transaction Rollback	<p>When a transaction rolls back, the following code is executed:</p> <pre>com.arjuna.ats.arjuna.coordinator.BasicAction: :Abort:1575 tsLogger.logger.trace("BasicAction::Abort() for action-id "+ get_uid());</pre>
Transaction Timeout	<p>When a transaction times out, the following code is executed:</p> <pre>com.arjuna.ats.arjuna.coordinator.Transaction Reaper::doCancellations:349 tsLogger.logger.trace("Reaper Worker " + Thread.currentThread() + " attempting to cancel " + e._control.get_uid());</pre> <p>You will then see the same thread rolling back the transaction as shown above.</p>

[Report a bug](#)

19.1.6. Configure Logging for the Transaction Subsystem

Summary

Use this procedure to control the amount of information logged about transactions, independent of other logging settings in JBoss EAP 6. The main procedure shows how to do this in the web-based Management Console. The Management CLI command is given afterward.

Procedure 19.2. Configure the Transaction Logger Using the Management Console

1. **Navigate to the Logging configuration area.**

In the Management Console, click the **Configuration** tab. If you use a managed domain, choose the server profile you wish to configure, from the **Profile** selection box at the top left.

Expand the **Core** menu, and select **Logging**.

2. **Edit the `com.arjuna` attributes.**

Select the **Log Categories** tab. Select **com.arjuna** and click **Edit** in the **Details** section. This is where you can add class-specific logging information. The **com.arjuna** class is already present. You can change the log level and whether to use parent handlers.

Log Level

The log level is **WARN** by default. Because transactions can produce a large quantity of logging output, the meaning of the standard logging levels is slightly different for the transaction logger. In general, messages tagged with levels at a lower severity than the chosen level are discarded.

Transaction Logging Levels, from Most to Least Verbose

- TRACE
- DEBUG
- INFO
- WARN
- ERROR
- FAILURE

Use Parent Handlers

Whether the logger should send its output to its parent logger. The default behavior is **true**.

3. Changes take effect immediately.

[Report a bug](#)

19.2. TRANSACTION ADMINISTRATION

19.2.1. Browse and Manage Transactions

The Management CLI supports the ability to browse and manipulate transaction records. This functionality is provided by the interaction between the Transaction Manager and the management API of JBoss EAP 6.

The Transaction Manager stores information about each pending transaction and the participants involved the transaction, in a persistent storage called the *object store*. The management API exposes the object store as a resource called the **log-store**. An API operation called **probe** reads the transaction logs and creates a node for each log. You can call the **probe** command manually, whenever you need to refresh the **log-store**. It is normal for transaction logs to appear and disappear quickly.

Example 19.1. Refresh the Log Store

This command refreshes the log store for server groups which use the profile **default** in a managed domain. For a standalone server, remove the **profile=default** from the command.

```
/profile=default/subsystem=transactions/log-store=log-store/:probe
```

Example 19.2. View All Prepared Transactions

To view all prepared transactions, first refresh the log store (see [Example 19.1, "Refresh the Log Store"](#)), then run the following command, which functions similarly to a filesystem **ls** command.

```
ls /profile=default/subsystem=transactions/log-store=log-store/transactions
```

Each transaction is shown, along with its unique identifier. Individual operations can be run against an individual transaction (see [Manage a Transaction](#)).

Manage a Transaction

View a transaction's attributes.

To view information about a transaction, such as its JNDI name, EIS product name and version, or its status, use the **:read-resource** CLI command.

```
/profile=default/subsystem=transactions/log-store=log-store/transactions=0\:\:ffff7f000001\:-  
b66efc2\:\:4f9e6f8f\:\:9:read-resource
```

View the participants of a transaction.

Each transaction log contains a child element called **participants**. Use the **read-resource** CLI command on this element to see the participants of the transaction. Participants are identified by their JNDI names.

```
/profile=default/subsystem=transactions/log-store=log-store/transactions=0\:\:ffff7f000001\:-  
b66efc2\:\:4f9e6f8f\:\:9/participants=java\:\:JmsXA:read-resource
```

The result may look similar to this:

```
{  
  "outcome" => "success",  
  "result" => {  
    "eis-product-name" => "HornetQ",
```

```

    "eis-product-version" => "2.0",
    "jndi-name" => "java:/JmsXA",
    "status" => "HEURISTIC",
    "type" => "/StateManager/AbstractRecord/XAResourceRecord"
  }
}

```

The outcome status shown here is in a **HEURISTIC** state and is eligible for recovery. See [Recover a transaction](#) for more details.

In special cases it is possible to create orphan records in the object store, that is XAResourceRecords, which do not have any corresponding transaction record in the log. For example, XA resource prepared but crashed before the TM recorded and is inaccessible for the domain management API. To access such records you need to set management option **expose-all-logs** to **true**. This option is not saved in management model and is restored to **false** when the server is restarted.

```

/profile=default/subsystem=transactions/log-store=log-store:write-attribute(name=expose-all-logs,
value=true)

```

Delete a transaction.

Each transaction log supports a **:delete** operation, to delete the transaction log representing the transaction.

```

/profile=default/subsystem=transactions/log-store=log-store/transactions=0:fff7f000001\:-
b66efc2\4f9e6f8f\9:delete

```

Recover a transaction.

Each transaction participant supports recovery via the **:recover** CLI command.

```

/profile=default/subsystem=transactions/log-store=log-store/transactions=0:fff7f000001\:-
b66efc2\4f9e6f8f\9/participants=2:recover

```

Recovery of heuristic transactions and participants

- If the transaction's status is **HEURISTIC**, the recovery operation changes the state to **PREPARE** and triggers a recovery.
- If one of the transaction's participants is heuristic, the recovery operation tries to replay the **commit** operation. If successful, the participant is removed from the transaction log. You can verify this by re-running the **:probe** operation on the **log-store** and checking that the participant is no longer listed. If this is the last participant, the transaction is also deleted.

Refresh the status of a transaction which needs recovery.

If a transaction needs recovery, you can use the **:refresh** CLI command to be sure it still requires recovery, before attempting the recovery.

```

/profile=default/subsystem=transactions/log-store=log-store/transactions=0:fff7f000001\:-
b66efc2\4f9e6f8f\9/participants=2:refresh

```

View Transaction Statistics

If Transaction Manager statistics are enabled, you can view statistics about the Transaction Manager and transaction subsystem. See [Section 19.1.2, “Configure the Transaction Manager”](#) for information about how to enable Transaction Manager statistics.

You can view statistics either via the management console or the Management CLI. In the management console, transaction statistics are available via **Runtime** → **Status** → **Subsystems** → **Transactions**. Transaction statistics are available for each server in a managed domain. To view the status of a different server, select **Change Server** in the left-hand menu and select the server from the list.

The following table shows each available statistic, its description, and the Management CLI command to view the statistic.

Table 19.4. Transaction Subsystem Statistics

Statistic	Description	CLI Command
Total	The total number of transactions processed by the Transaction Manager on this server.	<pre>/host=<i>master</i>/server=<i>server</i> - <i>one</i>/subsystem=transactions/ :read- attribute(name=number-of- transactions,include- defaults=true)</pre>
Committed	The number of committed transactions processed by the Transaction Manager on this server.	<pre>/host=<i>master</i>/server=<i>server</i> - <i>one</i>/subsystem=transactions/ :read- attribute(name=number-of- committed- transactions,include- defaults=true)</pre>
Aborted	The number of aborted transactions processed by the Transaction Manager on this server.	<pre>/host=<i>master</i>/server=<i>server</i> - <i>one</i>/subsystem=transactions/ :read- attribute(name=number-of- aborted- transactions,include- defaults=true)</pre>

Statistic	Description	CLI Command
Timed Out	The number of timed out transactions processed by the Transaction Manager on this server.	<pre> /host=<i>master</i>/server=<i>server</i> - <i>one</i>/subsystem=transactions/ :read- attribute(name=number-of- timed-out- transactions,include- defaults=true) </pre>
Heuristics	Not available in the Management Console. Number of transactions in a heuristic state.	<pre> /host=<i>master</i>/server=<i>server</i> - <i>one</i>/subsystem=transactions/ :read- attribute(name=number-of- heuristics,include- defaults=true) </pre>
In-Flight Transactions	Not available in the Management Console. Number of transactions which have begun but not yet terminated.	<pre> /host=<i>master</i>/server=<i>server</i> - <i>one</i>/subsystem=transactions/ :read- attribute(name=number-of- inflight-transactions,include- defaults=true) </pre>
Failure Origin - Applications	The number of failed transactions whose failure origin was an application.	<pre> /host=<i>master</i>/server=<i>server</i> - <i>one</i>/subsystem=transactions/ :read- attribute(name=number-of- application- rollbacks,include- defaults=true) </pre>

Statistic	Description	CLI Command
Failure Origin - Resources	The number of failed transactions whose failure origin was a resource.	<pre>/host=master/server=server - one/subsystem=transactions/ :read- attribute(name=number-of- resource-rollback,include- defaults=true)</pre>
Participant ID	The ID of the participant.	<pre>/host=master/server=server - one/subsystem=transactions/ log-store=log- store/transactions=0\:\:ffff7f00 0001\:- b66efc2\:\:4f9e6f8f\:\:9:read- children-names(child- type=participants)</pre>
List of all transactions	The complete list of transactions.	<pre>/host=master/server=server - one/subsystem=transactions/ log-store=log-store:read- children-names(child- type=transactions)</pre>

[Report a bug](#)

19.3. TRANSACTION REFERENCES

19.3.1. JBoss Transactions Errors and Exceptions

For details about exceptions thrown by methods of the **UserTransaction** class, see the *UserTransaction API* specification at <http://docs.oracle.com/javaee/6/api/javax/transaction/UserTransaction.html>.

[Report a bug](#)

19.3.2. Limitations on JTA Transactions

JTA transactions are not fully transaction distribution aware across multiple instances of JBoss EAP 6. In the current implementation, transaction context is passed along the remote EJB calls. However, this works only for simple scenarios where one server calls another server. If there are multiple servers connecting to the transaction distribution, the context propagation is not completely safe.

For full transaction distribution support behavior, you must use JTS transactions. You must configure the ORB in order to use JTS transactions, which includes:

- enabling transactions in the JacORB subsystem;
- configuring the Transaction subsystem to use JTS transactions;
- calling EJB by using IIOP protocol.
- [Section 19.4.3, "Configure the ORB for JTS Transactions"](#)

[Report a bug](#)

19.4. ORB CONFIGURATION

19.4.1. About Common Object Request Broker Architecture (CORBA)

Common Object Request Broker Architecture (CORBA) is a standard that enables applications and services to work together even when they are written in multiple, otherwise-incompatible, languages or hosted on separate platforms. CORBA requests are brokered by a server-side component called an *Object Request Broker (ORB)*. JBoss EAP 6 provides an ORB instance, by means of the JacORB component.

The ORB is used internally for *Java Transaction Service (JTS)* transactions, and is also available for use by your own applications.

[Report a bug](#)

19.4.2. JacORB Configuration



NOTE

In a managed domain, the JacORB subsystem is available in **full** and **full-ha** profiles only. In a standalone server, it is available when you use the **standalone-full.xml** or **standalone-full-ha.xml** configurations.

JacORB properties are most easily configured using the Management CLI. Some attributes are set in the subsystem directly, and others must be set at the system level.

To view the settings that can be configured directly in the subsystem, use the following Management CLI command:

```
/subsystem=jacorb:read-resource(include-runtime=true, recursive=true)
```

The current settings will be listed:

```
"add-component-via-interceptor" => "on",  
"cache-poa-names" => "off",  
"cache-typecodes" => "off",  
"chunk-custom-rmi-valuetypes" => "on",  
"client-requires" => "None",  
"client-supports" => "MutualAuth",  
"client-timeout" => 0,
```

```

"comet" => "off",
"export-corballoc" => "on",
"giop-minor-version" => 2,
"indirection-encoding-disable" => "off",
"iona" => "off",
"lax-boolean-encoding" => "off",
"max-managed-buf-size" => 24,
"max-server-connections" => 2147483647,
"max-threads" => 32,
"monitoring" => "off",
"name" => "JBoss",
"outbuf-cache-timeout" => -1,
"outbuf-size" => 2048,
"pool-size" => 5,
"print-version" => "off",
"properties" => undefined,
"queue-max" => 100,
"queue-min" => 10,
"queue-wait" => "off",
"retries" => 5,
"retry-interval" => 500,
"root-context" => "JBoss/Naming/root",
"security" => "identity",
"security-domain" => undefined,
"server-requires" => "None",
"server-supports" => "MutualAuth",
"server-timeout" => 0,
"socket-binding" => "jacorb",
"ssl-socket-binding" => "jacorb-ssl",
"strict-check-on-tc-creation" => "off",
"sun" => "on",
"support-ssl" => "off",
"transactions" => "spec",
"use-bom" => "off",
"use-imr" => "off",
"ior-settings" => undefined

```

Other settings must be configured at the system level. Be aware that system-level settings are not transparent in the JBoss EAP model and, as such, are not visible when interrogating the JacORB subsystem specifically. They will not appear, for example, when using the **jacorb:read-resource** command shown above.

Use the following command examples to set JacORB attributes using system-level properties:

```
/system-property=jacorb.connection.client.pending_reply_timeout:add(value=600000)
```

```
/system-property=jacorb.connection.client.idle_timeout:add(value=120000)
```

```
/system-property=jacorb.connection.server.timeout:add(value=300000)
```

```
/system-property=jacorb.native_char_codeset:add(value=UTF8)
```

```
/system-property=jacorb.native_wchar_codeset:add(value=UTF16)
```

[Report a bug](#)

19.4.3. Configure the ORB for JTS Transactions

In a default installation of JBoss EAP 6, the ORB is disabled. You can enable the ORB using the command-line Management CLI.

Procedure 19.3. Configure the ORB using the Management Console

1. **View the profile settings.**

Select **Configuration** from the top of the management console. If you use a managed domain, select either the **full** or **full-ha** profile from the selection box at the top left.

2. **Modify the Initializers Settings**

Expand the **Subsystems** menu. Expand the **Container** menu and select **JacORB**.

In the form that appears in the main screen, select the **Initializers** tab and click the **Edit** button.

Enable the security interceptors by setting the value of **Security** to **on**.

To enable the ORB for JTS, set the **Transaction Interceptors** value to **on**, rather than the default **spec**.

Refer to the **Need Help?** link in the form for detailed explanations about these values. Click **Save** when you have finished editing the values.

3. **Advanced ORB Configuration**

Refer to the other sections of the form for advanced configuration options. Each section includes a **Need Help?** link with detailed information about the parameters.

Configure the ORB using the Management CLI

You can configure each aspect of the ORB using the Management CLI. The following commands configure the initializers to the same values as the procedure above, for the Management Console. This is the minimum configuration for the ORB to be used with JTS.

These commands are configured for a managed domain using the **full** profile. If necessary, change the profile to suit the one you need to configure. If you use a standalone server, omit the **/profile=full** portion of the commands.

Example 19.3. Enable the Security Interceptors

```
/profile=full/subsystem=jacorb/:write-attribute(name=security,value=on)
```

Example 19.4. Enable Transactions in the JacORB Subsystem

```
/profile=full/subsystem=jacorb/:write-attribute(name=transactions,value=on)
```

Example 19.5. Enable JTS in the Transaction Subsystem

```
/profile=full/subsystem=transactions:write-attribute(name=jts,value=true)
```

**NOTE**

For JTS activation, the server must be restarted as reload is not enough.

[Report a bug](#)

19.5. JDBC OBJECT STORE SUPPORT

19.5.1. JDBC Store for Transactions

Prerequisites:

- [Section 3.4.4, "Connect to a Managed Server Instance Using the Management CLI"](#)

Transactions can use a JDBC datasource as its object store. If the database to be used is configured for failover and recovery, this may be a better option than using disk space on an application server. The advantages must be weighed up against the fact that a raw JDBC object store is a special object store and may not perform as well as a file system or HornetQ journal object store.

**NOTE**

A JDBC datasource used as a Transactions object store *must* specify **jta="false"** in the **datasource** section of the server's configuration file.

Procedure 19.4. Enable Use of a JDBC Datasource as a Transactions Object Store

1. Set **use-jdbc-store** to **true**.

```
/subsystem=transactions:write-attribute(name=use-jdbc-store, value=true)
```

2. Set **jdbc-store-datasource** to the JNDI name for the data source to use.

```
/subsystem=transactions:write-attribute(name=jdbc-store-datasource,
value=java:jboss/datasources/TransDS)
```

3. Restart the JBoss EAP server for the changes to take effect.

```
shutdown --restart=true
```

The complete set of attributes is provided below.

Table 19.5. Transactions JDBC Store Properties

Property	Description
use-jdbc-store	Set this to "true" to enable the JDBC store for transactions.

Property	Description
jdbc-store-datasource	The JNDI name of the JDBC datasource used for storage.
jdbc-action-store-drop-table	Drop and recreate the action store tables at launch. Optional, defaults to "false".
jdbc-action-store-table-prefix	The prefix for the action store table names. Optional.
jdbc-communication-store-drop-table	Drop and recreate the communication store tables at launch. Optional, defaults to "false".
jdbc-communication-store-table-prefix	The prefix for the communication store table names. Optional.
jdbc-state-store-drop-table	Drop and recreate the state store tables at launch. Optional, defaults to "false".
jdbc-state-store-table-prefix	The prefix for the state store table names. Optional.

See Also:

- [Section 19.1.3, "Configure Your Datasource to Use JTA Transaction API"](#)

[Report a bug](#)

CHAPTER 20. MAIL SUBSYSTEM

20.1. USE CUSTOM TRANSPORTS IN MAIL SUBSYSTEM

When using a standard mail server (POP3, IMAP) the server has a set of attributes that can be defined, some of which are required.

The most important of these is the **outbound-socket-binding-ref** which is a reference to the outbound mail socket binding and is defined with the host address and port number.

This is not the most effective solution for some users as their host configuration used multiple hosts for load balancing purposes. This configuration, however, is not supported by standard JavaMail requiring some users to implement custom mail transports.

These custom transports do not require the **outbound-socket-binding-ref** and allow custom host property formats.

A custom transport can be configured through the CLI using the following commands:

Procedure 20.1.

1. Add new mail session. The command below creates new session called `mySession` and sets JNDI to **java:jboss/mail/MySession**:

```
/subsystem=mail/mail-session=mySession:add(jndi-name=java:jboss/mail/MySession)
```

2. Add an outbound socket binding. The command below adds a socket binding named **my-smtp-binding** which points to **localhost:25**.

```
/socket-binding-group=standard-sockets/remote-destination-outbound-socket-binding=my-smtp-binding:add(host=localhost, port=25)
```

3. Add an SMTP server with **outbound-socket-binding-ref**. The command below adds an SMTP called **my-smtp-binding** and defines a username, password and TLS configuration.

```
/subsystem=mail/mail-session=mySession/server=smtp:add(outbound-socket-binding-ref=my-smtp-binding, username=user, password=pass, tls=true)
```

4. Repeat this process for POP3 and IMAP:

```
/socket-binding-group=standard-sockets/remote-destination-outbound-socket-binding=my-pop3-binding:add(host=localhost, port=110)
```

```
/subsystem=mail/mail-session=mySession/server=pop3:add(outbound-socket-binding-ref=my-pop3-binding, username=user, password=pass)
```

```
/socket-binding-group=standard-sockets/remote-destination-outbound-socket-binding=my-imap-binding:add(host=localhost, port=143)
```

```
/subsystem=mail/mail-session=mySession/server=imap:add(outbound-socket-binding-ref=my-imap-binding, username=user, password=pass)
```

- To use a custom server, create a new custom mail server without an outbound socket binding (as it is optional) and instead provide the host information as part of properties.

```
/subsystem=mail/mail-
session=mySession/custom=myCustomServer:add(username=user,password=pass,
properties={"host" => "myhost", "my-property" =>"value"})
```

When defining custom protocols, any property name that contains a dot (.) is considered to be a fully-qualified name and passed as it is supplied. Any other format (*my-property*, for example) will be translated into the following format: **mail.server-name.my-property**.

Below is an example complete configuration XML configuration that highlights a custom format in the custom-server attribute:

```
<subsystem xmlns="urn:jboss:domain:mail:1.1">
  <mail-session jndi-name="java:/Mail" from="user.name@domain.org">
    <smtp-server outbound-socket-binding-ref="mail-smtp" tls="true">
      <login name="user" password="password"/>
    </smtp-server>
    <pop3-server outbound-socket-binding-ref="mail-pop3"/>
    <imap-server outbound-socket-binding-ref="mail-imap">
      <login name="nobody" password="password"/>
    </imap-server>
  </mail-session>
  <mail-session debug="true" jndi-name="java:jboss/mail/Default">
    <smtp-server outbound-socket-binding-ref="mail-smtp"/>
  </mail-session>
  <mail-session debug="true" jndi-name="java:jboss/mail/Custom">
    <custom-server name="smtp">
      <login name="username" password="password"/>
      <property name="host" value="mail.example.com"/>
    </custom-server>
    <custom-server name="pop3" outbound-socket-binding-ref="mail-pop3">
      <property name="custom_prop" value="some-custom-prop-value"/>
      <property name="some.fully.qualified.property" value="fully-qualified-prop-name"/>
    </custom-server>
  </mail-session>
  <mail-session debug="true" jndi-name="java:jboss/mail/Custom2">
    <custom-server name="pop3" outbound-socket-binding-ref="mail-pop3">
      <property name="custom_prop" value="some-custom-prop-value"/>
    </custom-server>
  </mail-session>
</subsystem>
```

[Report a bug](#)

CHAPTER 21. ENTERPRISE JAVABEANS

21.1. INTRODUCTION

21.1.1. Overview of Enterprise JavaBeans

Enterprise JavaBeans (EJB) 3.1 is an API for developing distributed, transactional, secure and portable Java EE applications through the use of server-side components called Enterprise Beans. Enterprise Beans implement the business logic of an application in a decoupled manner that encourages reuse. Enterprise JavaBeans 3.1 is documented as the Java EE specification JSR-318.

JBoss EAP 6 has full support for applications built using the Enterprise JavaBeans 3.1 specification.

[Report a bug](#)

21.1.2. Overview of Enterprise JavaBeans for Administrators

JBoss administrators have many configuration options available to them to control the performance of Enterprise Beans in JBoss EAP 6. These options can be accessed using the Management Console or the command line configuration tool. Editing the XML server configuration file to apply changes is also possible but not recommended.

The EJB configuration options are located in slightly different places in the Management Console depending on how the server is being run.

1. Click on the **Configuration** tab at the top of the Management Console.
2. If you are running in Domain mode, select a profile from the **Profiles** drop down menu on the top left.
3. Expand the **Subsystems** menu.
4. Expand the **Container** menu, then select **EJB 3**.

[Report a bug](#)

21.1.3. Enterprise Beans

Enterprise beans are server-side application components as defined in the Enterprise JavaBeans (EJB) 3.1 specification, JSR-318. Enterprise beans are designed for the implementation of application business logic in a decoupled manner to encourage reuse.

Enterprise beans are written as Java classes and annotated with the appropriate EJB annotations. They can be deployed to the application server in their own archive (a JAR file) or be deployed as part of a Java EE application. The application server manages the lifecycle of each enterprise bean and provides services to them such as security, transactions, and concurrency management.

An enterprise bean can also define any number of business interfaces. Business interfaces provide greater control over which of the bean's methods are available to clients and can also allow access to clients running in remote JVMs.

There are three types of Enterprise Bean: Session beans, Message-driven beans and Entity beans.



IMPORTANT

Entity beans are now deprecated in EJB 3.1 and Red Hat recommends the use of JPA entities instead. Red Hat only recommends the use of Entity beans for backwards compatibility with legacy systems.

[Report a bug](#)

21.1.4. Session Beans

Session Beans are Enterprise Beans that encapsulate a set of related business processes or tasks and are injected into the classes that request them. There are three types of session bean: stateless, stateful, and singleton.

[Report a bug](#)

21.1.5. Message-Driven Beans

Message-driven Beans (MDBs) provide an event driven model for application development. The methods of MDBs are not injected into or invoked from client code but are triggered by the receipt of messages from a messaging service such as a Java Messaging Service (JMS) server. The Java EE 6 specification requires that JMS is supported but other messaging systems can be supported as well.

[Report a bug](#)

21.2. CONFIGURING BEAN POOLS

21.2.1. Bean Pools

JBoss EAP 6 maintains a number of instances of deployed stateless enterprise beans in memory to provide faster performance. This technique is called bean pooling. When a bean is required the application server can take one from the appropriate pool of already available beans instead of instantiating a new one. When the bean is no longer required it is returned to the pool for reuse.

Bean pools are configured and maintained separately for stateless session beans and for message-driven beans.

@org.jboss.ejb3.annotation.Pool annotation can be used on EJBs to identify the pool, which has to be used for that EJB. This annotation points to the name of that pool.

[Report a bug](#)

21.2.2. Create a Bean Pool

Bean pools can be created using the Management Console and the CLI tool.

Bean pools can also be created by adding the required bean pool configuration to the server configuration file using a text editor. [Example 21.2, "XML Configuration Sample"](#) is an example of what this configuration looks like.

Procedure 21.1. Create a bean pool using the Management Console

1. Login to the Management Console. Refer to [Section 3.3.2, "Log in to the Management Console"](#).

2. Click on the **Configuration** tab at the top of the screen. Expand the **Container** menu and select **EJB 3**. Select the **Bean Pools** tab.
3. Click **Add**. The **Add EJB3 Bean Pools** dialog appears.
4. Specify the required details, **Name**, **Max Pool Size**, **Timeout** value, and **Timeout** unit.
5. Click **Save** button to finish.

Procedure 21.2. Create a bean pool using the CLI

1. Launch the CLI tool and connect to your server. Refer to [Section 3.4.4, "Connect to a Managed Server Instance Using the Management CLI"](#).
2. Use the **add** operation with the following syntax.

```
/subsystem=ejb3/strict-max-bean-instance-pool=BEANPOOLNAME:add(max-pool-size=MAXSIZE, timeout=TIMEOUT, timeout-unit="UNIT")
```

- Replace *BEANPOOLNAME* with the required name for the bean pool.
 - Replace *MAXSIZE* with the maximum size of the bean pool.
 - Replace *TIMEOUT*
 - Replace *UNIT* with the required time unit. Allowed values are: **NANOSECONDS**, **MICROSECONDS**, **MILLISECONDS**, **SECONDS**, **MINUTES**, **HOURS**, and **DAYS**.
3. Use the **read-resource** operation to confirm the creation of the bean pool.

```
/subsystem=ejb3/strict-max-bean-instance-pool=BEANPOOLNAME:read-resource
```

Example 21.1. Create a Bean Pool using the CLI

```
[standalone@localhost:9999 /] /subsystem=ejb3/strict-max-bean-instance-pool=ACCTS_BEAN_POOL:add(max-pool-size=500, timeout=5000, timeout-unit="SECONDS") {"outcome" => "success"}
```

Example 21.2. XML Configuration Sample

```
<subsystem xmlns="urn:jboss:domain:ejb3:1.2">
  <pools>
    <bean-instance-pools>
      <strict-max-pool name="slsb-strict-max-pool" max-pool-size="20"
        instance-acquisition-timeout="5"
        instance-acquisition-timeout-unit="MINUTES" />
      <strict-max-pool name="mdb-strict-max-pool" max-pool-size="20"
        instance-acquisition-timeout="5"
        instance-acquisition-timeout-unit="MINUTES" />
    
```

```

    </bean-instance-pools>
  </pools>
</subsystem>

```

[Report a bug](#)

21.2.3. Remove a Bean Pool

Unused bean pools can be removed using the Management Console.

Prerequisites:

- The bean pool that you want to remove cannot be in use. Refer to [Section 21.2.5, “Assign Bean Pools for Session and Message-Driven Beans”](#) to ensure that it is not being used.

Procedure 21.3. Remove a bean pool using the Management Console

1. Login to the Management Console. Refer to [Section 3.3.2, “Log in to the Management Console”](#).
2. Click on the **Configuration** tab at the top of the screen. Expand the **Container** menu and select **EJB 3**. Select the **Bean Pools** tab.
3. Select the bean pool to remove in the list.
4. Click **Remove**. The **Remove Item** dialog appears.
5. Click **Confirm** to confirm.

Procedure 21.4. Remove a bean pool using the CLI

1. Launch the CLI tool and connect to your server. Refer to [Section 3.4.4, “Connect to a Managed Server Instance Using the Management CLI”](#).
2. Use the **remove** operation with the following syntax.

```
/subsystem=ejb3/strict-max-bean-instance-pool=BEANPOOLNAME:remove
```

- Replace *BEANPOOLNAME* with the required name for the bean pool.

Example 21.3. Removing a Bean Pool using the CLI

```
[standalone@localhost:9999 /] /subsystem=ejb3/strict-max-bean-instance-
pool=ACCTS_BEAN_POOL:remove
{"outcome" => "success"}
```

[Report a bug](#)

21.2.4. Edit a Bean Pool

Bean pools can be edited using the Management Console.

Procedure 21.5. Edit a bean pool using the Management Console

1. Login to the Management Console. [Section 3.3.2, “Log in to the Management Console”](#)
2. Click on the **Configuration** tab at the top of the screen. Expand the **Container** menu and select **EJB 3**. Select the **Bean Pools** tab.
3. Select the bean pool you want to edit.
4. Click **Edit**.
5. Edit the details you want to change. Only **Max Pool Size**, **Timeout** value, and **Timeout Unit** can be changed.
6. Click **Save** to finish.

Procedure 21.6. Edit a bean pool using the CLI

1. Launch the CLI tool and connect to your server. Refer to [Section 3.4.4, “Connect to a Managed Server Instance Using the Management CLI”](#).
2. Use the **write-attribute** operation with the following syntax for each attribute of the bean pool to be changed.

```
/subsystem=ejb3/strict-max-bean-instance-pool=BEANPOOLNAME:write-attribute(name="ATTRIBUTE", value="VALUE")
```

- Replace *BEANPOOLNAME* with the required name for the bean pool.
 - Replace *ATTRIBUTE* with the name of the attribute to be edited. The attributes that can be edited in this way are **max-pool-size**, **timeout**, and **timeout-unit**.
 - Replace *VALUE* with the required value of the attribute.
3. Use the **read-resource** operation to confirm the changes to the bean pool.

```
/subsystem=ejb3/strict-max-bean-instance-pool=BEANPOOLNAME:read-resource
```

Example 21.4. Set the Timeout Value of a Bean Pool using the CLI

```
[standalone@localhost:9999 /] /subsystem=ejb3/strict-max-bean-instance-pool=HSBeanPool:write-attribute(name="timeout", value="1500")
{"outcome" => "success"}
```

[Report a bug](#)

21.2.5. Assign Bean Pools for Session and Message-Driven Beans

JBoss Administrators can assign individual bean pools for use by session beans and message-driven beans. Bean pools can be assigned by using the Management Console or the Management CLI.

By default, two bean pools are provided, **slsb-strict-max-pool** and **mdb-strict-max-pool** for stateless session beans and message-driven beans respectively.

To create or edit bean pools, refer to [Section 21.2.2, "Create a Bean Pool"](#) and [Section 21.2.4, "Edit a Bean Pool"](#).

Additionally, the **@Pool** annotation can be used on EJBs to identify the pool to be used for that EJB.



NOTE

Using the **@Pool** annotation on a particular EJB will override any default settings specified using the management interfaces.

Procedure 21.7. Assign Bean Pools for Session and Message-Driven Beans using the Management Console

1. Login to the Management Console. [Section 3.3.2, "Log in to the Management Console"](#)
2. Click on the **Configuration** tab at the top of the screen. Expand the **Container** menu and select **EJB 3**. Select the **Container** tab.
3. Click **Edit**.
4. Select the bean pool to use for each type of bean from the appropriate combo-box.
5. Click **Save** to finish.

Procedure 21.8. Assign Bean Pools for Session and Message-Driven Beans using the CLI

1. Launch the CLI tool and connect to your server. Refer to [Section 3.4.4, "Connect to a Managed Server Instance Using the Management CLI"](#).
2. Use the **write-attribute** operation with the following syntax.

```
/subsystem=ejb3:write-attribute(name="BEANTYPE", value="BEANPOOL")
```

- Replace *BEANTYPE* with **default-mdb-instance-pool** for Message-Driven Beans or **default-slsb-instance-pool** for stateless session beans.
- Replace *BEANPOOL* with the name of the bean pool to assign.

3. Use the **read-resource** operation to confirm the changes.

```
/subsystem=ejb3:read-resource
```

Example 21.5. Assign a Bean Pool for Session Beans using the CLI

```
[standalone@localhost:9999 /] /subsystem=ejb3:write-attribute(name="default-slsb-instance-pool", value="LV_SLSB_POOL")
{"outcome" => "success"}
```

Example 21.6. XML Configuration Sample

```

<subsystem xmlns="urn:jboss:domain:ejb3:1.2">
  <session-bean>
    <stateless>
      <bean-instance-pool-ref pool-name="slsb-strict-max-pool"/>
    </stateless>
    <stateful default-access-timeout="5000" cache-ref="simple"/>
    <singleton default-access-timeout="5000"/>
  </session-bean>
  <mdb>
    <resource-adapter-ref resource-adapter-name="hornetq-ra"/>
    <bean-instance-pool-ref pool-name="mdb-strict-max-pool"/>
  </mdb>
</subsystem>

```

Procedure 21.9. Assign a Bean Pool for a Session or Message-Driven Bean using the `@Pool` annotation

1. Add the `@Pool` annotation to the bean and specify the name of the bean pool to be used.

```

@Stateless
@Pool("slsb-strict-max-pool")
public class HelloBean implements HelloBeanRemote {

```

This will override any default settings created in the management interfaces.

2. The `@org.jboss.ejb3.annotation.Pool` annotation is part of the JBoss EJB3 External API and must be added as a dependency. If you are using Maven, the following dependency should be added to your `pom.xml` file:

```

<dependency>
  <groupId>org.jboss.ejb3</groupId>
  <artifactId>jboss-ejb3-ext-api</artifactId>
  <version>2.1.0</version>
</dependency>

```

[Report a bug](#)

21.3. CONFIGURING EJB THREAD POOLS

21.3.1. Enterprise Bean Thread Pools

JBoss EAP 6 maintains number of instances of Java thread objects in memory for use by enterprise bean services, including remote invocation, the timer service, and asynchronous invocation.

This technique is called thread pooling. It provides improved performance by eliminating the overhead of thread creation and gives the system administrator a mechanism for controlling resource usage.

Multiple thread pools can be created with different parameters and each service can be allocated a different thread pool.

[Report a bug](#)

21.3.2. Create a Thread Pool

EJB Thread pools can be created using the Management Console or the CLI.

Procedure 21.10. Create an EJB Thread Pool using the Management Console

1. Login to the Management Console. [Section 3.3.2, "Log in to the Management Console"](#)
2. Click on the **Configuration** tab at the top of the screen.
3. Expand the **Container** menu and select **EJB 3**.
4. Select the **Thread Pools** tab and click **Add**.
5. Specify the **Name** and **Max Threads** values.
6. Click **Save** to finish.

Procedure 21.11. Create a Thread Pool using the CLI

1. Launch the CLI tool and connect to your server. Refer to [Section 3.4.4, "Connect to a Managed Server Instance Using the Management CLI"](#).
2. Use the **add** operation with the following syntax.

```
/subsystem=ejb3/thread-pool=THREAD_POOL_NAME:add(max-threads=MAX_SIZE)
```

- Replace *THREAD_POOL_NAME* with the name of the thread pool.
- Replace *MAX_SIZE* with the maximum size of the thread pool.

3. Use the **read-resource** operation to confirm the creation of the bean pool.

```
/subsystem=ejb3/thread-pool=THREAD_POOL_NAME:read-resource
```

Example 21.7. Create a Thread Pool using the CLI

```
[standalone@localhost:9999 /] /subsystem=ejb3/thread-pool=my-test-pool:add(max-threads=20)
{"outcome" => "success"}
```

Example 21.8. XML Configuration Sample

```
<subsystem xmlns="urn:jboss:domain:ejb3:1.5">
...
<thread-pools>
...
<thread-pool name="my-test-pool" max-threads="20"/>
</thread-pools>
...
</subsystem>
```


[Report a bug](#)

21.3.3. Remove a Thread Pool

Unused EJB thread pools can be removed using the Management Console.

Prerequisites

- The thread pool that you want to remove cannot be in use. Refer to the following tasks to ensure that the thread pool is not in use:
 - [Section 21.6.2, “Configure the EJB3 Timer Service”](#)
 - [Section 21.7.2, “Configure the EJB3 Asynchronous Invocation Service Thread Pool”](#)
 - [Section 21.8.2, “Configure the EJB3 Remote Service”](#)

Procedure 21.12. Remove an EJB thread pool using the Management Console

1. Login to the Management Console. [Section 3.3.2, “Log in to the Management Console”](#) .
2. Click on the **Configuration** tab at the top of the screen. Expand the **Container** menu and select **EJB 3**. Select the **Thread Pools** tab.
3. Select the thread pool to you want to remove.
4. Click **Remove**. The **Remove Item** dialog appears.
5. Click **Confirm**.

Procedure 21.13. Remove a thread pool using the CLI

1. Launch the CLI tool and connect to your server. Refer to [Section 3.4.4, “Connect to a Managed Server Instance Using the Management CLI”](#) .
2. Use the **remove** operation with the following syntax.

```
/subsystem=ejb3/thread-pool=THREADPOOLNAME:remove
```

- Replace *THREADPOOLNAME* with the name of the thread pool.

Example 21.9. Removing a Thread Pool using the CLI

```
[standalone@localhost:9999 /] /subsystem=ejb3/thread-pool=ACCTS_THREADS:remove
{"outcome" => "success"}
```

[Report a bug](#)

21.3.4. Edit a Thread Pool

JBoss Administrators can edit Thread Pools using the Management Console and the CLI.

Procedure 21.14. Edit a Thread Pool using the Management Console

1. Login to the Management Console. [Section 3.3.2, "Log in to the Management Console"](#) .
2. Click on the **Configuration** tab at the top of the screen. Expand the **Container** menu and select **EJB 3**. Select the **Thread Pools** tab.
3. Select the thread pool you want to edit.
4. Click **Edit**.
5. Edit the details you want to change. Only the **Thread Factory**, **Max Threads**, **Keepalive Timeout**, and **Keepalive Timeout Unit** values can be edited.
6. Click **Save** to finish.

Procedure 21.15. Edit a thread pool using the CLI

1. Launch the CLI tool and connect to your server. Refer to [Section 3.4.4, "Connect to a Managed Server Instance Using the Management CLI"](#).
2. Use the **write_attribute** operation with the following syntax for each attribute of the thread pool to be changed.

```
/subsystem=ejb3/thread-pool=THREADPOOLNAME:write-attribute(name="ATTRIBUTE",
value="VALUE")
```

- Replace *THREADPOOLNAME* with the name of the thread pool.
 - Replace *ATTRIBUTE* with the name of the attribute to be edited. The attributes that can be edited in this way are **keepalive-time**, **max-threads**, and **thread-factory**.
 - Replace *VALUE* with the required value of the attribute.
3. Use the **read-resource** operation to confirm the changes to the thread pool.

```
/subsystem=ejb3/thread-pool=THREADPOOLNAME:read-resource
```

IMPORTANT

When changing the value of the **keepalive-time** attribute with the CLI the required value is an object representation. It has the following syntax.

```
/subsystem=ejb3/thread-pool=THREADPOOLNAME:write-attribute(name="keepalive-
time", value={"time" => "VALUE", "unit" => "UNIT"})
```

Example 21.10. Set the Maxsize Value of a Thread Pool using the CLI

```
[standalone@localhost:9999 /] /subsystem=ejb3/thread-pool=HSThreads:write-
attribute(name="max-threads", value="50")
{"outcome" => "success"}
```

Example 21.11. Set the `keepalive-time` Time Value of a Thread Pool using the CLI

```
[standalone@localhost:9999 /] /subsystem=ejb3/thread-pool=HSThreads:write-
attribute(name="keepalive-time", value={"time"=>"150"})
{"outcome" => "success"}
```

[Report a bug](#)

21.4. CONFIGURING SESSION BEANS

21.4.1. Session Bean Access Timeout

Stateful and Singleton Session Beans have an access timeout value specified for managing concurrent access. This value is the period of time that a request to a session bean method can be blocked before it will timeout.

The timeout value and the time unit used can be specified using the `@javax.ejb.AccessTimeout` annotation on the method. It can be specified on the session bean (which applies to all the bean's methods) and on specific methods to override the configuration for the bean.

If they are not specified JBoss EAP 6 supplies a default timeout value of 5000 milliseconds.

Refer to the Javadocs for `AccessTimeout` at <http://docs.oracle.com/javaee/6/api/javax/ejb/AccessTimeout.html>

[Report a bug](#)

21.4.2. Set Default Session Bean Access Timeout Values

JBoss Administrators can specify the default timeout values for Singleton and Stateful session beans. The default timeout values can be changed using the Management Console or the CLI. The default value is 5000 milliseconds.

Procedure 21.16. Set Default Session Bean Access Timeout Values using the Management Console

1. Login to the Management Console. See [Section 3.3.2, "Log in to the Management Console"](#).
2. Click on the **Configuration** tab at the top of the screen. Expand the **Container** menu and select **EJB 3**. Select the **Container** tab.
3. Click **Edit**. The fields in the **Details** area can now be edited.
4. Enter the required values in the **Stateful Access Timeout** and/or **Singleton Access Timeout** text boxes.
5. Click **Save** to finish.

Procedure 21.17. Set Session Bean Access Timeout Values Using the CLI

1. Launch the CLI tool and connect to your server. Refer to [Section 3.4.4, "Connect to a Managed Server Instance Using the Management CLI"](#).
2. Use the **write-attribute** operation with the following syntax.

```
/subsystem=ejb3:write-attribute(name="BEANTYPE", value=TIME)
```

- Replace *BEANTYPE* with **default-stateful-bean-access-timeout** for Stateful Session Beans, or **default-singleton-bean-access-timeout** for Singleton Session Beans.
 - Replace *TIME* with the required timeout value.
3. Use the **read-resource** operation to confirm the changes.

```
/subsystem=ejb3:read-resource
```

Example 21.12. Setting the Default Stateful Bean Access Timeout value to 9000 with the CLI

```
[standalone@localhost:9999 /] /subsystem=ejb3:write-attribute(name="default-stateful-bean-
access-timeout", value=9000)
{"outcome" => "success"}
```

Example 21.13. XML Configuration Sample

```
<subsystem xmlns="urn:jboss:domain:ejb3:1.2">
  <session-bean>
    <stateless>
      <bean-instance-pool-ref pool-name="slsb-strict-max-pool"/>
    </stateless>
    <stateful default-access-timeout="5000" cache-ref="simple"/>
    <singleton default-access-timeout="5000"/>
  </session-bean>
</subsystem>
```

[Report a bug](#)

21.4.3. Session Bean Transaction Timeout

The **TransactionTimeout** annotation is used to specify the transaction timeout for a given method. The value of the annotation is the timeout used in the given unit element. It must be a positive integer or 0. Whenever 0 is specified, the default domain configured timeout is used.

The unit element specifies the measure of the value.



NOTE

Specifying a measure lesser than seconds is considered an error, even when the computed value will result in an integral number of seconds. For example:
@TransactionTimeout(value = 1000, unit=TimeUnit.MILLISECONDS)

Specifying Transaction Timeout in the Deployment Descriptor

The **trans-timeout** element is used to define the transaction timeout for business, home, component, and message listener interface methods; no interface view methods; web service endpoint methods; and

timeout callback methods. The **trans-timeout** element resides in the **urn:trans-timeout** namespace and is part of the standard **container-transaction** element as defined in the **jboss** namespace.

Example 21.14. trans-timeout XML Configuration Sample

```
<ejb-name>*</ejb-name>
<tx:trans-timeout>
<tx:timeout>2</tx:timeout>
<tx:unit>Seconds</tx:unit>
</tx:trans-timeout>
```

ejb-name can be specified to a particular EJB name, or a wildcard (*). Specifying a wildcard (*) for the **ejb-name** means that this particular transaction timeout will be the default for all EJBs in the application.

[Report a bug](#)

21.4.4. Configure Stateful Session Bean Cache

In JBoss EAP 6, stateful EJB cache is configured in the **ejb3** subsystem of the server configuration file. The following procedure describes how to configure stateful EJB cache and stateful timeout.

Procedure 21.18. Configure Stateful EJB Cache

1. Find the **<caches>** element in the **ejb3** subsystem of the server configuration file. Add a **<cache>** element. The following example creates a cache named "my-cache".

```
<cache name="my-cache" passivation-store-ref="my-cache-file" aliases="my-custom-cache"/>
```

2. Find the **<passivation-stores>** element in the **ejb3** subsystem of the server configuration file. Create a **<file-passivation-store>** for the cache defined in the previous step.

```
<file-passivation-store name="my-cache-file" idle-timeout="1260" idle-timeout-unit="SECONDS" max-size="200"/>
```

3. The **ejb3** subsystem configuration should now look like the following example.

```
<subsystem xmlns="urn:jboss:domain:ejb3:1.4">
...
<caches>
<cache name="simple" aliases="NoPassivationCache"/>
<cache name="passivating" passivation-store-ref="file" aliases="SimpleStatefulCache"/>
<cache name="clustered" passivation-store-ref="infinispan" aliases="StatefulTreeCache"/>
<cache name="my-cache" passivation-store-ref="my-cache-file" aliases="my-custom-cache"/>
</caches>
<passivation-stores>
<file-passivation-store name="file" idle-timeout="120" idle-timeout-unit="SECONDS" max-size="500"/>
<cluster-passivation-store name="infinispan" cache-container="ejb"/>
<file-passivation-store name="my-cache-file" idle-timeout="1260" idle-timeout-
```

```

unit="SECONDS" max-size="200"/>
</passivation-stores>
...
</subsystem>

```

The passivating cache, "my-cache", passivates stateful session beans to the file system as configured in the "my-cache-file" passivation store, which has the **idle-timeout**, **idle-timeout-unit** and **max-size** options.

4. Create a **jboss-ejb3.xml** file in the EJB JAR **META-INF/** directory. The following example configures the EJBs to use the cache defined in the previous steps.

```

<jboss:ejb-jar xmlns:jboss="http://www.jboss.com/xml/ns/javaee"
  xmlns="http://java.sun.com/xml/ns/javaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:c="urn:ejb-cache:1.0"
  xsi:schemaLocation="http://www.jboss.com/xml/ns/javaee
http://www.jboss.org/j2ee/schema/jboss-ejb3-2_0.xsd http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/ejb-jar_3_1.xsd"
  version="3.1"
  impl-version="2.0">
  <assembly-descriptor>
    <c:cache>
      <ejb-name>*/</ejb-name>
      <c:cache-ref>my-cache</c:cache-ref>
    </c:cache>
  </assembly-descriptor>
</jboss:ejb-jar>

```

5. To method to configure a timeout value depends on whether you are implementing EJB 2 or EJB 3.
 - EJB 3 introduced annotations, so you can specify the **javax.ejb.StatefulTimeout** annotation in the EJB code as follows.

```

@StatefulTimeout(value = 1320, unit=java.util.concurrent.TimeUnit.SECONDS)
@Stateful
@Remote(MyStatefulEJBRemote.class)
public class MyStatefulEJB implements MyStatefulEJBRemote {
  ...
}

```

The **@StatefulTimeout** value can be set to one of the following.

- A value of **0** means the bean is immediately eligible for removal.
- A value greater than **0** indicates a timeout value in the units specified by the **unit** parameter. The default timeout unit is **MINUTES**. If you are using a passivating cache configuration and the **idle-timeout** value is less than the **StatefulTimeout** value, JBoss EAP will passivate the bean when it is idle for the **idle-timeout** period specified. The bean is then eligible for removal after the **StatefulTimeout** period specified.
- A value of **-1** means the bean will never be removed due to timeout. If you are using a passivating cache configuration and the bean is idle for **idle-timeout**, JBoss EAP will passivate the bean instance to the **passivation-store**.

- Values less than **-1** are not valid.
- For both EJB 2 and EJB 3, you can configure the stateful timeout in the **ejb-jar.xml** file.

```
<?xml version="1.0" encoding="UTF-8"?>
<ejb-jar xmlns="http://java.sun.com/xml/ns/javaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/ejb-jar_3_1.xsd"
  version="3.1">
<enterprise-beans>
<session>
  <ejb-name>HelloBean</ejb-name>
  <session-type>Stateful</session-type>
  <stateful-timeout>
    <timeout>1320</timeout>
    <unit>Seconds</unit>
  </stateful-timeout>
</session>
</enterprise-beans>
</ejb-jar>
```

- For both EJB 2 and EJB 3, you can configure the stateful timeout in the **jboss-ejb3.xml** file.

```
<jboss:ejb-jar xmlns:jboss="http://www.jboss.com/xml/ns/javaee"
  xmlns="http://java.sun.com/xml/ns/javaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:c="urn:ejb-cache:1.0"
  xsi:schemaLocation="http://www.jboss.com/xml/ns/javaee
http://www.jboss.org/j2ee/schema/jboss-ejb3-2_0.xsd http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/ejb-jar_3_1.xsd"
  version="3.1"
  impl-version="2.0">
<enterprise-beans>
<session>
  <ejb-name>HelloBean</ejb-name>
  <session-type>Stateful</session-type>
  <stateful-timeout>
    <timeout>1320</timeout>
    <unit>Seconds</unit>
  </stateful-timeout>
</session>
</enterprise-beans>
<assembly-descriptor>
  <c:cache>
    <ejb-name>*</ejb-name>
    <c:cache-ref>my-cache</c:cache-ref>
  </c:cache>
</assembly-descriptor>
</jboss:ejb-jar>
```

Additional Information

- To disable passivation of stateful session beans, do one of the following:

- If you implement stateful session beans using EJB 3 annotations, you can disable the passivation of the stateful session bean the annotation `@org.jboss.ejb3.annotation.Cache("NoPassivationCache")`
- If the stateful session bean is configured in the `jboss-ejb3.xml` file, set the `<c:cache-ref>` element value to "simple", which is the equivalent of `NoPassivationCache`.

```
<c:cache-ref>simple</c:cache-ref>
```

- EJB cache policy "LRUStatefulContextCachePolicy" has been changed in JBoss EAP 6 so it is impossible to have 1-to-1 configuration mapping in JBoss EAP 6.
- In JBoss EAP 6, you can set up the following cache properties:
 - Bean life time is configured using the `@StatefulTimeout` in EJB 3.1.
 - Configure passivation of a bean to disk in the `ejb3` subsystem of the server configuration file using the `idle-timeout` attribute of the `<file-passivation-store>` element.
 - Configure the maximum size of the passivation store in the `ejb3` subsystem of the server configuration file using the `max-size` attribute of the `<file-passivation-store>` element.
- In JBoss EAP 6, you can not configure the following cache properties:
 - The minimum and maximum numbers in memory cache.
 - The minimum numbers in passivation store.
 - The `*-period` configurations that control the frequency of cache operations.

[Report a bug](#)

21.5. CONFIGURING MESSAGE-DRIVEN BEANS

21.5.1. Set Default Resource Adapter for Message-Driven Beans

JBoss Administrators can specify the default resource adapter used by message-driven beans. The default resource adapter can be specified using the Management Console and the CLI. The default resource adapter supplied with JBoss EAP 6 is `hornetq-ra`.

Procedure 21.19. Set the Default Resource Adapter for Message-Driven Beans using the Management Console

1. Login to the Management Console. [Section 3.3.2, "Log in to the Management Console"](#)
2. Click on the **Configuration** tab at the top of the screen. Expand the **Container** menu and select **EJB 3**. Select the **Container** tab.
3. Click **Edit**. The fields in the **Details** area can now be edited.
4. Enter the name of the resource adapter to be used in the **Default Resource Adapter** text box.
5. Click **Save** to finish.

Procedure 21.20. Set the Default Resource Adapter for Message-Driven Beans using the CLI

1. Launch the CLI tool and connect to your server. Refer to [Section 3.4.4, “Connect to a Managed Server Instance Using the Management CLI”](#).
2. Use the **write-attribute** operation with the following syntax.

```
/subsystem=ejb3:write-attribute(name="default-resource-adapter-name",
value="RESOURCE-ADAPTER")
```

Replace *RESOURCE-ADAPTER* with name of the resource adapter to be used.

3. Use the **read-resource** operation to confirm the changes.

```
/subsystem=ejb3:read-resource
```

Example 21.15. Set the Default Resource Adapter for Message-Driven Beans using the CLI

```
[standalone@localhost:9999 subsystem=ejb3] /subsystem=ejb3:write-attribute(name="default-
resource-adapter-name", value="EDIS-RA")
{"outcome" => "success"}
[standalone@localhost:9999 subsystem=ejb3]
```

Example 21.16. XML Configuration Sample

```
<subsystem xmlns="urn:jboss:domain:ejb3:1.2">
  <mdb>
    <resource-adapter-ref resource-adapter-name="hornetq-ra"/>
    <bean-instance-pool-ref pool-name="mdb-strict-max-pool"/>
  </mdb>
</subsystem>
```

[Report a bug](#)

21.6. CONFIGURING THE EJB3 TIMER SERVICE

21.6.1. EJB3 Timer Service

The EJB3 Timer Service is a standard Java EE 6 service for scheduling the invocation of the methods from enterprise beans. Stateless session beans, singleton session beans, and message-driven beans can all schedule any of their methods for callback at specified times. Method callback can occur at a specific time, after a duration, at a recurring interval, or on a calendar-based schedule.

[Report a bug](#)

21.6.2. Configure the EJB3 Timer Service

The EJB3 Timer Service can be configured via either the Management Console or Management CLI. You can configure the thread pool used for scheduled bean invocation, and either the directory or datasource used to store the Timer Service data. You might change the default Timer Service directory if faster storage is available than the default directory.

Procedure 21.21. Configure the EJB3 Timer Service Thread Pool via the Management Console

Prerequisite

- The thread pool to be used by the EJB3 Timer Service must already have been created.
1. Login to the Management Console.
 2. Click on the **Configuration** tab at the top of the screen. Expand the **Container** menu and select **EJB 3**. Select the **Services** tab, click on **Timer Service**. Click **Edit**.
 3. Click on the **EJB3 Thread Pool** drop-down list and click on the preferred thread pool's name.
 4. Restart the JBoss EAP instance.

Procedure 21.22. Configure the EJB3 Timer Service Thread Pool via the Management CLI



NOTE

Add the prefix `/profile=PROFILE_NAME` to the command for a managed domain.

1. Run the following Management CLI command.

```
/subsystem=ejb3/service=timer-service:write-attribute(name=thread-pool-name,value="thread-pool-name")
```

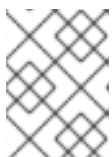
2. Restart the JBoss EAP instance.

Procedure 21.23. Configure the EJB3 Timer Service Directory via the Management Console

1. Login to the Management Console.
2. Click on the **Configuration** tab at the top of the screen. Expand the **Container** menu and select **EJB 3**. Select the **Services** tab, click on **Timer Service**. Click **Edit**.
3. Enter your desired values into the **Path** and **Relative To** fields.
4. Click **Save**.
5. Restart the JBoss EAP instance.

Procedure 21.24. Configure the EJB3 Timer Service Directory via the Management CLI

1. Depending on which paths you want to change, run one or both of the following Management CLI commands. For either path you can use a system value - for example, `${jboss.server.data.dir}`.



NOTE

Add the prefix `/profile=PROFILE_NAME` to the command for a managed domain.

```
/subsystem=ejb3/service=timer-service/file-data-store=default-file-store:write-attribute(name=path,value="path")
```

```
/subsystem=ejb3/service=timer-service/file-data-store=default-file-store:write-attribute(name=relative-to,value="relative-path")
```

- Restart the JBoss EAP instance.

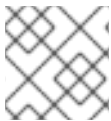
Procedure 21.25. Configure the EJB3 Timer Service to use a Datasource via the Management CLI

From JBoss EAP 6.4 you can configure the EJB3 Timer Service to use a datasource instead of a local directory. There is a minor performance cost to this option but it has the advantage of decreasing risk to timer data in the event of a local storage issue.

Once the EJB3 Timer Service is configured to use a datasource, you then must either configure an EJB deployment to use the datastore or configure it as the default for all deployments. For instructions on how to do so, see the procedure *Configure one or all EJB3 Deployments to use the Datasource* .

Prerequisite

- The datasource to be used by the EJB3 Timer Service must already exist and the underlying database must support and be configured for READ_COMMITTED or SERIALIZABLE isolation mode.



NOTE

Add the prefix **/profile=PROFILE_NAME** to the command for a managed domain.

- Run the following Management CLI command.
 - `datastore_name` - A name of your choice.
 - `datasource_name` - The name of the datasource to be used for storage.
 - `database` - either **postgresql**, **mssql**, **sybase**, **mysql**, **oracle**, **db2**, or **hsqldb**.
 - `partition_name` - A name of your choice. This attribute is used to distinguish timers pertaining to a particular server instance if multiple JBoss EAP instances share the same database for storing EJB timers. In this case, every server instance should have its own partition name. If the database is used by only one server instance, you can leave this attribute blank.

```
/subsystem=ejb3/service=timer-service/database-data-store=datastore_name:add(datasource-jndi-name='java:/datasource_name', database='database', partition='partition_name')
```

Procedure 21.26. Configure one or all EJB3 Deployments to use the Datasource

Either configure an EJB3 deployment to use the Timer Service's datasource or configure it as the default for *all* deployments.

- To configure an EJB3 deployment to use the datasource, edit the **jboss-ejb3.xml** of the deployment so the **timer** section looks as follows. Replace **datastore_name** with the name of the datastore.

```
[<assembly-descriptor>
  <timer:timer>
    <ejb-name> * </ejb-name>
    <timer:persistence-store-name>datastore_name</timer:persistence-store-name>
  </timer:timer>
</assembly-descriptor>
```

- o To configure the datasource as the default for *all* deployments, run the following Management CLI command, then restart the JBoss EAP instance. Replace **datastore_name** with the name of the datastore.



NOTE

Add the prefix **/profile=PROFILE_NAME** to the command for a managed domain.

```
[/subsystem=ejb3/service=timer-service:write-attribute(name=default-data-
store,value=datastore_name)
```

[Report a bug](#)

21.7. CONFIGURING THE EJB ASYNCHRONOUS INVOCATION SERVICE

21.7.1. EJB3 Asynchronous Invocation Service

The Asynchronous Invocation Service is an Enterprise JavaBeans container service that manages asynchronous invocation of session bean methods. This service maintains a configurable number of threads (a thread pool) that are allocated for asynchronous method execution.

Enterprise JavaBeans 3.1 allows for any method of a session bean (stateful, stateless or singleton) to be annotated to permit asynchronous execution.

[Report a bug](#)

21.7.2. Configure the EJB3 Asynchronous Invocation Service Thread Pool

JBoss Administrators can configure the EJB3 Asynchronous Invocation Service in the JBoss EAP 6 Management Console to use a specific thread pool.

Procedure 21.27. Configure the EJB3 Asynchronous Invocation Service thread pool

1. Login to the Management Console. See [Section 3.3.2, "Log in to the Management Console"](#) .
2. Click on the **Configuration** tab at the top of the screen. Expand the **Container** menu and select **EJB 3**. Select the **Services** tab, click on **Async Service**.
3. Click **Edit**.
4. Select the EJB3 thread pool to use from the list. The thread pool must have been already created.

5. Click **Save** to finish.

[Report a bug](#)

21.8. CONFIGURING THE EJB3 REMOTE INVOCATION SERVICE

21.8.1. EJB3 Remote Service

The EJB3 Remote Service manages the remote execution of Enterprise Beans with remote business interfaces.

[Report a bug](#)

21.8.2. Configure the EJB3 Remote Service

JBoss Administrators can configure the EJB3 Remote Service in the JBoss EAP 6 Management Console. The features that can be configured are the thread pool that is used for remote bean invocation and the connector on which the EJB3 remoting channel is registered.

Procedure 21.28. Configure the EJB3 Remote Service

1. Login to the Management Console. See [Section 3.3.2, “Log in to the Management Console”](#) .
2. Click on the **Configuration** tab at the top of the screen. Expand the **Container** menu and select **EJB 3**. Select the **Services** tab, click on **Remote Service**.
3. Click **Edit**.
4. You can select a different EJB3 thread pool used for the Remote Service if additional thread pools have been configured. You can change the connector used to register the EJB remoting channel.
5. Click **Save** to finish.

[Report a bug](#)

21.9. CONFIGURING EJB 2.X ENTITY BEANS

21.9.1. EJB Entity Beans

EJB Entity Beans are a type of enterprise bean from version 2.x of the EJB specification that represented persistent data that was maintained in a database. Entity beans have been superseded by JPA entities and officially listed for removal (pruning) from future versions of the specification. Red Hat does not recommend the use of Entity Beans except for backwards compatibility.

Support for Entity Beans is disabled by default in JBoss EAP 6.



NOTE

JBoss EAP 6.x supports EJB 2.0 and above, with the exception that EJB 2.0 deployment descriptors only work in JBoss EAP 6.4 and above.

[Report a bug](#)

21.9.2. Container-Managed Persistence

Container-Managed Persistence (CMP) is an application server provided service that provides data persistence for Entity beans.

[Report a bug](#)

21.9.3. Enable EJB 2.x Container-Managed Persistence

Container-Managed Persistence (CMP) is handled by the **org.jboss.as.cmp** extension. CMP is enabled by default in the managed domain and standalone server full configurations, e.g. **standalone-full.xml**.

To enable CMP in a different configuration, add the **org.jboss.as.cmp** module to the list of enabled extensions in the server configuration file.

```
<extensions>
  <extension module="org.jboss.as.cmp"/>
</extensions>
```

Once the extension has been added, you must also add the following element in the profile's XML configuration file under the `<profile>` element.

```
<subsystem xmlns="urn:jboss:domain:cmp:1.1"/>
```

To disable CMP in a server configuration, remove the extension entry for the **org.jboss.as.cmp** module.

[Report a bug](#)

21.9.4. Configure EJB 2.x Container-Managed Persistence

The EJB 2.x Container Managed Persistence (CMP) subsystem can be configured to specify any number of key generators. Key generators are used to generate unique keys to identify each entity persisted by the CMP service.

Two types of key generator can be defined: UUID-based and HiLo key generators.

UUID-based key generators

A UUID-based key generator creates keys using Universally Unique Identifiers. UUID key generators only need to have a unique name, they have no other configuration.

UUID-based key generators can be added using the CLI using the following command syntax.

```
/subsystem=cmp/uuid-keygenerator=UNIQUE_NAME:add
```

Example 21.17. Add UUID Key Generator

To add a UUID-based key generator with the name of **uuid_identities**, use this CLI command:

```
/subsystem=cmp/uuid-keygenerator=uuid_identities:add
```

The XML configuration created by this command is:

```
<subsystem xmlns="urn:jboss:domain:cmp:1.0">
```

```

<key-generators>
  <uuid name="uuid_identities" />
</key-generators>
</subsystem>

```

HiLo Key Generators

HiLo key generators use a database to create and store entity identity keys. HiLo Key generators must have unique names and are configured with properties that specify the datasource used to store the data as well as the names of the table and columns that store the keys.

HiLo key generators can be added using the CLI using the following command syntax:

```
/subsystem=cmp/hilo-keygenerator=UNIQUE_NAME:/add(property=value, property=value, ...)
```

Example 21.18. Add a HiLo Key Generator

```
/subsystem=cmp/hilo-keygenerator=HiLoKeyGeneratorFactory:add(create-table=true,create-
table-ddl="create table HILOSEQUENCES (SEQUENCENAME varchar(50) not null,
HIGHVALUES integer not null, constraint hilo_pk primary key (SEQUENCENAME))",data-
source=java:jboss/datasources/ExampleDS, id-column=HIGHVALUES,sequence-
column=SEQUENCENAME,table-name=HILOSEQUENCES,sequence-name=general,block-
size=10)
```

The XML configuration created by this command is:

```

<subsystem xmlns="urn:jboss:domain:cmp:1.1">
  <key-generators>
    <hilo name="HiLoKeyGeneratorFactory">
      <block-size>10</block-size>
      <create-table>true</create-table>
      <create-table-ddl>create table HILOSEQUENCES (SEQUENCENAME varchar(50) not
null, HIGHVALUES integer not null, constraint hilo_pk primary key
(SEQUENCENAME))</create-table-ddl>
      <data-source>java:jboss/datasources/ExampleDS</data-source>
      <id-column>HIGHVALUES</id-column>
      <sequence-column>SEQUENCENAME</sequence-column>
      <sequence-name>general</sequence-name>
      <table-name>HILOSEQUENCES</table-name>
    </hilo>
  </key-generators>
</subsystem>

```



NOTE

The block-size must be set to a value !=0, otherwise the generated PKey will not be incremented and therefore the creation of entities fails with a DuplicateKeyException.

**NOTE**

The `select-hi-ddl` must be set as 'FOR UPDATE' in case of cluster to ensure the consistency. All databases do not support the locking feature.

[Report a bug](#)

21.9.5. CMP Subsystem Properties for HiLo Key Generators

Table 21.1. CMP Subsystem Properties for HiLo Key Generators

Property	Data type	Description
block-size	long	The block size.
create-table	boolean	If set to TRUE , a table called table-name will be created using the contents of create-table-ddl if that table is not found.
create-table-ddl	string	The DDL commands used to create the table specified in table-name if the table is not found and create-table is set to TRUE .
data-source	token	The data source used to connect to the database.
drop-table	boolean	To determine whether to drop the tables.
id-column	token	The ID column name.
select-hi-ddl	string	The SQL command which will return the largest key currently stored.
sequence-column	token	The sequence column name.
sequence-name	token	The name of the sequence.
table-name	token	The name of the table used to store the key information.

[Report a bug](#)

CHAPTER 22. JAVA CONNECTOR ARCHITECTURE (JCA)

22.1. INTRODUCTION

22.1.1. About the Java EE Connector API (JCA)

JBoss EAP 6 provides full support for the Java EE Connector API (JCA) 1.6 specification. See [JSR 322: Java EE Connector Architecture 1.6](#) for more information about the JCA specification.

A resource adapter is a component that implements the Java EE Connector API architecture. It is similar to a datasource object, however, it provides connectivity from an Enterprise Information System (EIS) to a broader range of heterogeneous systems, such as databases, messaging systems, transaction processing, and Enterprise Resource Planning (ERP) systems.



NOTE

Java Platform Enterprise Edition 6 comes with the **javax.resource.cci** package. The **javax.resource.cci** package comprises the APIs that should be implemented by a resource adapter that supports the **Common Client Interface** (CCI). **javax.resource.cci.ResultSet** is a member of this package and extends **java.sql.ResultSet**. The interface of **java.sql.ResultSet** depends on the Java version used, and hence when using **Common Client Interface** (CCI), all applications should assume that only **java.sql.ResultSet** methods from Java 6 can be used for data interaction.

[Report a bug](#)

22.1.2. Java Connector Architecture (JCA)

The Java EE Connector Architecture (JCA) defines a standard architecture for Java EE systems to external heterogeneous Enterprise Information Systems (EIS). Examples of EISs include Enterprise Resource Planning (ERP) systems, mainframe transaction processing (TP), databases and messaging systems.

JCA 1.6 provides features for managing:

- connections
- transactions
- security
- life-cycle
- work instances
- transaction inflow
- message inflow

JCA 1.6 was developed under the Java Community Process as JSR-322, <http://jcp.org/en/jsr/detail?id=313>.

Alternative managed connection pools

JBoss EAP 6.4 features the following alternative pool implementations:

- `org.jboss.jca.core.connectionmanager.pool.mcp.SemaphoreArrayListManagedConnectionPool`: This is the default connection pool.
- `org.jboss.jca.core.connectionmanager.pool.mcp.SemaphoreConcurrentLinkedQueueManagedConnectionPool`: This connection pool sometimes provides better performance profile and is enabled by using the system property -
`Dironjacamar.mcp=org.jboss.jca.core.connectionmanager.pool.mcp.SemaphoreConcurrentLinkedQueueManagedConnectionPool`
- `org.jboss.jca.core.connectionmanager.pool.mcp.LeakDumperManagedConnectionPool`: This connection pool is used only for debugging purposes. For more information about the `LeakDetectorPool`, refer to http://www.ironjacamar.org/doc/userguide/1.2/en-US/html/ch04.html#configuration_ironjacamar_leakpool

[Report a bug](#)

22.1.3. Resource Adapters

A resource adapter is a deployable Java EE component that provides communication between a Java EE application and an Enterprise Information System (EIS) using the Java Connector Architecture (JCA) specification. A resource adapter is often provided by EIS vendors to allow easy integration of their products with Java EE applications.

An Enterprise Information System can be any other software system within an organization. Examples include Enterprise Resource Planning (ERP) systems, database systems, e-mail servers and proprietary messaging systems.

A resource adapter is packaged in a Resource Adapter Archive (RAR) file which can be deployed to JBoss EAP 6. A RAR file may also be included in an Enterprise Archive (EAR) deployment.

[Report a bug](#)

22.2. CONFIGURE THE JAVA CONNECTOR ARCHITECTURE (JCA) SUBSYSTEM

The JCA subsystem in the JBoss EAP 6 configuration file controls the general settings for the JCA container and resource adapter deployments.

Key elements of the JCA subsystem

Archive validation

- This setting whether archive validation will be performed on the deployment units.
- The following table describes the attributes you can set for archive validation.

Table 22.1. Archive validation attributes

Attribute	Default Value	Description
enabled	true	Specifies whether archive validation is enabled.

Attribute	Default Value	Description
fail-on-error	true	Specifies whether an archive validation error report fails the deployment.
fail-on-warn	false	Specifies whether an archive validation warning report fails the deployment.

- If an archive does not implement the Java EE Connector Architecture specification correctly and archive validation is enabled, an error message will display during deployment describing the problem. For example:

```
Severity: ERROR
Section: 19.4.2
Description: A ResourceAdapter must implement a "public int hashCode()" method.
Code: com.mycompany.myproject.ResourceAdapterImpl
```

```
Severity: ERROR
Section: 19.4.2
Description: A ResourceAdapter must implement a "public boolean equals(Object)"
method.
Code: com.mycompany.myproject.ResourceAdapterImpl
```

- If archive validation is not specified, it is considered present and the **enabled** attribute defaults to true.

Bean validation

- This setting determines whether bean validation (JSR-303) will be performed on the deployment units.
- The following table describes the attributes you can set for bean validation.

Table 22.2. Bean validation attributes

Attribute	Default Value	Description
enabled	true	Specifies whether bean validation is enabled.

- If bean validation is not specified, it is considered present and the **enabled** attribute defaults to true.

Work managers

- There are two types of work managers:

Default work manager

The default work manager and its thread pools.

Custom work manager

A custom work manager definition and its thread pools.

- The following table describes the attributes you can set for work managers.

Table 22.3. Work manager attributes

Attribute	Description
name	Specifies the name of the work manager. This is required for custom work managers.
short-running-threads	Thread pool for standard Work instances. Each work manager has one short-running thread pool.
long-running-threads	Thread pool for JCA 1.6 Work instances that set the LONG_RUNNING hint. Each work manager can have one optional long-running thread pool.

- The following table describes the attributes you can set for work manager thread pools.

Table 22.4. Thread pool attributes

Attribute	Description
allow-core-timeout	Boolean setting that determines whether core threads may time out. The default value is false.
core-threads	The core thread pool size. This must be equal to or smaller than the maximum thread pool size.
queue-length	The maximum queue length.
max-thread	The maximum thread pool size.
keepalive-time	Specifies the amount of time that pool threads should be kept after doing work.
thread-factory	Reference to the thread factory .

Bootstrap contexts

- Used to define custom bootstrap contexts.
- The following table describes the attributes you can set for bootstrap contexts.

Table 22.5. Bootstrap context attributes

Attribute	Description
name	Specifies the name of the bootstrap context.
workmanager	Specifies the name of the work manager to use for this context.

Cached connection manager

- Used for debugging connections and supporting lazy enlistment of a connection in a transaction, tracking whether they are used and released properly by the application.
- The following table describes the attributes you can set for the cached connection manager.

Table 22.6. Cached connection manager attributes

Attribute	Default Value	Description
debug	false	Outputs warning on failure to explicitly close connections.
error	false	Throws exception on failure to explicitly close connections.

Procedure 22.1. Configure the JCA subsystem using the Management Console

The JCA subsystem of JBoss EAP 6 can be configured in the Management Console. The JCA configuration options are located in slightly different places in the Management Console depending on how the server is being run.

1. Click on the **Configuration** tab at the top of the screen. Expand the **Connector** menu and select **JCA**.
2. If the server is running in Domain mode, select a profile from the **Profile** drop-down menu at top left.
3. Configure the settings for the JCA subsystem using the three tabs.
 - a. **Common Config**
The **Common Config** tab contains settings for the cached connection manager, archive validation and bean validation (JSR-303). Each of these is contained in their own tab as well. These settings can be changed by opening the appropriate tab, clicking the edit button, making the required changes, and then clicking on the save button.

RED HAT JBOSS® ENTERPRISE APPLICATION PLATFORM 6.3.0.Alpha3 Messages: 0 admin

Home Configuration Domain Runtime Administration

Profile: full

Subsystems

- Connector
 - JCA**
 - Datasources
 - Resource Adapters
 - Mail
 - Container
 - Core
 - Infinispan
 - Messaging
 - Security
 - Web
- General Configuration

COMMON CONFIG BOOTSTRAP CONTEXTS WORK MANAGER

JCA Subsystem

The Java EE Connector Architecture (JCA) subsystem providing general configuration for resource adapters.

Connection Manager Archive Validation **Bean Validation**

[Need Help?](#)

[Edit](#)

Is Enabled?:	true
--------------	------

Figure 22.1. JCA Common Configuration

b. Work Managers

The **Work Manager** tab contains the list of configured Work Managers. New Work Managers can be added, removed, and their thread pools configured here. Each Work Manager can have one short-running thread pool and an optional long-running thread pool.

RED HAT JBOSS® ENTERPRISE APPLICATION PLATFORM 6.3.0.Alpha3 Messages: 0 admin

Home Configuration Domain Runtime Administration

Profile: full

Subsystems

- Connector
 - JCA**
 - Datasources
 - Resource Adapters
 - Mail
 - Container
 - Core
 - Infinispan
 - Messaging
 - Security
 - Web
- General Configuration

COMMON CONFIG BOOTSTRAP CONTEXTS **WORK MANAGER**

JCA Workmanager

Work manager for resource adapters.

Available Work Manager

[Add](#) [Remove](#)

Name	Option
default	View >

1-1 of 1

Figure 22.2. Work Managers

The thread pool attributes can be configured by clicking **View** on the selected resource adapter.

The screenshot shows the configuration console for the 'full' profile. The 'Configuration' tab is active, and the 'Work Manager' sub-tab is selected. The page title is 'Thread Pools'. Below the title, it states 'Work Manager: default' and 'Thread pool configurations used by a JCA workmanager.' A section titled 'Available Thread Pools' contains a table with the following data:

Name	Type	Max Threads
default	short-running	50
default	long-runnig	50

Below the table are links for 'Attributes' and 'Sizing'. An 'Edit' button is visible, and a 'Need Help?' link is in the top right. The configuration details for the selected thread pool are shown below:

- Name: default
- Keep Alive Timeout: 10
- Keepalive Timeout Unit: SECONDS
- Allow Core Timeout?: false
- Thread Factory:

Figure 22.3. Work Manager Thread Pools

c. Bootstrap Contexts

The **Bootstrap Contexts** tab contains the list of configured Bootstrap Contexts. New Bootstrap Context objects can be added, removed, and configured. Each Bootstrap Context must be assigned a Work Manager.

The screenshot shows the configuration console for the 'full' profile. The 'Configuration' tab is active, and the 'Bootstrap Contexts' sub-tab is selected. The page title is 'JCA Bootstrap Contexts'. Below the title, it states 'Bootstrap context for resource adapters. Each context does reference a workmanager.' A section titled 'Available Bootstrap Context' contains a table with the following data:

Name
default

Buttons for 'Add' and 'Remove' are visible. Below the table are links for 'Selection' and 'Need Help?'. The configuration details for the selected bootstrap context are shown below:

- Name: default
- Work Manager: default

Figure 22.4. Bootstrap Contexts

[Report a bug](#)

22.3. DEPLOY A RESOURCE ADAPTER

Resource adapters can be deployed to JBoss EAP 6 using the Management CLI tool, the Web-based Management Console, or by manually copying the files. The process is the same as other deployable artifacts.

Procedure 22.2. Deploy a resource adapter using the Management CLI

1. Open a command prompt for your operating system.
2. Connect to the Management CLI.

- For Linux, enter the following at the command line:

```
$ EAP_HOME/bin/jboss-cli.sh --connect  
$ Connected to standalone controller at localhost:9999
```

- For Windows, enter the following at a command line:

```
C:\>EAP_HOME\bin\jboss-cli.bat --connect  
C:\> Connected to standalone controller at localhost:9999
```

3. Deploy the resource adapter.

- To deploy the resource adapter to a standalone server, enter the following at a command line:

```
$ deploy path/to/resource-adapter-name.rar
```

- To deploy the resource adapter to all server groups in a managed domain, enter the following at a command line:

```
$ deploy path/to/resource-adapter-name.rar --all-server-groups
```

Procedure 22.3. Deploy a resource adapter using the Management Console

1. Login to the Management Console. See [Section 3.3.2, “Log in to the Management Console”](#).
2. Click on the **Runtime** tab at the top of the screen. Select **Manage Deployments**. Click **Add**.
3. Browse to the resource adapter archive and select it. Then click **Next**.
4. Verify the deployment names, then click **Save**.
5. The resource adapter archive should now appear in the list in a disabled state.
6. Enable the resource adapter.
 - In Domain mode, click **Assign**. Select which Server Groups to assign the resource adapter to. Click **Save** to finish.

- In Standalone mode, select the Application Component from the list. Click **En/Disable**. Click **Confirm** on the **Are You Sure?** dialog to enable the component.

Procedure 22.4. Deploy a resource adapter manually

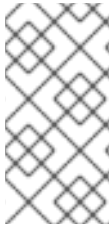
- Copy the resource adapter archive to the server deployments directory,
 - For a standalone server, copy the resource adapter archive to the ***EAP_HOME/standalone/deployments/*** directory.
 - For a managed domain, you must use the Management Console or Management CLI to deploy the resource adapter archive to the server groups.

[Report a bug](#)

22.4. CONFIGURE A DEPLOYED RESOURCE ADAPTER

JBoss administrators can configure resource adapters for JBoss EAP 6 using the Management CLI tool, the Web-based Management Console, or by manually editing the configuration the files.

Refer to the vendor document for your resource adapter for information about supported properties and other details.



NOTE

In the following procedure, the command line you must type follows the **[standalone@localhost:9999 /]** prompt. Do not type the text within the curly braces. That is the output you should see as a result of the command, for example, **{"outcome" => "success"}**.

Procedure 22.5. Configure a resource adapter using the Management CLI

1. Open a command prompt for your operating system.
2. Connect to the Management CLI.
 - For Linux, enter the following at the command line:

```
$ EAP_HOME/bin/jboss-cli.sh --connect
```

You should see the following result output:

```
$ Connected to standalone controller at localhost:9999
```

- For Windows, enter the following at a command line:

```
C:\>EAP_HOME\bin\jboss-cli.bat --connect
```

You should see the following result output:

```
C:\> Connected to standalone controller at localhost:9999
```

3. Add the resource adapter configuration.

```
[standalone@localhost:9999 /] /subsystem=resource-adapters/resource-adapter=eis.rar:add(archive=eis.rar, transaction-support=XATransaction) {"outcome" => "success"}
```

4. Configure the **server** resource adapter level <config-property>.

```
[standalone@localhost:9999 /] /subsystem=resource-adapters/resource-adapter=eis.rar/config-properties=server/:add(value=localhost) {"outcome" => "success"}
```

5. Configure the **port** resource adapter level <config-property>.

```
[standalone@localhost:9999 /] /subsystem=resource-adapters/resource-adapter=eis.rar/config-properties=port/:add(value=9000) {"outcome" => "success"}
```

6. Add a connection definition for a managed connection factory.

```
[standalone@localhost:9999 /] /subsystem=resource-adapters/resource-adapter=eis.rar/connection-definitions=cfName:add(class-name=com.acme.eis.ra.EISManagedConnectionFactory, jndi-name=java:/eis/AcmeConnectionFactory) {"outcome" => "success"}
```

7. Configure the **name** managed connection factory level <config-property>.

```
[standalone@localhost:9999 /] /subsystem=resource-adapters/resource-adapter=eis.rar/connection-definitions=cfName/config-properties=name/:add(value=Acme Inc) {"outcome" => "success"}
```

8. Add an admin object.

```
[standalone@localhost:9999 /] /subsystem=resource-adapters/resource-adapter=eis.rar/admin-objects=aoName:add(class-name=com.acme.eis.ra.EISAdminObjectImpl, jndi-name=java:/eis/AcmeAdminObject) {"outcome" => "success"}
```

9. Configure the **threshold** admin object property.

```
[standalone@localhost:9999 /] /subsystem=resource-adapters/resource-adapter=eis.rar/admin-objects=aoName/config-properties=threshold/:add(value=10) {"outcome" => "success"}
```

10. Activate the resource adapter.

```
[standalone@localhost:9999 /] /subsystem=resource-adapters/resource-adapter=eis.rar:activate {"outcome" => "success"}
```

11. View the newly configured and activated resource adapter.

-

```
[standalone@localhost:9999 /] /subsystem=resource-adapters/resource-
adapter=eis.rar:read-resource(recursive=true)
{
  "outcome" => "success",
  "result" => {
    "archive" => "eis.rar",
    "beanvalidationgroups" => undefined,
    "bootstrap-context" => undefined,
    "transaction-support" => "XATransaction",
    "admin-objects" => {"aoName" => {
      "class-name" => "com.acme.eis.ra.EISAdminObjectImpl",
      "enabled" => true,
      "jndi-name" => "java:/eis/AcmeAdminObject",
      "use-java-context" => true,
      "config-properties" => {"threshold" => {"value" => 10}}
    }},
    "config-properties" => {
      "server" => {"value" => "localhost"},
      "port" => {"value" => 9000}
    },
  },
  "connection-definitions" => {"cfName" => {
    "allocation-retry" => undefined,
    "allocation-retry-wait-millis" => undefined,
    "background-validation" => false,
    "background-validation-millis" => undefined,
    "blocking-timeout-wait-millis" => undefined,
    "class-name" => "com.acme.eis.ra.EISManagedConnectionFactory",
    "enabled" => true,
    "flush-strategy" => "FailingConnectionOnly",
    "idle-timeout-minutes" => undefined,
    "interleaving" => false,
    "jndi-name" => "java:/eis/AcmeConnectionFactory",
    "max-pool-size" => 20,
    "min-pool-size" => 0,
    "no-recovery" => undefined,
    "no-tx-separate-pool" => false,
    "pad-xid" => false,
    "pool-prefill" => false,
    "pool-use-strict-min" => false,
    "recovery-password" => undefined,
    "recovery-plugin-class-name" => undefined,
    "recovery-plugin-properties" => undefined,
    "recovery-security-domain" => undefined,
    "recovery-username" => undefined,
    "same-rm-override" => undefined,
    "security-application" => undefined,
    "security-domain" => undefined,
    "security-domain-and-application" => undefined,
    "use-ccm" => true,
    "use-fast-fail" => false,
    "use-java-context" => true,
    "use-try-lock" => undefined,
    "wrap-xa-resource" => true,
    "xa-resource-timeout" => undefined,
    "config-properties" => {"name" => {"value" => "Acme Inc"}}
  }
}
```

```

|    }}
|   }}
|  }}
| }
|}

```

Procedure 22.6. Configure a resource adapter using the Web-based Management Console

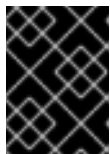
1. Login to the Management Console. See [Section 3.3.2, “Log in to the Management Console”](#) .
2. Click on the **Configuration** tab at the top of the screen. Expand the **Connectors** menu and select **Resource Adapters**.
 - a. In Domain mode, select a **Profile** from the drop-down at top left.

Click **Add**.
3. Enter the archive name and choose transaction type **XATransaction** from the **TX:** drop-down box. Then click **Save**.
4. Select the **Properties** tab. Click **Add**.
5. Enter **server** for the **Name** and the host name, for example **localhost**, for the **Value**. Then click **Save** to finish.
6. Click **Add** again. Enter **port** for the **Name** and the port number, for example **9000**, for the **Value**. Then click **Save** to finish.
7. The **server** and **port** properties now appear in the **Properties** panel. Click the **View** link under the **Option** column for the listed resource adapter to view the **Connection Definitions**.
8. Click **Add** above the **Available Connection Definitions** table to add a connection definition.
9. Enter the **JNDI Name** and the fully qualified class name of the **Connection Class**. Then click **Save** to finish.
10. Select the new Connection Definition, then select the **Properties** tab. Click **Add** to enter the **Key** and **Value** data for this connection definition. Click **Save** to finish.
11. The connection definition is complete, but disabled. Select the connection definition and click **Enable** to enable the connection definition.
12. A dialog asks **Really modify Connection Definition?** for the JNDI name. Click **Confirm**. The connection definition should now appear as **Enabled**.
13. Click the **Admin Objects** tab at the top of the page to create and configure admin objects. Then click **Add**.
14. Enter the **JNDI Name** and the fully qualified **Class Name** for the admin object. Then click **Save**.
15. Select the **Properties** tab, then click **Add** to add admin object properties.
16. Enter an admin object configuration property, for example **threshold**, in the **Name** field. Enter the configuration property value, for example **10**, in the **Value** field. Then click **Save** to save the property.
17. The admin object is complete, but disabled. Click **Enable** to enable the admin object.

18. A dialog asks **Really modify Admin Object?** for the JNDI name. Click **Confirm**. The admin object should now appear as **Enabled**.
19. You must reload the server configuration to complete the process. Click on the **Runtime** tab. Expand the **Server** menu. Select **Overview** in the left navigation panel.
 - a. Reload the servers
 - In Domain mode, hover the mouse over a server group. Select **Restart Group**.
 - In Standalone mode, a **Reload** button will be available. Click **Reload**.
20. A dialog asks **Do you want to reload the server configuration?** for the specified server. Click **Confirm**. The server configuration is up to date.

Procedure 22.7. Configure a resource adapter manually

1. Stop the JBoss EAP 6 server.



IMPORTANT

You must stop the server before editing the server configuration file for your change to be persisted on server restart.

2. Open the server configuration file for editing.
 - For a standalone server, this is the ***EAP_HOME/standalone/configuration/standalone.xml*** file.
 - For a managed domain, this is the ***EAP_HOME/domain/configuration/domain.xml*** file.
3. Find the **urn:jboss:domain:resource-adapters** subsystem in the configuration file.
4. If there are no resource adapters defined for this subsystem, first replace:

```
<subsystem xmlns="urn:jboss:domain:resource-adapters:1.1"/>
```

with this:

```
<subsystem xmlns="urn:jboss:domain:resource-adapters:1.1">
  <resource-adapters>
    <!-- <resource-adapter> configuration listed below -->
  </resource-adapters>
</subsystem>
```

5. Replace the **<!-- <resource-adapter> configuration listed below -->** with the XML definition for your resource adapter. The following is the XML representation of the resource adapter configuration created using the Management CLI and Web-based Management Console described above.

```

<resource-adapter id="NAME">
  <archive>
    eis.rar
  </archive>
  <transaction-support>XATransaction</transaction-support>
  <config-property name="server">
    localhost
  </config-property>
  <config-property name="port">
    9000
  </config-property>
  <connection-definitions>
    <connection-definition class-name="com.acme.eis.ra.EISManagedConnectionFactory"
      jndi-name="java:/eis/AcmeConnectionFactory"
      pool-name="java:/eis/AcmeConnectionFactory">
      <config-property name="name">
        Acme Inc
      </config-property>
    </connection-definition>
  </connection-definitions>
  <admin-objects>
    <admin-object class-name="com.acme.eis.ra.EISAdminObjectImpl"
      jndi-name="java:/eis/AcmeAdminObject"
      pool-name="java:/eis/AcmeAdminObject">
      <config-property name="threshold">
        10
      </config-property>
    </admin-object>
  </admin-objects>
</resource-adapter>

```

6. Start the server

Relaunch the JBoss EAP 6 server to start it running with the new configuration.

[Report a bug](#)

22.5. RESOURCE ADAPTER DESCRIPTOR REFERENCE

The following tables describe the resource adapter descriptor elements.

Table 22.7. Main elements

Element	Description
bean-validation-groups	Specifies bean validation group that should be used
bootstrap-context	Specifies the unique name of the bootstrap context that should be used
config-property	The config-property specifies resource adapter configuration properties.

Element	Description
transaction-support	Define the type of transaction supported by this resource adapter. Valid values are: NoTransaction , LocalTransaction , XATransaction
connection-definitions	Specifies the connection definitions
admin-objects	Specifies the administration objects

Table 22.8. Bean validation groups elements

Element	Description
bean-validation-group	Specifies the fully qualified class name for a bean validation group that should be used for validation

Table 22.9. Connection definition / admin object attributes

Attribute	Description
class-name	Specifies the fully qualified class name of a managed connection factory or admin object
jndi-name	Specifies the JNDI name
enabled	Should the object be activated
use-java-context	Specifies if a java:/ JNDI context should be used
pool-name	Specifies the pool name for the object
use-ccm	Enable the cached connection manager

Table 22.10. Connection definition elements

Element	Description
config-property	The config-property specifies managed connection factory configuration properties.
pool	Specifies pooling settings
xa-pool	Specifies XA pooling settings
security	Specifies security settings

Element	Description
timeout	Specifies time out settings
validation	Specifies validation settings
recovery	Specifies the XA recovery settings

Table 22.11. Pool elements

Element	Description
min-pool-size	The min-pool-size element indicates the minimum number of connections a pool should hold. These are not created until a Subject is known from a request for a connection. This default to 0
max-pool-size	The max-pool-size element indicates the maximum number of connections for a pool. No more than max-pool-size connections will be created in each sub-pool. This defaults to 20 .
prefill	Whether to attempt to prefill the connection pool. Default is false
use-strict-min	Specifies if the min-pool-size should be considered strictly. Default false
flush-strategy	Specifies how the pool should be flush in case of an error. Valid values are: FailingConnectionOnly (default), IdleConnections , EntirePool

Table 22.12. XA pool elements

Element	Description
min-pool-size	The min-pool-size element indicates the minimum number of connections a pool should hold. These are not created until a Subject is known from a request for a connection. This default to 0
max-pool-size	The max-pool-size element indicates the maximum number of connections for a pool. No more than max-pool-size connections will be created in each sub-pool. This defaults to 20 .
prefill	Whether to attempt to prefill the connection pool. Default is false

Element	Description
use-strict-min	Specifies if the min-pool-size should be considered strictly. Default false
flush-strategy	Specifies how the pool should be flush in case of an error. Valid values are: FailingConnectionOnly (default), IdleConnections , EntirePool
is-same-rm-override	The is-same-rm-override element allows one to unconditionally set whether the <code>javax.transaction.xa.XAResource.isSameRM(XAResource)</code> returns true or false
interleaving	An element to enable interleaving for XA connection factories
no-tx-separate-pools	Oracle does not like XA connections getting used both inside and outside a JTA transaction. To workaround the problem you can create separate sub-pools for the different contexts
pad-xid	Should the Xid be padded
wrap-xa-resource	Should the XAResource instances be wrapped in a <code>org.jboss.tm.XAResourceWrapper</code> instance

Table 22.13. Security elements

Element	Description
application	Indicates that application supplied parameters (such as from getConnection(user, pw)) are used to distinguish connections in the pool.
security-domain	Indicates Subject (from security domain) are used to distinguish connections in the pool. The content of the security-domain is the name of the JAAS security manager that will handle authentication. This name correlates to the JAAS login-config.xml descriptor <code>application-policy/name</code> attribute.
security-domain-and-application	Indicates that either application supplied parameters (such as from getConnection(user, pw)) or Subject (from security domain) are used to distinguish connections in the pool. The content of the security-domain is the name of the JAAS security manager that will handle authentication. This name correlates to the JAAS login-config.xml descriptor <code>application-policy/name</code> attribute.

Table 22.14. Time out elements

Element	Description
blocking-timeout-millis	The blocking-timeout-millis element indicates the maximum time in milliseconds to block while waiting for a connection before throwing an exception. Note that this blocks only while waiting for a permit for a connection, and will never throw an exception if creating a new connection takes an inordinately long time. The default is 30000 (30 seconds).
idle-timeout-minutes	The idle-timeout-minutes elements indicates the maximum time in minutes a connection may be idle before being closed. The actual maximum time depends also on the IdleRemover scan time, which is 1/2 the smallest idle-timeout-minutes of any pool.
allocation-retry	The allocation retry element indicates the number of times that allocating a connection should be tried before throwing an exception. The default is 0 .
allocation-retry-wait-millis	The allocation retry wait millis element indicates the time in milliseconds to wait between retrying to allocate a connection. The default is 5000 (5 seconds).
xa-resource-timeout	Passed to XAResource.setTransactionTimeout() . Default is zero which does not invoke the setter. Specified in seconds

Table 22.15. Validation elements

Element	Description
background-validation	An element to specify that connections should be validated on a background thread versus being validated prior to use
background-validation-minutes	The background-validation-minutes element specifies the amount of time, in minutes, that background validation will run.
use-fast-fail	Whether fail a connection allocation on the first connection if it is invalid (true) or keep trying until the pool is exhausted of all potential connections (false). Default is false

Table 22.16. Admin object elements

Element	Description
config-property	Specifies an administration object configuration property.

Table 22.17. Recovery elements

Element	Description
recover-credential	Specifies the user name / password pair or security domain that should be used for recovery.
recover-plugin	Specifies an implementation of the <code>org.jboss.jca.core.spi.recovery.RecoveryPlugin</code> class.

The deployment schemas are defined in **jboss-as-resource-adapters_1_0.xsd** and http://www.ironjacamar.org/doc/schema/ironjacamar_1_0.xsd for automatic activation.

[Report a bug](#)

22.6. VIEW DEFINED CONNECTION STATISTICS

You can read statistics for a defined connection from the **deployment=name.rar** subtree.

Statistics are defined at this level and not at the **/subsystem** level as this ensures they are accessible for any **rar** that is not defined in any configuration in the **standalone.xml** or **domain.xml** files.

For example:

Example 22.1. View Defined Connection Statistics

```
/deployment=example.rar/subsystem=resource-adapters/statistics=statistics/connection-definitions=java:\testMe:read-resource(include-runtime=true)
```



NOTE

Ensure you specify the ***include-runtime=true*** argument, as all statistics are runtime only information and the default is **false**.

[Report a bug](#)

22.7. RESOURCE ADAPTER STATISTICS

Core Statistics

The following table contains a list of the supported resource adapter core statistics:

Table 22.18. Core Statistics

Name	Description
ActiveCount	The number of active connections. Each of the connections is either in use by an application or available in the pool
AvailableCount	The number of available connections in the pool.

Name	Description
AverageBlockingTime	The average time spent blocking on obtaining an exclusive lock on the pool. The value is in milliseconds.
AverageCreationTime	The average time spent creating a connection. The value is in milliseconds.
CreatedCount	The number of connections created.
DestroyedCount	The number of connections destroyed.
InUseCount	The number of connections currently in use.
MaxCreationTime	The maximum time it took to create a connection. The value is in milliseconds.
MaxUsedCount	The maximum number of connections used.
MaxWaitCount	The maximum number of requests waiting for a connection at the same time.
MaxWaitTime	The maximum time spent waiting for an exclusive lock on the pool.
TimedOut	The number of timed out connections.
TotalBlockingTime	The total time spent waiting for an exclusive lock on the pool. The value is in milliseconds.
TotalCreationTime	The total time spent creating connections. The value is in milliseconds.
WaitCount	The number of requests that had to wait for a connection.

[Report a bug](#)

22.8. DEPLOY THE WEBSPHERE MQ RESOURCE ADAPTER

About WebSphere MQ

WebSphere MQ is IBM's Messaging Oriented Middleware (MOM) software that allows applications on distributed systems to communicate with each other. This is accomplished through the use of messages and message queues. WebSphere MQ is responsible for delivering messages to the message queues and for transferring data to other queue managers using message channels. For more information about WebSphere MQ, see [WebSphere MQ](#).

Summary

This topic covers the steps to deploy and configure the WebSphere MQ Resource Adapter in Red Hat JBoss Enterprise Application Platform 6. This can be accomplished by manually editing configuration files, using the Management CLI tool, or using the web-based Management Console.



NOTE

There is a known issue in WebSphere MQ Resource Adapter version 7.5.0.3 and earlier that causes periodic recovery to fail with an XA exception with messages similar to the following in the JBoss EAP server log:

```
WARN [com.arjuna.ats.jta] (Periodic Recovery) ARJUNA016027: Local
XARecoveryModule.xaRecovery got XA exception XAException.XAER_INVALID:
javax.transaction.xa.XAException: The method 'xa_recover' has failed with errorCode
'-5'.
```

A fix is available in version 7.5.0.4. A detailed description of this issue can be found here: <http://www-01.ibm.com/support/docview.wss?uid=swg11C97579>.

Be aware that WebSphere MQ 8.0 and above is not supported in EAP 6.x.

Prerequisites

Before you get started, you must verify the version of the WebSphere MQ resource adapter and understand some of the WebSphere MQ configuration properties.

- The WebSphere MQ resource adapter is supplied as a Resource Archive (RAR) file called **wmq.jmsra-VERSION.rar**. You must use version 7.5.0.x. See the note above for information about the required version.
- You must know the values of the following WebSphere MQ configuration properties. Refer to the WebSphere MQ product documentation for details about these properties.
 - MQ.QUEUE.MANAGER: The name of the WebSphere MQ queue manager
 - MQ.HOST.NAME: The host name used to connect to the WebSphere MQ queue manager
 - MQ.CHANNEL.NAME: The server channel used to connect to the WebSphere MQ queue manager
 - MQ.QUEUE.NAME: The name of the destination queue
 - MQ.TOPIC.NAME: The name of the destination topic
 - MQ.PORT: The port used to connect to the WebSphere MQ queue manager
 - MQ.CLIENT: The transport type
- For outbound connections, you must also be familiar with the following configuration property:
 - MQ.CONNECTIONFACTORY.NAME: The name of the connection factory instance that will provide the connection to the remote system



NOTE

The following are default configurations provided by IBM and are subject to change. Please refer to WebSphere MQ documentation for more information.

Procedure 22.8. Deploy the Resource Adapter Manually

1. Copy the **wmq.jmsra-VERSION.rar** file to the **EAP_HOME/standalone/deployments/** directory.
2. Add the resource adapter to the server configuration file.
 - a. Open the **EAP_HOME/standalone/configuration/standalone-full.xml** file in an editor.
 - b. Find the **urn:jboss:domain:resource-adapters** subsystem in the configuration file.
 - c. If there are no resource adapters defined for this subsystem, first replace:

```
<subsystem xmlns="urn:jboss:domain:resource-adapters:1.1"/>
```

with this:

```
<subsystem xmlns="urn:jboss:domain:resource-adapters:1.1">
  <resource-adapters>
    <!-- <resource-adapter> configuration listed below -->
  </resource-adapters>
</subsystem>
```

- d. The resource adapter configuration depends on whether you need transaction support and recovery. If you do not need transaction support, choose the first configuration step below. If you do need transaction support, choose the second configuration step. In both examples, the **config-property name** elements are case sensitive, and must be entered as seen in the example.
 - For non-transactional deployments, replace the **<!-- <resource-adapter> configuration listed below -->** with the following:

```
<resource-adapter>
  <archive>
    wmq.jmsra-VERSION.rar
  </archive>
  <transaction-support>NoTransaction</transaction-support>
  <connection-definitions>
    <connection-definition
      class-
name="com.ibm.mq.connector.outbound.ManagedConnectionFactoryImpl"
      jndi-name="java:jboss/MQ.CONNECTIONFACTORY.NAME"
      pool-name="MQ.CONNECTIONFACTORY.NAME">
      <config-property name="hostName">
        MQ.HOST.NAME
      </config-property>
      <config-property name="port">
        MQ.PORT
      </config-property>
      <config-property name="channel">
        MQ.CHANNEL.NAME
      </config-property>
      <config-property name="transportType">
        MQ.CLIENT
      </config-property>
      <config-property name="queueManager">
        MQ.QUEUE.MANAGER
      </config-property>
```

```

    <security>
      <security-domain>MySecurityDomain</security-domain>
    </security>
  </connection-definition>
</connection-definitions>
<admin-objects>
  <admin-object
    class-name="com.ibm.mq.connector.outbound.MQQueueProxy"
    jndi-name="java:jboss/MQ.QUEUE.NAME"
    pool-name="MQ.QUEUE.NAME">
    <config-property name="baseQueueName">
      MQ.QUEUE.NAME
    </config-property>
    <config-property name="baseQueueManagerName">
      MQ.QUEUE.MANAGER
    </config-property>
  </admin-object>
  <admin-object class-name="com.ibm.mq.connector.outbound.MQTopicProxy"
    jndi-name="java:jboss/MQ.TOPIC.NAME" pool-
name="MQ.TOPIC.NAME">
    <config-property name="baseTopicName">
      MQ.TOPIC.NAME
    </config-property>
    <config-property name="brokerPubQueueManager">
      MQ.QUEUE.MANAGER
    </config-property>
  </admin-object>
</admin-objects>
</resource-adapter>

```

Be sure to replace the *VERSION* with the actual version in the name of the RAR.

- For transactional deployments, replace the **<!-- <resource-adapter> configuration listed below -->** with the following:

```

<resource-adapter>
  <archive>
    wmq.jmsra-VERSION.rar
  </archive>
  <transaction-support>XATransaction</transaction-support>
  <connection-definitions>
    <connection-definition
      class-
name="com.ibm.mq.connector.outbound.ManagedConnectionFactoryImpl"
      jndi-name="java:jboss/MQ.CONNECTIONFACTORY.NAME"
      pool-name="MQ.CONNECTIONFACTORY.NAME">
      <config-property name="hostName">
        MQ.HOST.NAME
      </config-property>
      <config-property name="port">
        MQ.PORT
      </config-property>
      <config-property name="channel">
        MQ.CHANNEL.NAME
      </config-property>
      <config-property name="transportType">

```

```

    MQ.CLIENT
  </config-property>
  <config-property name="queueManager">
    MQ.QUEUE.MANAGER
  </config-property>
  <security>
    <security-domain>MySecurityDomain</security-domain>
  </security>
  <recovery>
    <recover-credential>
      <security-domain>RECOVERY_SECURITY_DOMAIN</security-domain>
    </recover-credential>
  </recovery>
</connection-definition>
</connection-definitions>
<admin-objects>
  <admin-object
    class-name="com.ibm.mq.connector.outbound.MQQueueProxy"
    jndi-name="java:jboss/MQ.QUEUE.NAME"
    pool-name="MQ.QUEUE.NAME">
    <config-property name="baseQueueName">
      MQ.QUEUE.NAME
    </config-property>
    <config-property name="baseQueueManagerName">
      MQ.QUEUE.MANAGER
    </config-property>
  </admin-object>
  <admin-object class-name="com.ibm.mq.connector.outbound.MQTopicProxy"
    jndi-name="java:jboss/MQ.TOPIC.NAME" pool-
name="MQ.TOPIC.NAME">
    <config-property name="baseTopicName">
      MQ.TOPIC.NAME
    </config-property>
    <config-property name="brokerPubQueueManager">
      MQ.QUEUE.MANAGER
    </config-property>
  </admin-object>
</admin-objects>
</resource-adapter>

```

Be sure to replace the *VERSION* with the actual version in the name of the RAR. You must also replace the *USER_NAME* and *PASSWORD* with the valid user name and password.



NOTE

To support transactions, the `<transaction-support>` element was set to **XATransaction**. To support XA recovery, the `<recovery>` element was added to the connection definition.

- e. If you want to change the default provider for the EJB3 messaging system in JBoss EAP 6 from HornetQ to WebSphere MQ, modify the **urn:jboss:domain:ejb3:1.2** subsystem as follows:

Replace:


```

<mdb>
  <resource-adapter-ref resource-adapter-name="hornetq-ra"/>
  <bean-instance-pool-ref pool-name="mdb-strict-max-pool"/>
</mdb>

```

with:

```

<mdb>
  <resource-adapter-ref resource-adapter-name="wmq.jmsra-VERSION.rar"/>
  <bean-instance-pool-ref pool-name="mdb-strict-max-pool"/>
</mdb>

```

Be sure to replace the *VERSION* with the actual version in the name of the RAR.

Procedure 22.9. Modify the MDB code to use the resource adapter

- Configure the `ActivationConfigProperty` and `ResourceAdapter` in the MDB code as shown below. All of the **propertyName** elements are case sensitive, and must be entered as seen below.

```

@MessageDriven(name="WebSphereMQMDB", activationConfig = {
    @ActivationConfigProperty(propertyName = "destinationType",propertyValue =
"javax.jms.Queue"),
    @ActivationConfigProperty(propertyName = "useJNDI", propertyValue = "false"),
    @ActivationConfigProperty(propertyName = "hostName", propertyValue =
"MQ.HOST.NAME"),
    @ActivationConfigProperty(propertyName = "port", propertyValue = "MQ.PORT"),
    @ActivationConfigProperty(propertyName = "channel", propertyValue =
"MQ.CHANNEL.NAME"),
    @ActivationConfigProperty(propertyName = "queueManager", propertyValue =
"MQ.QUEUE.MANAGER"),
    @ActivationConfigProperty(propertyName = "destination", propertyValue =
"MQ.QUEUE.NAME"),
    @ActivationConfigProperty(propertyName = "transportType", propertyValue =
"MQ.CLIENT")
})

@ResourceAdapter(value = "wmq.jmsra-VERSION.rar")
@TransactionAttribute(TransactionAttributeType.NOT_SUPPORTED)
public class WebSphereMQMDB implements MessageListener {
}

```

Be sure to replace the *VERSION* with the actual version in the name of the RAR.

[Report a bug](#)

22.9. INSTALL JBOSS ACTIVE MQ RESOURCE ADAPTER

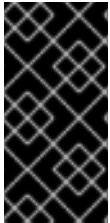
To install JBoss Active MQ (A-MQ) resource adapter to JBoss EAP 6 in order to make it work with JBoss Fuse A-MQ 6.1.0, follow the steps provided at: https://access.redhat.com/documentation/en-US/Red_Hat_JBoss_A-MQ/6.1/html/Integrating_with_JBoss_Enterprise_Application_Platform/DeployRar-InstallRar.html.

[Report a bug](#)

22.10. CONFIGURE A GENERIC JMS RESOURCE ADAPTER FOR USE WITH A THIRD-PARTY JMS PROVIDER

Summary

JBoss EAP 6 can be configured to work with third-party JMS providers, however not all JMS providers produce a JMS JCA resource adapter for integration with Java application platforms. This procedure covers the steps required to configure the generic JMS resource adapter included in JBoss EAP 6 to connect to a JMS provider. In this procedure, Tibco EMS 6.3 is used as an example JMS provider, however other JMS providers may need a different configuration.



IMPORTANT

Before using the generic JMS resource adapter, check with the JMS provider to see if they have their own resource adapter that can be used with JBoss EAP 6. The generic JMS JCA resource adapter should only be used when a JMS provider does not provide its own resource adapter.

Prerequisites

- JMS provider server is already configured and ready for use. Any binaries required for the provider's JMS implementation will be needed.
- You will also need to know the values of the following JMS provider properties to be able to lookup its JMS resources (connection factories, queues and topics).
 - `java.naming.factory.initial`
 - `java.naming.provider.url`
 - `java.naming.factory.url.pkgs`

In the example XML used in this procedure, these parameters are written as **PROVIDER_FACTORY_INITIAL**, **PROVIDER_URL**, and **PROVIDER_CONNECTION_FACTORY** respectively. Replace these placeholders with the JMS provider values for your environment.

Procedure 22.10. Configure a Generic JMS Resource Adapter for Use with a Third-party JMS Provider

1. Create a JBoss Module for the JMS provider

Create a JBoss module that contains all the libraries required to connect and communicate with the JMS provider. This module will be named **org.jboss.genericjms.provider**.

- a. Create the following directory structure:
`EAP_HOME/modules/system/layers/base/org/jboss/genericjms/provider/main`
- b. Copy the binaries required for the provider's JMS implementation to
`EAP_HOME/modules/system/layers/base/org/jboss/genericjms/provider/main`.



NOTE

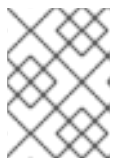
For Tibco EMS, the binaries required are **tibjms.jar** and **tibcrypt.jar** from the Tibco installation's **lib** directory.

- c. Create a **module.xml** file in **EAP_HOME/modules/system/layers/base/org/jboss/genericjms/provider/main** as below, listing the JAR files from the previous steps as resources:

```
<module xmlns="urn:jboss:module:1.1" name="org.jboss.genericjms.provider">
  <resources>
    <!-- all jars required by the JMS provider, in this case Tibco -->
    <resource-root path="tibjms.jar"/>
    <resource-root path="tibcrypt.jar"/>
  </resources>
  <dependencies>
    <module name="javax.api"/>
    <module name="javax.jms.api"/>
  </dependencies>
</module>
```

2. Configure a JNDI external context to the JMS provider

The JMS resources (connection factories and destinations) are looked up in the JMS provider. We will add an external context in the JBoss EAP 6 instance so that any *local* lookup for this resource will automatically look up the resource on the remote JMS provider.



NOTE

In this procedure, **EAP_HOME/standalone/configuration/standalone-full.xml** is used as the JBoss EAP 6 configuration file.

In **EAP_HOME/standalone/configuration/standalone-full.xml**, under **<subsystem xmlns="urn:jboss:domain:naming:1.4">**, add:

```
<bindings>
  <external-context name="java:global/remoteJMS/"
    module="org.jboss.genericjms.provider"
    class="javax.naming.InitialContext">
    <environment>
      <property name="java.naming.factory.initial"
        value="${PROVIDER_FACTORY_INITIAL}"/>
      <property name="java.naming.provider.url" value="${PROVIDER_URL}"/>
      <property name="java.naming.factory.url.pkgs" value="${PROVIDER_URL_PKGS}"/>
    </environment>
  </external-context>
</bindings>
```

These three properties' values must be replaced by the correct value to connect to the remote JMS provider. Take care when replacing the placeholder text to keep the **\${}** intact.

3. Enable Lookup by String

There are some JMS provider (such as Tibco EMS) that do not support the JNDI **lookup(Name)** method. In these cases, add the **org.jboss.as.naming.lookup.by.string** property with a value **true** to workaround this issue.

Example 22.2. Enable Lookup by String with Tibco EMS

A complete definition for an **external-context** to Tibco EMS would be as follows.

```

<bindings>
  <external-context name="java:global/remoteJMS/"
    module="org.jboss.genericjms.provider"
    class="javax.naming.InitialContext">
    <environment>
      <property name="java.naming.factory.initial"
value="com.tibco.tibjms.naming.TibjmsInitialContextFactory"/>
      <property name="java.naming.provider.url"
value="TIBCO_EMS_SERVER_HOST_NAME:PORT"/>
      <property name="java.naming.factory.url.pkgs" value="com.tibco.tibjms.naming"/>
      <property name="org.jboss.as.naming.lookup.by.string" value="true"/>
    </environment>
  </external-context>
</bindings>

```

With this external context, any JNDI lookup to a resource starting with **java:global/remoteJMS/** will be done on the remote JMS provider (after removing this prefix). As an example, if a Message-Driven Bean perform a JNDI lookup for **java:global/remoteJMS/Queue1**, the external context will connect to the remote JMS provider and perform a lookup for the **Queue1** resource.

4. Configure the Generic JMS Resource Adapter

In **EAP_HOME/standalone/configuration/standalone-full.xml**, add the generic resource adapter configuration to **<subsystem xmlns="urn:jboss:domain:resource-adapters:1.1">**.

Example 22.3. Tibco EMS Resource Adapter Configuration

A complete resource adapter definition for Tibco EMS would be as follows.

```

<resource-adapter id="org.jboss.genericjms">
  <module slot="main" id="org.jboss.genericjms"/>
  <transaction-support>NoTransaction</transaction-support>
  <connection-definitions>
    <connection-definition class-
name="org.jboss.resource.adapter.jms.JmsManagedConnectionFactory"
  jndi-name="java:/jms/XAQCF"
  pool-name="XAQCF">
      <config-property name="ConnectionFactory">
        XAQCF
      </config-property>
      <config-property name="JndiParameters">
        java.naming.factory.initial=com.tibco.tibjms.naming.TibjmsInitialContextFactory;java.naming.
rovider.url=TIBCO_EMS_SERVER_HOST_NAME:PORT
      </config-property>
      <security>
        <application/>
      </security>
    </connection-definition>
  </connection-definitions>
</resource-adapter>

```

5. Configure the default message-driven bean pool with the generic resource adapter. In `EAP_HOME/standalone/configuration/standalone-full.xml`, in `<subsystem xmlns="urn:jboss:domain:ejb3:1.5">`, update the `<mdb>` configuration with:

```
<mdb>
  <resource-adapter-ref resource-adapter-name="org.jboss.genericjms"/>
  <bean-instance-pool-ref pool-name="mdb-strict-max-pool"/>
</mdb>
```

Result

The generic JMS resource adapter is now configured and ready for use. When creating a new Message-driven Bean, use code similar below to use the resource adapter.

Example 22.4. Use the Generic Resource Adapter

```
@MessageDriven(name = "HelloWorldQueueMDB", activationConfig = {
  // The generic JMS resource adapter requires the JNDI bindings
  // for the actual remote connection factory and destination
  @ActivationConfigProperty(propertyName = "connectionFactory", propertyValue =
"java:global/remoteJMS/XAQCF"),
  @ActivationConfigProperty(propertyName = "destination", propertyValue =
"java:global/remoteJMS/Queue1"),
  @ActivationConfigProperty(propertyName = "destinationType", propertyValue =
"javax.jms.Queue"),
  @ActivationConfigProperty(propertyName = "acknowledgeMode", propertyValue = "Auto-
acknowledge" ) })
public class HelloWorldQueueMDB implements MessageListener {
  public void onMessage(Message message) {
    // called every time a message is received from the `Queue1` queue on the JMS provider.
  }
}
```



WARNING

When using the generic JMS resource adapter, ensure you set the session to be transacted, to avoid a potential **NullPointerException** (NPE) error. The NPE error occurs because the generic JMS resource adapter attempts processing of parameters, when the Java EE specification states that they are *not* to be processed.

```
connection.createSession(true, Session.SESSION_TRANSACTED);
```

Example 22.5. Use the Pooled Connection

You can also use the pooled connection factory from the resource adapter.

```
@Resource(lookup = "java:/jms/XAQCF")
private ConnectionFactory cf;
```

Example 22.6. Perform a Lookup

It is not possible to inject a resource from an external context directly but it is possible to inject an external context and then perform a lookup. For example, a lookup for a queue deployed in a Tibco EMS broker would be as follows.

```
@Resource(lookup = "java:global/remoteJMS")
private Context context;

...

Queue queue = (Queue) context.lookup("Queue1")
```

[Report a bug](#)

22.11. CONFIGURING THE HORNETQ JCA ADAPTER FOR REMOTE CONNECTIONS

Procedure 22.11. Configure an RA adapter to connect to two remote JBoss EAP instances

Before configuring the HornetQ RA to connect to remote EAP instance, we must create two connectors that point to remote JBoss EAP instances:

1. Create outbound socket bindings:

```
<outbound-socket-binding name="remote-jms-server-a">
  <remote-destination host="${remote.jms.server.one.bind.address:127.0.0.1}"
  port="${remote.jms.server.one.bind.port:5445}"/>
</outbound-socket-binding>
<outbound-socket-binding name="remote-jms-server-b">
  <remote-destination host="${remote.jms.server.two.bind.address:127.0.0.1}"
  port="${remote.jms.server.two.bind.port:5545}"/>
</outbound-socket-binding>
```

2. Create two netty connectors:

```
<netty-connector name="netty-remote-node-a" socket-binding="remote-jms-server-a"/>
<netty-connector name="netty-remote-node-b" socket-binding="remote-jms-server-a"/>
```

3. Create the RA configuration using the two netty connectors:

```
<pooled-connection-factory name="hornetq-remote-ra">
  <inbound-config>
    <use-jndi>true</use-jndi>

  <jndiparams>java.naming.factory.initial=org.jboss.naming.remote.client.InitialContextFactory;jav
a.naming.provider.url=remote://${remote.jms.server.one.bind.address}:4447,${remote.jms.serv
```

```

er.two.bind.address}:4547;java.naming.security.principal=${user.name};java.naming.security.cred
entials=${user.password}</jndi-params>
</inbound-config>
<transaction mode="xa"/>
<user>${user.name}</user>
<password>${user.password}</password>
<connectors>
  <connector-ref connector-name="netty-remote-node-a"/>
  <connector-ref connector-name="netty-remote-node-b"/>
</connectors>
<entries>
<entry name="java:/RemoteJmsXA"/>
</entries>
</pooled-connection-factory>

```

- Annotate the MDB to use the resource adapter using the @ResourceAdapter annotation:

```

@MessageDriven(name = "InQueueMDB", activationConfig = {
  @ActivationConfigProperty(propertyName = "destinationType", propertyValue =
    "javax.jms.Queue"),
  @ActivationConfigProperty(propertyName = "destination", propertyValue =
    "${hornetq.in.queue.short}"),
  @ActivationConfigProperty(propertyName = "acknowledgeMode", propertyValue = "Auto-
    acknowledge"),
  @ActivationConfigProperty(propertyName = "useJNDI",propertyValue = "true")
},mappedName = "${hornetq.inf.queue.long}")
@ResourceAdapter("hornetq-remote-ra")

```

- Run the following CLI commands to enable property substitution in order to use properties in deployment descriptors:

```

/subsystem=ee:write-attribute(name=jboss-descriptor-property-replacement,value=true)
/subsystem=ee:write-attribute(name=spec-descriptor-property-replacement,value=true)
/subsystem=ee:write-attribute(name=annotation-property-replacement,value=true)

```

- Create an external context to find the remote destinations in order to send message from the MDB:

```

<bindings>
  <external-context name="java:global/remote" module="org.jboss.remote-naming"
class="javax.naming.InitialContext">
    <environment>
      <property name="java.naming.factory.initial"
value="org.jboss.naming.remote.client.InitialContextFactory"/>
      <property name="java.naming.provider.url"
value="remote://${remote.jms.server.one.bind.address:ragga}:4447,${remote.jms.server.two.bi
nd.address:ragga}:4547"/>
      <property name="java.naming.security.principal" value="${user.name}"/>
      <property name="java.naming.security.credentials" value="${user.password}"/>
    </environment>
  </external-context>
</bindings>

```

- The MDB code would look like:

■

```

@MessageDriven(name = "InQueueMDB", activationConfig = {
  @ActivationConfigProperty(propertyName = "destinationType", propertyValue =
    "javax.jms.Queue"),
  @ActivationConfigProperty(propertyName = "destination", propertyValue =
    "${hornetq.in.queue.short}"),
  @ActivationConfigProperty(propertyName = "acknowledgeMode", propertyValue = "Auto-
    acknowledge"),
  @ActivationConfigProperty(propertyName = "useJNDI",propertyValue = "true"),
  @ActivationConfigProperty(propertyName = "hA", propertyValue = "true")
},mappedName = "${hornetq.inf.queue.full}")
@ResourceAdapter("hornetq-remote-ra")
public class InQueueMDB implements MessageListener {
  private static final Logger log = Logger.getLogger(InQueueMDB.class);
  @Resource(lookup = "java:global/remote")
  private InitialContext context;
  @Resource( name = "${hornetq.jms.connection}")
  private ConnectionFactory qcf;
  public void onMessage(Message message){
  try {
    if ( message instanceof TextMessage){
      Object obj = (Queue) context.lookup("/jms/queue/outQueue");
      qConnection = (QueueConnection) qcf.createConnection("quickuser","quick123+");
      qSession = qConnection.createQueueSession(true, Session.SESSION_TRANSACTED);
      qSender = qSession.createSender(outQueue);
      qSender.send(message);
    }
  }
}

```

8. You must remember that the hornetq RA module contains remoting-naming dependency for the MDB code given above to work:

```

<module xmlns="urn:jboss:module:1.1" name="org.hornetq.ra">
  <properties>
    <property name="jboss.api" value="private"/>
  </properties>
  <resources>
    <resource-root path="hornetq-ra-2.3.25.Final-redhat-1.jar"/>
    <!-- Insert resources here -->
  </resources>
  <dependencies>
    <module name="org.hornetq"/>
    <module name="org.jboss.as.transactions"/>
    <module name="org.jboss.as.messaging"/>
    <module name="org.jboss.jboss-transaction-spi"/>
    <module name="org.jboss.logging"/>
    <module name="org.jboss.remote-naming"/>
    <module name="javax.api"/>
    <module name="javax.jms.api" />
    <module name="org.jboss.jts"/>
    <module name="org.jboss.netty"/>
    <module name="org.jgroups"/>
    <module name="javax.resource.api"/>
  </dependencies>
</module>

```

9. You must also add org.hornetq to global modules so the JMS API is visible to the application:


```
<global-modules>  
  <module name="org.jboss.common-core" slot="main"/>  
  <module name="org.hornetq" slot="main"/>  
</global-modules>
```

[Report a bug](#)

CHAPTER 23. HIBERNATE SEARCH

23.1. GETTING STARTED WITH HIBERNATE SEARCH

23.1.1. About Hibernate Search

Hibernate Search provides full-text search capability to Hibernate applications. It is especially suited to search applications for which SQL-based solutions are not suited, including: full-text, fuzzy and geolocation searches. Hibernate Search uses Apache Lucene as its full-text search engine, but is designed to minimize the maintenance overhead. Once it is configured, indexing, clustering and data synchronization is maintained transparently, allowing you to focus on meeting your business requirements.

[Report a bug](#)

23.1.2. Overview

Hibernate Search consists of an indexing component as well as an index search component, both are backed by Apache Lucene. Each time an entity is inserted, updated or removed in/from the database, Hibernate Search keeps track of this event (through the Hibernate event system) and schedules an index update. All these updates are handled without you having to interact with the Apache Lucene APIs directly. Instead, interaction with the underlying Lucene indexes is handled via an **IndexManager**.

Once the index is created, you can search for entities and return lists of managed entities instead of dealing with the underlying Lucene infrastructure. The same persistence context is shared between Hibernate and Hibernate Search. The **FullTextSession** class is built on top of the Hibernate **Session** class so that the application code can use the unified **org.hibernate.Query** or **javax.persistence.Query** APIs exactly the same way an HQL, JPA-QL, or native query would do.



NOTE

It is recommended - for both your database and Hibernate Search - to execute your operations in a transaction, be it JDBC or JTA.



NOTE

Hibernate Search works perfectly fine in the Hibernate / EntityManager long conversation pattern, known as atomic conversation.

[Report a bug](#)

23.1.3. About the Index Manager

Each time an entity is inserted, updated or removed from the database, Hibernate Search keeps track of this event through the Hibernate event system and schedules an index update. Interaction with the underlying Lucene indexes is handled by an *IndexManager*, each of which is uniquely identified by name. By default there is a one-to-one relationship between *IndexManager* and Lucene index. The *IndexManager* abstracts the specific index configuration, including the selected *backend*, *reader strategy* and the chosen *DirectoryProvider*.

[Report a bug](#)

23.1.4. About the Directory Provider

Apache Lucene, which is part of the Hibernate Search infrastructure, has the concept of a Directory for storage of indexes. Hibernate Search handles the initialization and configuration of a Lucene Directory instance via a *Directory Provider*.

The **directory_provider** property specifies the directory provider to be used to store the indexes. The default filesystem directory provider is **filesystem**, which uses the local filesystem to store indexes.

[Report a bug](#)

23.1.5. About the Worker

Updates to Lucene indexes are handled by the Hibernate Search *Worker*, which receives all entity changes, queues them by context and applies them once a context ends. The most common context is the transaction, but may be dependent on the number of entity changes or some other application (life cycle) events.

For better efficiency, interactions are batched and generally applied once the context ends. Outside a transaction, the index update operation is executed right after the actual database operation. In the case of an ongoing transaction, the index update operation is scheduled for the transaction commit phase and discarded in case of transaction rollback. A worker may be configured with a specific batch size limit, after which indexing occurs regardless of the context.

For details of Worker configuration options see [Section 23.2.5, "Worker Configuration"](#).

There are two immediate benefits to this method of handling index updates:

- Performance: Lucene indexing works better when operation are executed in batch.
- ACIDity: The work executed has the same scoping as the one executed by the database transaction and is executed if and only if the transaction is committed. This is not ACID in the strict sense, but ACID behavior is rarely useful for full text search indexes since they can be rebuilt from the source at any time.

The two batch modes - no scope vs transactional - are the equivalent of autocommit versus transactional behavior. From a performance perspective, the *transactional* mode is recommended. The scoping choice is made transparently. Hibernate Search detects the presence of a transaction and adjust the scoping (see [Section 23.2.5, "Worker Configuration"](#)).

[Report a bug](#)

23.1.6. Back End Setup and Operations

23.1.6.1. Back End

Hibernate Search uses various back ends to process batches of work. The back end is not limited to the configuration option **default.worker.backend**. This property specifies a implementation of the **BackendQueueProcessor** interface which is a part of a back end configuration. Additional settings are required to set up a back end, for example the JMS back end.

[Report a bug](#)

23.1.6.2. Lucene

In the Lucene mode, all index updates for a node (JVM) are executed by the same node to the Lucene directories using the directory providers. Use this mode in a non-clustered environment or in clustered environments with a shared directory store.

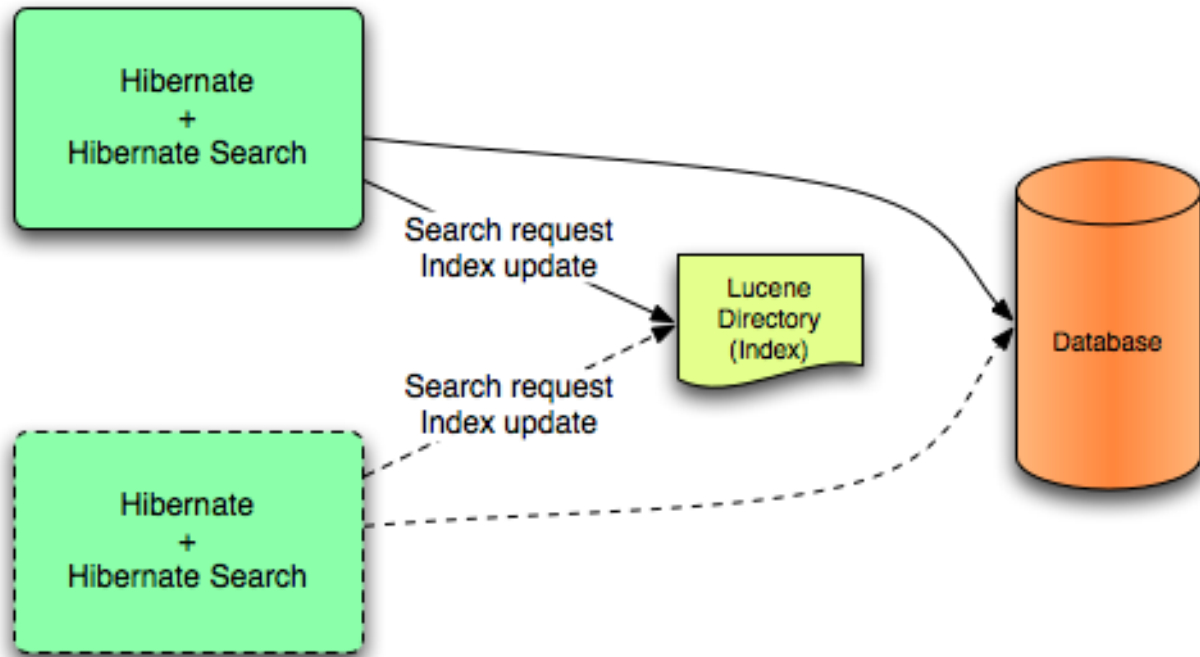


Figure 23.1. Lucene Back End Configuration

Lucene mode targets non-clustered or clustered applications where the **Directory** manages the locking strategy. The primary advantage of Lucene mode is simplicity and immediate visibility of changes in Lucene queries. The Near Real Time (NRT) back end is an alternate back end for non-clustered and non-shared index configurations.

[Report a bug](#)

23.1.6.3. JMS

Index updates for a node are sent to the JMS queue. A unique reader processes the queue and updates the master index. The master index is subsequently replicated regularly to slave copies to establish the master/slave pattern. The master is responsible for Lucene index updates. The slaves accept read and write operations but process read operations on local index copies. The master is the sole responsible for updating the Lucene index. Only the master applies the local changes in an update operation.

IndexReader is not updated, a new one is opened and provided. Each **IndexReader** is made of several **SegmentReader**s. The shared strategy reopens segments that have been modified or created after the last opening and shares the already loaded segments from the previous instance. This is the default strategy.

[Report a bug](#)

23.1.7.2. The Not-shared Strategy

Using the **not-shared** strategy, a Lucene **IndexReader** opens every time a query executes. Opening and starting up a **IndexReader** is an expensive operation. As a result, opening an **IndexReader** for each query execution is not an efficient strategy.

[Report a bug](#)

23.1.7.3. Custom Reader Strategies

You can write a custom reader strategy using an implementation of **org.hibernate.search.reader.ReaderProvider**. The implementation must be thread safe.

[Report a bug](#)

23.1.7.4. Reader Strategy Configuration

Change the strategy from the default (**shared**) to **not-shared** as follows:

```
hibernate.search.[default|<indexname>].reader.strategy = not-shared
```

Alternately, customize the reader strategy by replacing **my.corp.myapp.CustomReaderProvider** with the custom strategy implementation:

```
hibernate.search.[default|<indexname>].reader.strategy = my.corp.myapp.CustomReaderProvider
```

[Report a bug](#)

23.2. CONFIGURATION

23.2.1. Minimum Configuration

Hibernate Search has been designed to provide flexibility in its configuration and operation, with default values carefully chosen to suit the majority of use cases. At a minimum a **Directory Provider** must be configured, along with its properties. The default Directory Provider is **filesystem**, which uses the local filesystem for index storage. For details of available Directory Providers and their configuration, see [Section 23.2.3, "DirectoryProvider Configuration"](#).

If you are using Hibernate directly, settings such as the DirectoryProvider must be set in the configuration file, either **hibernate.properties** or **hibernate.cfg.xml**. If you are using Hibernate via JPA the configuration file is **persistence.xml**.

[Report a bug](#)

23.2.2. Configuring the IndexManager

Hibernate Search offers several implementations for this interface:

- **directory-based**: the default implementation which uses the Lucene **Directory** abstraction to manage index files.
- **near-real-time**: avoids flushing writes to disk at each commit. This index manager is also **Directory** based, but uses Lucene's near real-time (NRT) functionality.

To specify an `IndexManager` other than the default, specify the following property:

```
hibernate.search.[default|<indexname>].indexmanager = near-real-time
```

[Report a bug](#)

23.2.2.1. Directory-based

The **Directory-based** implementation is the default **IndexManager** implementation. It is highly configurable and allows separate configurations for the reader strategy, back ends, and directory providers.

[Report a bug](#)

23.2.2.2. Near Real Time

The **NRTIndexManager** is an extension of the default **IndexManager** and leverages the Lucene NRT (Near Real Time) feature for low latency index writes. However, it ignores configuration settings for alternative back ends other than **lucene** and acquires exclusive write locks on the **Directory**.

The **IndexWriter** does not flush every change to the disk to provide low latency. Queries can read the updated states from the unflushed index writer buffers. However, this means that if the **IndexWriter** is killed or the application crashes, updates can be lost so the indexes must be rebuilt.

The Near Real Time configuration is recommended for non-clustered websites with limited data due to the mentioned disadvantages and because a master node can be individually configured for improved performance as well.

[Report a bug](#)

23.2.2.3. Custom

Specify a fully qualified class name for the custom implementation to set up a customized **IndexManager**. Set up a no-argument constructor for the implementation as follows:

```
[default|<indexname>].indexmanager = my.corp.myapp.CustomIndexManager
```

The custom index manager implementation does not require the same components as the default implementations. For example, delegate to a remote indexing service which does not expose a **Directory** interface.

[Report a bug](#)

23.2.3. DirectoryProvider Configuration

A **DirectoryProvider** is the Hibernate Search abstraction around a Lucene **Directory** and handles the configuration and the initialization of the underlying Lucene resources. [Directory Providers and their](#)

[Properties](#) shows the list of the directory providers available in Hibernate Search together with their corresponding options.

Each indexed entity is associated with a Lucene index (except of the case where multiple entities share the same index). The name of the index is given by the **index** property of the **@Indexed** annotation. If the **index** property is not specified the fully qualified name of the indexed class will be used as name (recommended).

The `DirectoryProvider` and any additional options can be configured by using the prefix **hibernate.search.<indexname>**. The name **default** (**hibernate.search.default**) is reserved and can be used to define properties which apply to all indexes. [Example 23.2, "Configuring Directory Providers"](#) shows how **hibernate.search.default.directory_provider** is used to set the default directory provider to be the filesystem one. **hibernate.search.default.indexBase** sets then the default base directory for the indexes. As a result the index for the entity **Status** is created in **/usr/lucene/indexes/org.hibernate.example.Status**.

The index for the **Rule** entity, however, is using an in-memory directory, because the default directory provider for this entity is overridden by the property **hibernate.search.Rules.directory_provider**.

Finally the **Action** entity uses a custom directory provider **CustomDirectoryProvider** specified via **hibernate.search.Actions.directory_provider**.

Example 23.1. Specifying the Index Name

```
package org.hibernate.example;

@Indexed
public class Status { ... }

@Indexed(index="Rules")
public class Rule { ... }

@Indexed(index="Actions")
public class Action { ... }
```

Example 23.2. Configuring Directory Providers

```
hibernate.search.default.directory_provider = filesystem
hibernate.search.default.indexBase=/usr/lucene/indexes
hibernate.search.Rules.directory_provider = ram
hibernate.search.Actions.directory_provider = com.acme.hibernate.CustomDirectoryProvider
```



NOTE

Using the described configuration scheme you can easily define common rules like the directory provider and base directory, and override those defaults later on a per index basis.

Directory Providers and their Properties

ram

None

filesystem

File system based directory. The directory used will be `<indexBase>/<indexName >`

- **indexBase** : base directory
- **indexName**: override `@Indexed.index` (useful for sharded indexes)
- **locking_strategy** : optional, see [Section 23.2.7, “LockFactory Configuration”](#)
- **filesystem_access_type**: allows to determine the exact type of **FSDirectory** implementation used by this **DirectoryProvider**. Allowed values are **auto** (the default value, selects **NIOFSDirectory** on non Windows systems, **SimpleFSDirectory** on Windows), **simple (SimpleFSDirectory)**, **nio (NIOFSDirectory)**, **mmap (MMapDirectory)**. Refer to Javadocs of these **Directory** implementations before changing this setting. Even though **NIOFSDirectory** or **MMapDirectory** can bring substantial performance boosts they also have their issues.

filesystem-master

File system based directory. Like **filesystem**. It also copies the index to a source directory (aka copy directory) on a regular basis.

The recommended value for the refresh period is (at least) 50% higher than the time to copy the information (default 3600 seconds - 60 minutes).

Note that the copy is based on an incremental copy mechanism reducing the average copy time.

DirectoryProvider typically used on the master node in a JMS back end cluster.

The **buffer_size_on_copy** optimum depends on your operating system and available RAM; most people reported good results using values between 16 and 64MB.

- **indexBase**: base directory
- **indexName**: override `@Indexed.index` (useful for sharded indexes)
- **sourceBase**: source (copy) base directory.
- **source**: source directory suffix (default to `@Indexed.index`). The actual source directory name being `<sourceBase>/<source>`
- **refresh**: refresh period in seconds (the copy will take place every **refresh** seconds). If a copy is still in progress when the following **refresh** period elapses, the second copy operation will be skipped.
- **buffer_size_on_copy**: The amount of MegaBytes to move in a single low level copy instruction; defaults to 16MB.
- **locking_strategy** : optional, see [Section 23.2.7, “LockFactory Configuration”](#)
- **filesystem_access_type**: allows to determine the exact type of **FSDirectory** implementation used by this **DirectoryProvider**. Allowed values are **auto** (the default value, selects **NIOFSDirectory** on non Windows systems, **SimpleFSDirectory** on Windows), **simple (SimpleFSDirectory)**, **nio (NIOFSDirectory)**, **mmap (MMapDirectory)**. Refer to Javadocs

of these **Directory** implementations before changing this setting. Even though **NIOFSDirectory** or **MMapDirectory** can bring substantial performance boosts, there are also issues of which you need to be aware.

filesystem-slave

File system based directory. Like **filesystem**, but retrieves a master version (source) on a regular basis. To avoid locking and inconsistent search results, 2 local copies are kept.

The recommended value for the refresh period is (at least) 50% higher than the time to copy the information (default 3600 seconds - 60 minutes).

Note that the copy is based on an incremental copy mechanism reducing the average copy time. If a copy is still in progress when **refresh** period elapses, the second copy operation will be skipped.

DirectoryProvider typically used on slave nodes using a JMS back end.

The **buffer_size_on_copy** optimum depends on your operating system and available RAM; most people reported good results using values between 16 and 64MB.

- **indexBase**: Base directory
- **indexName**: override @Indexed.index (useful for sharded indexes)
- **sourceBase**: Source (copy) base directory.
- **source**: Source directory suffix (default to **@Indexed.index**). The actual source directory name being **<sourceBase>/<source>**
- **refresh**: refresh period in second (the copy will take place every refresh seconds).
- **buffer_size_on_copy**: The amount of MegaBytes to move in a single low level copy instruction; defaults to 16MB.
- **locking_strategy** : optional, see [Section 23.2.7, "LockFactory Configuration"](#)
- **retry_marker_lookup** : optional, default to 0. Defines how many times Hibernate Search checks for the marker files in the source directory before failing. Waiting 5 seconds between each try.
- **retry_initialize_period** : optional, set an integer value in seconds to enable the retry initialize feature: if the slave can't find the master index it will try again until it's found in background, without preventing the application to start: fullText queries performed before the index is initialized are not blocked but will return empty results. When not enabling the option or explicitly setting it to zero it will fail with an exception instead of scheduling a retry timer. To prevent the application from starting without an invalid index but still control an initialization timeout, see **retry_marker_lookup** instead.
- **filesystem_access_type**: allows to determine the exact type of **FSDirectory** implementation used by this **DirectoryProvider**. Allowed values are **auto** (the default value, selects **NIOFSDirectory** on non Windows systems, **SimpleFSDirectory** on Windows), **simple** (**SimpleFSDirectory**), **nio** (**NIOFSDirectory**), **mmap** (**MMapDirectory**). Refer to Javadocs of these **Directory** implementations before changing this setting. Even though **NIOFSDirectory** or **MMapDirectory** can bring substantial performance boosts you need also to be aware of the issues.

**NOTE**

If the built-in directory providers do not fit your needs, you can write your own directory provider by implementing the **org.hibernate.store.DirectoryProvider** interface. In this case, pass the fully qualified class name of your provider into the **directory_provider** property. You can pass any additional properties using the prefix **hibernate.search.<indexname>**.

[Report a bug](#)

23.2.4. Sharding Indexes

In some cases it can be useful to split (shard) the indexed data of a given entity into several Lucene indexes.

**WARNING**

Sharding should only be implemented if the advantages outweigh the disadvantages. Searching sharded indexes will typically be slower as all shards have to be opened for a single search.

Possible use cases for sharding are:

- A single index is so large that index update times are slowing the application down.
- A typical search will only hit a subset of the index, such as when data is naturally segmented by customer, region or application.

By default sharding is not enabled unless the number of shards is configured. To do this use the **hibernate.search.<indexName>.sharding_strategy.nbr_of_shards** property.

Example 23.3. Enabling Index Sharding

In this example 5 shards are enabled.

```
hibernate.search.<indexName>.sharding_strategy.nbr_of_shards = 5
```

Responsible for splitting the data into sub-indexes is the **IndexShardingStrategy**. The default sharding strategy splits the data according to the hash value of the ID string representation (generated by the **FieldBridge**). This ensures a fairly balanced sharding. You can replace the default strategy by implementing a custom **IndexShardingStrategy**. To use your custom strategy you have to set the **hibernate.search.<indexName>.sharding_strategy** property.

Example 23.4. Specifying a Custom Sharding Strategy

```
hibernate.search.<indexName>.sharding_strategy = my.shardingstrategy.Implementation
```

The **IndexShardingStrategy** property also allows for optimizing searches by selecting which shard to run the query against. By activating a filter a sharding strategy can select a subset of the shards used to answer a query (**IndexShardingStrategy.getIndexManagersForQuery**) and thus speed up the query execution.

Each shard has an independent **IndexManager** and so can be configured to use a different directory provider and back end configuration. The **IndexManager** index names for the **Animal** entity in [Example 23.5, “Sharding Configuration for Entity Animal”](#) are **Animal.0** to **Animal.4**. In other words, each shard has the name of its owning index followed by **.** (dot) and its index number.

Example 23.5. Sharding Configuration for Entity Animal

```
hibernate.search.default.indexBase = /usr/lucene/indexes
hibernate.search.Animal.sharding_strategy.nbr_of_shards = 5
hibernate.search.Animal.directory_provider = filesystem
hibernate.search.Animal.0.indexName = Animal00
hibernate.search.Animal.3.indexBase = /usr/lucene/sharded
hibernate.search.Animal.3.indexName = Animal03
```

In [Example 23.5, “Sharding Configuration for Entity Animal”](#), the configuration uses the default id string hashing strategy and shards the **Animal** index into 5 sub-indexes. All sub-indexes are filesystem instances and the directory where each sub-index is stored is as followed:

- for sub-index 0: **/usr/lucene/indexes/Animal00** (shared indexBase but overridden indexName)
- for sub-index 1: **/usr/lucene/indexes/Animal.1** (shared indexBase, default indexName)
- for sub-index 2: **/usr/lucene/indexes/Animal.2** (shared indexBase, default indexName)
- for sub-index 3: **/usr/lucene/sharded/Animal03** (overridden indexBase, overridden indexName)
- for sub-index 4: **/usr/lucene/indexes/Animal.4** (shared indexBase, default indexName)

When implementing a **IndexShardingStrategy** any field can be used to determine the sharding selection. Consider that to handle deletions, **purge** and **purgeAll** operations, the implementation might need to return one or more indexes without being able to read all the field values or the primary identifier. In that case the information is not enough to pick a single index, all indexes should be returned, so that the delete operation will be propagated to all indexes potentially containing the documents to be deleted.

[Report a bug](#)

23.2.5. Worker Configuration

It is possible to refine how Hibernate Search interacts with Lucene through the worker configuration. There exist several architectural components and possible extension points. Let's have a closer look.

First there is a **Worker**. An implementation of the **Worker** interface is responsible for receiving all entity changes, queuing them by context and applying them once a context ends. The most intuitive context, especially in connection with ORM, is the transaction. For this reason Hibernate Search will per default use the **TransactionalWorker** to scope all changes per transaction. One can, however, imagine a scenario where the context depends for example on the number of entity changes or some other application (lifecycle) events. For this reason the **Worker** implementation is configurable as shown in [Table 23.1, “Scope configuration”](#).

Table 23.1. Scope configuration

Property	Description
hibernate.search.worker.scope	The fully qualified class name of the Worker implementation to use. If this property is not set, empty or transaction the default TransactionalWorker is used.
hibernate.search.worker.*	All configuration properties prefixed with hibernate.search.worker are passed to the Worker during initialization. This allows adding custom, worker specific parameters.
hibernate.search.worker.batch_size	Defines the maximum number of indexing operation batched per context. Once the limit is reached indexing will be triggered even though the context has not ended yet. This property only works if the Worker implementation delegates the queued work to BatchedQueueingProcessor (which is what the TransactionalWorker does)

Once a context ends it is time to prepare and apply the index changes. This can be done synchronously or asynchronously from within a new thread. Synchronous updates have the advantage that the index is at all times in sync with the databases. Asynchronous updates, on the other hand, can help to minimize the user response time. The drawback is potential discrepancies between database and index states. Lets look at the configuration options shown in [Table 23.2, "Execution configuration"](#).

**NOTE**

The following options can be different on each index; in fact they need the indexName prefix or use **default** to set the default value for all indexes.

Table 23.2. Execution configuration

Property	Description
hibernate.search.<indexName>.worker.execution	sync : synchronous execution (default) async : asynchronous execution
hibernate.search.<indexName>.worker.thread_pool.size	The backend can apply updates from the same transaction context (or batch) in parallel, using a threadpool. The default value is 1. You can experiment with larger values if you have many operations per transaction.
hibernate.search.<indexName>.worker.buffer_queue.max	Defines the maximal number of work queue if the thread poll is starved. Useful only for asynchronous execution. Default to infinite. If the limit is reached, the work is done by the main thread.

So far all work is done within the same Virtual Machine (VM), no matter which execution mode. The total amount of work has not changed for the single VM. Luckily there is a better approach, namely delegation. It is possible to send the indexing work to a different server by configuring `hibernate.search.default.worker.backend` - see [Table 23.3, "Backend configuration"](#). Again this option can be configured differently for each index.

Table 23.3. Backend configuration

Property	Description
<code>hibernate.search.<indexName>.worker.backend</code>	<p>lucene: The default backend which runs index updates in the same VM. Also used when the property is undefined or empty.</p> <p>jms: JMS backend. Index updates are send to a JMS queue to be processed by an indexing master. See Table 23.4, "JMS backend configuration" for additional configuration options and Section 23.2.5.1, "JMS Master/Slave Back End" for a more detailed description of this setup.</p> <p>blackhole: Mainly a test/developer setting which ignores all indexing work</p> <p>You can also specify the fully qualified name of a class implementing BackendQueueProcessor. This way you can implement your own communication layer. The implementation is responsible for returning a Runnable instance which on execution will process the index work.</p>

Table 23.4. JMS backend configuration

Property	Description
<code>hibernate.search.<indexName>.worker.jndi.*</code>	Defines the JNDI properties to initiate the InitialContext (if needed). JNDI is only used by the JMS back end.
<code>hibernate.search.<indexName>.worker.jms.connection_factory</code>	Mandatory for the JMS back end. Defines the JNDI name to lookup the JMS connection factory from (ConnectionFactory by default in Red Hat JBoss Enterprise Application Platform)
<code>hibernate.search.<indexName>.worker.jms.queue</code>	Mandatory for the JMS back end. Defines the JNDI name to lookup the JMS queue from. The queue will be used to post work messages.



WARNING

As you probably noticed, some of the shown properties are correlated which means that not all combinations of property values make sense. In fact you can end up with a non-functional configuration. This is especially true for the case that you provide your own implementations of some of the shown interfaces. Make sure to study the existing code before you write your own **Worker** or **BackendQueueProcessor** implementation.

[Report a bug](#)

23.2.5.1. JMS Master/Slave Back End

This section describes in greater detail how to configure the Master/Slave Hibernate Search architecture.

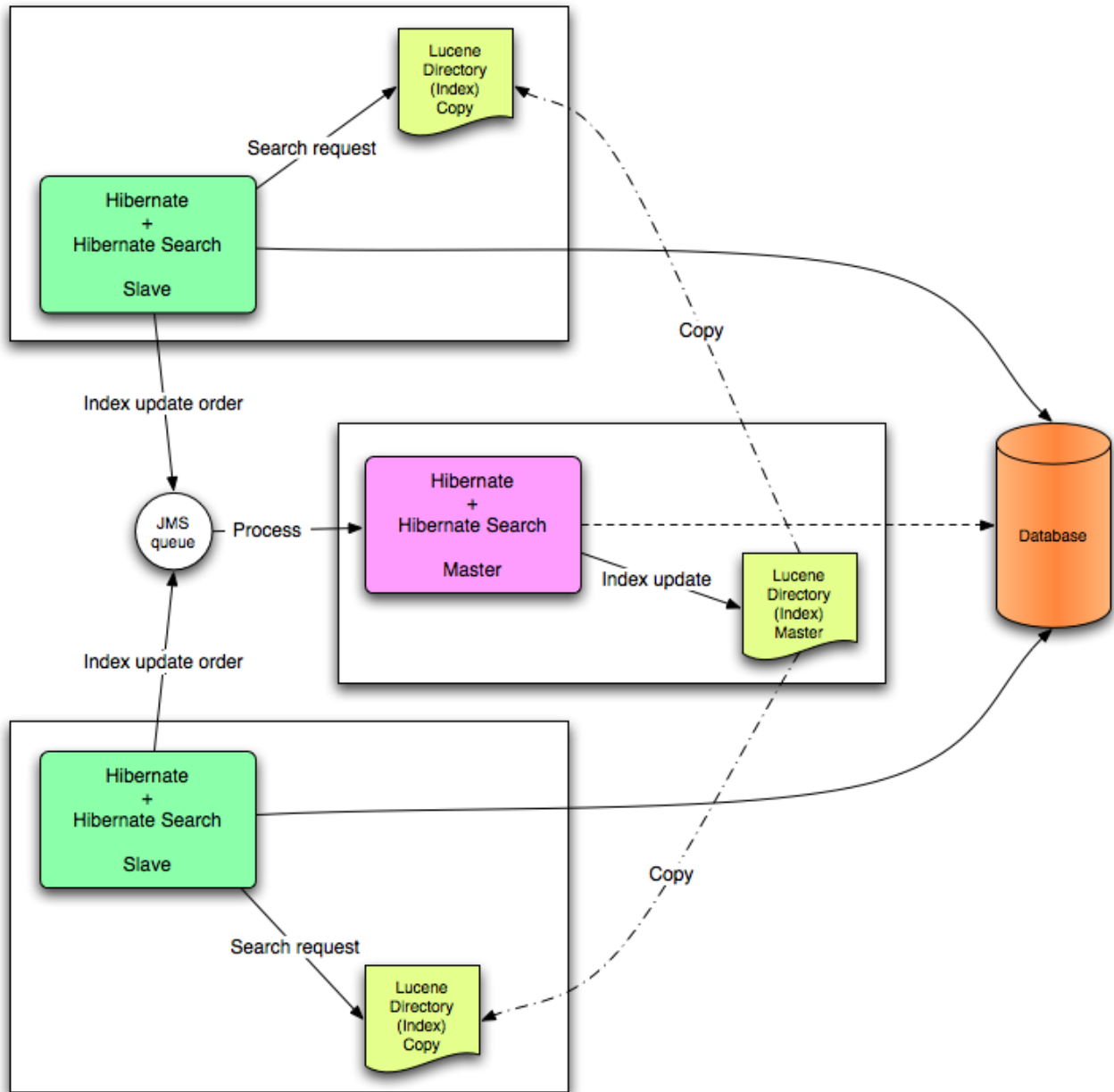


Figure 23.3. JMS Backend Configuration

[Report a bug](#)

23.2.5.2. Slave Nodes

Every index update operation is sent to a JMS queue. Index querying operations are executed on a local index copy.

Example 23.6. JMS Slave configuration

```
### slave configuration

## DirectoryProvider
# (remote) master location
hibernate.search.default.sourceBase = /mnt/mastervolume/lucenedirs/mastercopy

# local copy location
hibernate.search.default.indexBase = /Users/prod/lucenedirs
```



```
# refresh every half hour
hibernate.search.default.refresh = 1800

# appropriate directory provider
hibernate.search.default.directory_provider = filesystem-slave

## Backend configuration
hibernate.search.default.worker.backend = jms
hibernate.search.default.worker.jms.connection_factory = /ConnectionFactory
hibernate.search.default.worker.jms.queue = queue/hibernatesearch
#optional jndi configuration (check your JMS provider for more information)

## Optional asynchronous execution strategy
# hibernate.search.default.worker.execution = async
# hibernate.search.default.worker.thread_pool.size = 2
# hibernate.search.default.worker.buffer_queue.max = 50
```



NOTE

A file system local copy is recommended for faster search results.

[Report a bug](#)

23.2.5.3. Master Node

Every index update operation is taken from a JMS queue and executed. The master index is copied on a regular basis.

Example 23.7. JMS Master configuration

```
### master configuration

## DirectoryProvider
# (remote) master location where information is copied to
hibernate.search.default.sourceBase = /mnt/mastervolume/lucenedirs/mastercopy

# local master location
hibernate.search.default.indexBase = /Users/prod/lucenedirs

# refresh every half hour
hibernate.search.default.refresh = 1800

# appropriate directory provider
hibernate.search.default.directory_provider = filesystem-master

## Backend configuration
#Backend is the default lucene one
```

In addition to the Hibernate Search framework configuration, a Message Driven Bean has to be written and set up to process the index works queue through JMS.

Example 23.8. Message Driven Bean processing the indexing queue

```

@MessageDriven(activationConfig = {
    @ActivationConfigProperty(propertyName="destinationType",
        propertyValue="javax.jms.Queue"),
    @ActivationConfigProperty(propertyName="destination",
        propertyValue="queue/hibernatesearch"),
    @ActivationConfigProperty(propertyName="DLQMaxResent", propertyValue="1")
})
public class MDBSearchController extends AbstractJMSHibernateSearchController
    implements MessageListener {
    @PersistenceContext EntityManager em;

    //method retrieving the appropriate session
    protected Session getSession() {
        return (Session) em.getDelegate();
    }

    //potentially close the session opened in #getSession(), not needed here
    protected void cleanSessionIfNeeded(Session session)
    {
    }
}

```

This example inherits from the abstract JMS controller class available in the Hibernate Search source code and implements a JavaEE MDB. This implementation is given as an example and can be adjusted to make use of non Java EE Message Driven Beans. For more information about the **getSession()** and **cleanSessionIfNeeded()**, see **AbstractJMSHibernateSearchController**'s javadoc.

[Report a bug](#)

23.2.6. Tuning Lucene Indexing

23.2.6.1. Tuning Lucene Indexing Performance

Hibernate Search is used to tune the Lucene indexing performance by specifying a set of parameters which are passed through to underlying Lucene **IndexWriter** such as **mergeFactor**, **maxMergeDocs**, and **maxBufferedDocs**. Specify these parameters either as default values applying for all indexes, on a per index basis, or even per shard.

There are several low level **IndexWriter** settings which can be tuned for different use cases. These parameters are grouped by the **indexwriter** keyword:

```
hibernate.search.[default|<indexname>].indexwriter.<parameter_name>
```

If no value is set for an **indexwriter** value in a specific shard configuration, Hibernate Search checks the index section, then at the default section.

The configuration in the following table will result in these settings applied on the second shard of the **Animal** index:

- **max_merge_docs** = 10
- **merge_factor** = 20

- **ram_buffer_size** = 64MB
- **term_index_interval** = Lucene default

All other values will use the defaults defined in Lucene.

The default for all values is to leave them at Lucene's own default. The values listed in [Table 23.5, "List of indexing performance and behavior properties"](#) depend for this reason on the version of Lucene you are using. The values shown are relative to version **2.4**. For more information about Lucene indexing performance, see the Lucene documentation.



NOTE

Previous versions of Hibernate Search had the notion of **batch** and **transaction** properties. This is no longer the case as the backend will always perform work using the same settings.

Table 23.5. List of indexing performance and behavior properties

Property	Description	Default Value
hibernate.search.[default <indexname>].exclusive_index_use	Set to true when no other process will need to write to the same index. This enables Hibernate Search to work in exclusive mode on the index and improve performance when writing changes to the index.	true (improved performance, releases locks only at shutdown)
hibernate.search.[default <indexname>].max_queue_length	Each index has a separate "pipeline" which contains the updates to be applied to the index. When this queue is full adding more operations to the queue becomes a blocking operation. Configuring this setting doesn't make much sense unless the worker.execution is configured as async .	1000
hibernate.search.[default <indexname>].indexwriter.max_buffered_delete_terms	Determines the minimal number of delete terms required before the buffered in-memory delete terms are applied and flushed. If there are documents buffered in memory at the time, they are merged and a new segment is created.	Disabled (flushes by RAM usage)
hibernate.search.[default <indexname>].indexwriter.max_buffered_docs	Controls the amount of documents buffered in memory during indexing. The bigger the more RAM is consumed.	Disabled (flushes by RAM usage)
hibernate.search.[default <indexname>].indexwriter.max_merge_docs	Defines the largest number of documents allowed in a segment. Smaller values perform better on frequently changing indexes, larger values provide better search performance if the index does not change often.	Unlimited (Integer.MAX_VALUE)

Property	Description	Default Value
hibernate.search.[default <indexname>].indexwriter.merge_factor	<p>Controls segment merge frequency and size.</p> <p>Determines how often segment indexes are merged when insertion occurs. With smaller values, less RAM is used while indexing, and searches on unoptimized indexes are faster, but indexing speed is slower. With larger values, more RAM is used during indexing, and while searches on unoptimized indexes are slower, indexing is faster. Thus larger values (> 10) are best for batch index creation, and smaller values (< 10) for indexes that are interactively maintained. The value must not be lower than 2.</p>	10
hibernate.search.[default <indexname>].indexwriter.merge_min_size	<p>Controls segment merge frequency and size.</p> <p>Segments smaller than this size (in MB) are always considered for the next segment merge operation.</p> <p>Setting this too large might result in expensive merge operations, even though they are less frequent.</p> <p>See also org.apache.lucene.index.LogDocMergePolicy.minMergeSize.</p>	0 MB (actually ~1K)
hibernate.search.[default <indexname>].indexwriter.merge_max_size	<p>Controls segment merge frequency and size.</p> <p>Segments larger than this size (in MB) are never merged in bigger segments.</p> <p>This helps reduce memory requirements and avoids some merging operations at the cost of optimal search speed. When optimizing an index this value is ignored.</p> <p>See also org.apache.lucene.index.LogDocMergePolicy.maxMergeSize.</p>	Unlimited
hibernate.search.[default <indexname>].indexwriter.merge_max_optimize_size	<p>Controls segment merge frequency and size.</p> <p>Segments larger than this size (in MB) are not merged in bigger segments even when optimizing the index (see merge_max_size setting as well).</p> <p>Applied to org.apache.lucene.index.LogDocMergePolicy.maxMergeSizeForOptimize.</p>	Unlimited

Property	Description	Default Value
hibernate.search.[default <indexname>].indexwriter.merge_calibrate_by_deletes	<p>Controls segment merge frequency and size.</p> <p>Set to false to not consider deleted documents when estimating the merge policy.</p> <p>Applied to org.apache.lucene.index.LogMergePolicy.calibrateSizeByDeletes.</p>	true
hibernate.search.[default <indexname>].indexwriter.ram_buffer_size	<p>Controls the amount of RAM in MB dedicated to document buffers. When used together max_buffered_docs a flush occurs for whichever event happens first.</p> <p>Generally for faster indexing performance it's best to flush by RAM usage instead of document count and use as large a RAM buffer as you can.</p>	16 MB
hibernate.search.[default <indexname>].indexwriter.term_index_interval	<p>Expert: Set the interval between indexed terms.</p> <p>Large values cause less memory to be used by IndexReader, but slow random-access to terms. Small values cause more memory to be used by an IndexReader, and speed random-access to terms. See Lucene documentation for more details.</p>	128
hibernate.search.[default <indexname>].indexwriter.use_compound_file	<p>The advantage of using the compound file format is that less file descriptors are used. The disadvantage is that indexing takes more time and temporary disk space. You can set this parameter to false in an attempt to improve the indexing time, but you could run out of file descriptors if mergeFactor is also large.</p> <p>Boolean parameter, use "true" or "false". The default value for this option is true.</p>	true

Property	Description	Default Value
hibernate.search.enable_dirty_check	<p>Not all entity changes require a Lucene index update. If all of the updated entity properties (dirty properties) are not indexed, Hibernate Search skips the re-indexing process.</p> <p>Disable this option if you use custom FieldBridges which need to be invoked at each update event (even though the property for which the field bridge is configured has not changed).</p> <p>This optimization will not be applied on classes using a @ClassBridge or a @DynamicBoost.</p> <p>Boolean parameter, use "true" or "false". The default value for this option is true.</p>	true



WARNING

The **blackhole** backend is not meant to be used in production, only as a tool to identify indexing bottlenecks.

[Report a bug](#)

23.2.6.2. The Lucene IndexWriter

There are several low level **IndexWriter** settings which can be tuned for different use cases. These parameters are grouped by the **indexwriter** keyword:

```
default.<indexname>.indexwriter.<parameter_name>
```

If no value is set for **indexwriter** in a shard configuration, Hibernate Search looks at the index section and then at the default section.

[Report a bug](#)

23.2.6.3. Performance Option Configuration

The following configuration will result in these settings being applied on the second shard of the **Animal** index:

Example 23.9. Example performance option configuration

```
default.Animals.2.indexwriter.max_merge_docs = 10
default.Animals.2.indexwriter.merge_factor = 20
default.Animals.2.indexwriter.term_index_interval = default
```

```
default.indexwriter.max_merge_docs = 100
default.indexwriter.ram_buffer_size = 64
```

- **max_merge_docs** = 10
- **merge_factor** = 20
- **ram_buffer_size** = 64MB
- **term_index_interval** = Lucene default

All other values will use the defaults defined in Lucene.

The Lucene default values are the default setting for Hibernate Search. Therefore, the values listed in the following table depend on the version of Lucene being used. The values shown are relative to version **2.4**. For more information about Lucene indexing performance, see the Lucene documentation.



NOTE

The back end will always perform work using the same settings.

Table 23.6. List of indexing performance and behavior properties

Property	Description	Default Value
default.<indexname>.exclusive_index_use	Set to true when no other process will need to write to the same index. This enables Hibernate Search to work in exclusive mode on the index and improve performance when writing changes to the index.	true (improved performance, releases locks only at shutdown)
default.<indexname>.max_queue_length	Each index has a separate "pipeline" which contains the updates to be applied to the index. When this queue is full adding more operations to the queue becomes a blocking operation. Configuring this setting doesn't make much sense unless the worker.execution is configured as async .	1000
default.<indexname>.indexwriter.max_buffered_delete_terms	Determines the minimal number of delete terms required before the buffered in-memory delete terms are applied and flushed. If there are documents buffered in memory at the time, they are merged and a new segment is created.	Disabled (flushes by RAM usage)
default.<indexname>.indexwriter.max_buffered_docs	Controls the amount of documents buffered in memory during indexing. The bigger the more RAM is consumed.	Disabled (flushes by RAM usage)

Property	Description	Default Value
default. <indexname>.indexwriter.max_merge_docs	Defines the largest number of documents allowed in a segment. Smaller values perform better on frequently changing indexes, larger values provide better search performance if the index does not change often.	Unlimited (Integer.MAX_VALUE)
default. <indexname>.indexwriter.merge_factor	Controls segment merge frequency and size. Determines how often segment indexes are merged when insertion occurs. With smaller values, less RAM is used while indexing, and searches on unoptimized indexes are faster, but indexing speed is slower. With larger values, more RAM is used during indexing, and while searches on unoptimized indexes are slower, indexing is faster. Thus larger values (> 10) are best for batch index creation, and smaller values (< 10) for indexes that are interactively maintained. The value must not be lower than 2.	10
default. <indexname>.indexwriter.merge_min_size	Controls segment merge frequency and size. Segments smaller than this size (in MB) are always considered for the next segment merge operation. Setting this too large might result in expensive merge operations, even though they are less frequent. See also org.apache.lucene.index.LogDocMergePolicy.minMergeSize.	0 MB (actually ~1K)
default. <indexname>.indexwriter.merge_max_size	Controls segment merge frequency and size. Segments larger than this size (in MB) are never merged in bigger segments. This helps reduce memory requirements and avoids some merging operations at the cost of optimal search speed. When optimizing an index this value is ignored. See also org.apache.lucene.index.LogDocMergePolicy.maxMergeSize.	Unlimited

Property	Description	Default Value
default. <indexname>.indexwriter.merge_max_optimize_size	<p>Controls segment merge frequency and size.</p> <p>Segments larger than this size (in MB) are not merged in bigger segments even when optimizing the index (see merge_max_size setting as well).</p> <p>Applied to org.apache.lucene.index.LogDocMergePolicy.maxMergeSizeForOptimize.</p>	Unlimited
default. <indexname>.indexwriter.merge_calibrate_by_deletes	<p>Controls segment merge frequency and size.</p> <p>Set to false to not consider deleted documents when estimating the merge policy.</p> <p>Applied to org.apache.lucene.index.LogMergePolicy.calibrateSizeByDeletes.</p>	true
default. <indexname>.indexwriter.ram_buffer_size	<p>Controls the amount of RAM in MB dedicated to document buffers. When used together max_buffered_docs a flush occurs for whichever event happens first.</p> <p>Generally for faster indexing performance it's best to flush by RAM usage instead of document count and use as large a RAM buffer as you can.</p>	16 MB
default. <indexname>.indexwriter.term_index_interval	<p>Expert: Set the interval between indexed terms.</p> <p>Large values cause less memory to be used by IndexReader, but slow random-access to terms. Small values cause more memory to be used by an IndexReader, and speed random-access to terms. See Lucene documentation for more details.</p>	128
default. <indexname>.indexwriter.use_compound_file	<p>The advantage of using the compound file format is that less file descriptors are used. The disadvantage is that indexing takes more time and temporary disk space. You can set this parameter to false in an attempt to improve the indexing time, but you could run out of file descriptors if mergeFactor is also large.</p> <p>Boolean parameter, use "true" or "false". The default value for this option is true.</p>	true

Property	Description	Default Value
default.enable_dirty_check	<p>Not all entity changes require a Lucene index update. If all of the updated entity properties (dirty properties) are not indexed, Hibernate Search skips the re-indexing process.</p> <p>Disable this option if you use custom FieldBridges which need to be invoked at each update event (even though the property for which the field bridge is configured has not changed).</p> <p>This optimization will not be applied on classes using a @ClassBridge or a @DynamicBoost.</p> <p>Boolean parameter, use "true" or "false". The default value for this option is true.</p>	true

[Report a bug](#)

23.2.6.4. Tuning the Indexing Speed

When the architecture permits it, keep **default.exclusive_index_use=true** for improved index writing efficiency.

When tuning indexing speed the recommended approach is to focus first on optimizing the object loading, and then use the timings you achieve as a baseline to tune the indexing process. Set the **blackhole** as worker back end and start your indexing routines. This back end does not disable Hibernate Search: it generates the required change sets to the index, but discards them instead of flushing them to the index. In contrast to setting the **hibernate.search.indexing_strategy** to **manual**, using **blackhole** will possibly load more data from the database because associated entities are re-indexed as well.

```
hibernate.search.[default|<indexname>].worker.backend blackhole
```



WARNING

The **blackhole** back end is not to be used in production, only as a diagnostic tool to identify indexing bottlenecks.

[Report a bug](#)

23.2.6.5. Control Segment Size

The following options configure the maximum size of segments created:

- **merge_max_size**

- *merge_max_optimize_size*
- *merge_calibrate_by_deletes*

Example 23.10. Control Segment Size

```
//to be fairly confident no files grow above 15MB, use:
hibernate.search.default.indexwriter.ram_buffer_size = 10
hibernate.search.default.indexwriter.merge_max_optimize_size = 7
hibernate.search.default.indexwriter.merge_max_size = 7
```

Set the *max_size* for merge operations to less than half of the hard limit segment size, as merging segments combines two segments into one larger segment.

A new segment may initially be a larger size than expected, however a segment is never created significantly larger than the *ram_buffer_size*. This threshold is checked as an estimate.

[Report a bug](#)

23.2.7. LockFactory Configuration

The Lucene Directory can be configured with a custom locking strategy via **LockingFactory** for each index managed by Hibernate Search.

Some locking strategies require a filesystem level lock, and may be used on RAM-based indexes. When using this strategy the **IndexBase** configuration option must be specified to point to a filesystem location in which to store the lock marker files.

To select a locking factory, set the **hibernate.search.<index>.locking_strategy** option to one the following options:

- *simple*
- *native*
- *single*
- *none*

Table 23.7. List of available LockFactory implementations

name	Class	Description
simple	org.apache.lucene.store.SimpleFSLockFactory	Safe implementation based on Java's File API, it marks the usage of the index by creating a marker file. If for some reason you had to kill your application, you will need to remove this file before restarting it.

name	Class	Description
native	org.apache.lucene.store.NativeFSLockFactory	As does simple this also marks the usage of the index by creating a marker file, but this one is using native OS file locks so that even if the JVM is terminated the locks will be cleaned up. This implementation has known problems on NFS, avoid it on network shares. native is the default implementation for the filesystem , filesystem-master and filesystem-slave directory providers.
single	org.apache.lucene.store.SingleInstanceLockFactory	This LockFactory doesn't use a file marker but is a Java object lock held in memory; therefore it's possible to use it only when you are sure the index is not going to be shared by any other process. This is the default implementation for the ram directory provider.
none	org.apache.lucene.store.NoLockFactory	Changes to this index are not coordinated by a lock.

The following is an example of locking strategy configuration:

```
hibernate.search.default.locking_strategy = simple
hibernate.search.Animals.locking_strategy = native
hibernate.search.Books.locking_strategy = org.custom.components.MyLockingFactory
```

[Report a bug](#)

23.2.8. Exception Handling Configuration

Hibernate Search allows you to configure how exceptions are handled during the indexing process. If no configuration is provided then exceptions are logged to the log output by default. It is possible to explicitly declare the exception logging mechanism as follows:

```
hibernate.search.error_handler = log
```

The default exception handling occurs for both synchronous and asynchronous indexing. Hibernate Search provides an easy mechanism to override the default error handling implementation.

In order to provide your own implementation you must implement the **ErrorHandler** interface, which provides the **handle(ErrorContext context)** method. **ErrorContext** provides a reference to the primary **LuceneWork** instance, the underlying exception and any subsequent **LuceneWork** instances that could not be processed due to the primary exception.

```
public interface ErrorContext {
    List<LuceneWork> getFailingOperations();
    LuceneWork getOperationAtFault();
    Throwable getThrowable();
    boolean hasErrors();
}
```

To register this error handler with Hibernate Search you must declare the fully qualified classname of your **ErrorHandler** implementation in the configuration properties:

```
hibernate.search.error_handler = CustomerErrorHandler
```

[Report a bug](#)

23.2.9. Index Format Compatibility

Hibernate Search does not currently offer a backwards compatible API or tool to facilitate porting applications to newer versions. The API uses Apache Lucene for index writing and searching. Occasionally an update to the index format may be required. In this case, there is a possibility that data will need to be re-indexed if Lucene is unable to read the old format.



WARNING

Back up indexes before attempting to update the index format.

Hibernate Search exposes the **hibernate.search.lucene_version** configuration property. This property instructs Analyzers and other Lucene classes to conform to their behaviour as defined in an older version of Lucene. See also **org.apache.lucene.util.Version** contained in the **lucene-core.jar**. If the option is not specified, Hibernate Search instructs Lucene to use the version default. It is recommended that the version used is explicitly defined in the configuration to prevent automatic changes when an upgrade occurs. After an upgrade, the configuration values can be updated explicitly if required.

Example 23.11. Force Analyzers to be compatible with a Lucene 3.0 created index

```
hibernate.search.lucene_version = LUCENE_30
```

The configured **SearchFactory** is global and affects all Lucene APIs that contain the relevant parameter. If Lucene is used and Hibernate Search is bypassed, apply the same value to it for consistent results.

[Report a bug](#)

23.2.10. Disable Hibernate Search

Hibernate Search can be partially or completely disabled as required. Hibernate Search's indexing can be disabled, for example, if the index is read-only, or you prefer to perform indexing manually, rather than automatically. It is also possible to completely disable Hibernate Search, preventing indexing and searching.

Disable Indexing

To disable Hibernate Search indexing, change the **indexing_strategy** configuration option to **manual**, then restart JBoss EAP.

```
hibernate.search.indexing_strategy = manual
```

Disable Hibernate Search Completely

To disable Hibernate Search completely, disable all listeners by changing the **autoregister_listeners** configuration option to **false**, then restart JBoss EAP.

```
hibernate.search.autoregister_listeners = false
```

[Report a bug](#)

23.3. MONITORING

23.3.1. Monitoring

Hibernate Search offers access to a **Statistics** object via **SearchFactory.getStatistics()**. It allows you, for example, to determine which classes are indexed and how many entities are in the index. This information is always available. However, by specifying the **hibernate.search.generate_statistics** property in your configuration you can also collect total and average Lucene query and object loading timings.

Hibernate Search provides several methods of monitoring its operations. The list of indexed classes and number of entities per index are always available from the **Statistics** object via the **SearchFactory.getStatistics()** method. To obtain total and average Lucene query and object loading timings, specify the **hibernate.search.generate_statistics** property in your configuration.

Access to Statistics via JMX

To enable access to statistics via JMX, set the property **hibernate.search.jmx_enabled** to **true**. This will automatically register the **StatisticsInfoMBean** bean, providing access to statistics via the **Statistics** object. Depending on your configuration the **IndexingProgressMonitorMBean** bean may also be registered.

Monitoring Indexing

If the mass indexer API is used, you can monitor indexing progress via the **IndexingProgressMonitorMBean** bean. The bean is only bound to JMX while indexing is in progress.



NOTE

JMX beans can be accessed remotely via JConsole by setting the system property **com.sun.management.jmxremote** to **true**.

[Report a bug](#)

CHAPTER 24. DEPLOY JBOSS EAP 6 ON AMAZON EC2

24.1. INTRODUCTION

24.1.1. About Amazon EC2

Amazon Elastic Compute Cloud (Amazon EC2) is a service operated by amazon.com that provides customers with a customizable virtual computing environment. An Amazon Machine Image (AMI) can be booted using the service to create a virtual machine or instance. Users can install whatever software they require on an instance and are charged according to the capacity used. Amazon EC2 is designed to be flexible and allow users to quickly scale their deployed applications.

You can read more about it at the Amazon EC2 website, <http://aws.amazon.com/ec2/>.

[Report a bug](#)

24.1.2. About Amazon Machine Instances (AMIs)

An Amazon Machine Image (AMI) is a template for a EC2 virtual machine instance. Users create EC2 instances by selecting an appropriate AMI to create the instance from. The primary component of an AMI is a read-only filesystem that contains an installed operating system as well as other software. Each AMI has different software installed for different use cases. Amazon EC2 includes many AMIs to choose from provided by both amazon.com and third parties. Users can also create their own custom AMIs.

[Report a bug](#)

24.1.3. About JBoss Cloud Access

JBoss Cloud Access is a Red Hat subscription feature that provides support for JBoss EAP 6 on Red Hat certified cloud infrastructure providers such as Amazon EC2. JBoss Cloud Access allows you to move your subscriptions between traditional servers and public cloud-based resources in a simple and cost-effective manner.

You can find out more details about it at <http://www.redhat.com/en/technologies/cloud-computing/cloud-access>.

[Report a bug](#)

24.1.4. JBoss Cloud Access Features

Membership in the JBoss Cloud Access program provides access to supported private Amazon Machine Images (AMIs) created by Red Hat.

The Red Hat AMIs have the following software pre-installed and fully supported by Red Hat:

- Red Hat Enterprise Linux 6
- JBoss EAP 6
- The JBoss Operations Network (JON) 3 agent
- Product updates with RPMs using Red Hat Update Infrastructure.

Each of the Red Hat AMIs are only a starting point, requiring further configuration to the requirements of your application.



IMPORTANT

JBoss Cloud Access does not currently provide support for the full-ha profile, in either standalone instances or a managed domain.

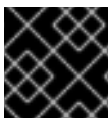
[Report a bug](#)

24.1.5. Supported Amazon EC2 Instance Types

JBoss Cloud Access supports the following Amazon EC2 instance types. Refer to the *Amazon EC2 User Guide* for more details about each instance type, <http://docs.amazonwebservices.com/AWSEC2/latest/UserGuide/instance-types.html>.

Table 24.1. Supported Amazon EC2 Instance Types

Instance Type	Description
Standard Instance	Standard Instances are general purpose environments with a balanced memory-to-CPU ratio.
High Memory Instance	High Memory Instances have more memory allocated to them than Standard Instances. High Memory Instances are suited for high throughput applications such as databases or memory caching applications.
High CPU Instance	High CPU Instances have more CPU resources allocated than memory and are suited for relatively low throughput but CPU intensive applications.



IMPORTANT

The instance type **Micro (t1.micro)** is not suitable for deployment of JBoss EAP 6.

[Report a bug](#)

24.1.6. Supported Red Hat AMIs

The supported Red Hat AMIs can be identified by their AMI Name.

The JBoss EAP 6 AMIs are named using the following syntax:

```
RHEL-osversion-JBEAP-version-arch-creationdate
```

version is the version number of JBoss EAP installed in the AMI. Example **6.3**.

osversion is the version number of Red Hat Enterprise Linux installed in the AMI. Example **6.2**.

arch is the architecture of the AMI. This will be **x86_64** or **i386**.

creationdate is the date that the AMI was created in the format of *YYYYMMDD*. Example **20120501**.

Example AMI name: **RHEL-6.2-JBEAP-6.0.0-x86_64-20120501**.

[Report a bug](#)

24.2. DEPLOYING JBOSS EAP 6 ON AMAZON EC2

24.2.1. Overview of Deploying JBoss EAP 6 on Amazon EC2

JBoss EAP 6 can be deployed using the Amazon EC2 AMI. The AMI contains everything that is required for deployment of clustered and non-clustered instances.

Deploying a non-clustered instance is the easiest scenario. It requires only that you make a few configuration changes to specify your application deployment when creating the instance.

Deploying clustered instances requires more configuration. It is recommended you create a Virtual Private Cloud to contain your cluster. Use of a JBoss EAP instance to act as a mod_cluster proxy is optional but if you take this option, an S3 bucket for the S3_PING JGroups discovery protocol is also required.

Each of these steps is detailed below but it is assumed that you have some experience with JBoss EAP 6, Red Hat Enterprise Linux 6 and Amazon EC2.

The following documentation is recommended additional reference:

- JBoss EAP 6

https://access.redhat.com/documentation/en-us/red_hat_jboss_enterprise_application_platform/?version=6.4

- Red Hat Enterprise Linux 6

https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/

- Amazon Web Services

<http://aws.amazon.com/documentation/>

[Report a bug](#)

24.3. NON-CLUSTERED JBOSS EAP 6

24.3.1. About Non-clustered Instances

A non-clustered instance is a single Amazon EC2 instance running JBoss EAP 6. It is not part of a cluster.

[Report a bug](#)

24.4. NON-CLUSTERED INSTANCES

24.4.1. Launch a Non-clustered JBoss EAP 6 Instance

Summary

This topic covers the steps required to launch a non-clustered instance of JBoss EAP 6 on a Red Hat AMI (Amazon Machine Image).

Prerequisites

- A suitable Red Hat AMI. Refer to [Section 24.1.6, “Supported Red Hat AMIs”](#).
- A pre-configured Security Group which allows incoming requests on at least ports 22, 8080, and 9990.

Procedure 24.1. Launch a Non-clustered Instance of JBoss EAP 6 on a Red Hat AMI (Amazon Machine Image)

1. Configure the **User Data** field. The configurable parameters are available here: [Section 24.10.1, “Permanent Configuration Parameters”](#), [Section 24.10.2, “Custom Script Parameters”](#).

Example 24.1. Example User Data Field

The example shows the User Data field for a non-clustered JBoss EAP 6 instance. The password for the user **admin** has been set to **adminpwd**.

```
JBOSSAS_ADMIN_PASSWORD=adminpwd
JBOSS_IP=0.0.0.0 #listen on all IPs and interfaces

# In production, access to these ports needs to be restricted for security reasons
PORTS_ALLOWED="9990 9443"

cat> $USER_SCRIPT << "EOF"

# Get the application to be deployed from an Internet URL
# mkdir -p /usr/share/java/jboss-ec2-eap-applications
# wget https://<your secure storage hostname>/<path>/<app name>.war -O
/usr/share/java/jboss-ec2-eap-applications/<app name>.war

# Create a file of CLI commands to be executed after starting the server
cat> $USER_CLI_COMMANDS << "EOC"
# deploy /usr/share/java/jboss-ec2-eap-applications/<app name>.war
EOC

EOF
```

2. **For Production Instances**

For a production instance, add the following line beneath the **USER_SCRIPT** line of the **User Data** field, to ensure security updates are applied on boot.

```
yum -y update
```



NOTE

yum -y update should be run regularly, to apply security fixes and enhancements.

3. Launch the Red Hat AMI instance.

Result

A non-clustered instance of JBoss EAP 6 has been configured, and launched on a Red Hat AMI.

[Report a bug](#)

24.4.2. Deploy an Application on a non-clustered JBoss EAP 6 Instance

Summary

This topic covers deploying an application to a non-clustered JBoss EAP 6 instance on a Red Hat AMI.

1. ◦ Deploy the Sample Application

Add the following lines to the **User Data** field:

```
# Deploy the sample application from the local filesystem
deploy --force /usr/share/java/jboss-ec2-eap-samples/hello.war
```

Example 24.2. Example User Data Field with Sample Application

This example uses the sample application provided on the Red Hat AMI. It also includes basic configuration for a non-clustered JBoss EAP 6 instance. The password for the user **admin** has been set to **adminpwd**.

```
JBOSSAS_ADMIN_PASSWORD=adminpwd
JBOSS_IP=0.0.0.0 #listen on all IPs and interfaces

# In production, access to these ports needs to be restricted for security reasons
PORTS_ALLOWED="9990 9443"

cat> $USER_SCRIPT << "EOF"

# Create a file of CLI commands to be executed after starting the server
cat> $USER_CLI_COMMANDS << "EOC"

# Deploy the sample application from the local filesystem
deploy --force /usr/share/java/jboss-ec2-eap-samples/hello.war
EOC

EOF
```

◦ Deploy a Custom Application

Add the following lines to the **User Data** field, configuring the application name and the URL:

```
# Get the application to be deployed from an Internet URL
mkdir -p /usr/share/java/jboss-ec2-eap-applications
wget https://<your secure storage hostname>/<path>/<app name>.war -O
/usr/share/java/jboss-ec2-eap-applications/<app name>.war
```

Example 24.3. Example User Data Field with Custom Application

This example uses an application called **MyApp**, and includes basic configuration for a non-clustered JBoss EAP 6 instance. The password for the user **admin** has been set to **adminpwd**.

```
JBOSSAS_ADMIN_PASSWORD=adminpwd
JBOSS_IP=0.0.0.0 #listen on all IPs and interfaces

# In production, access to these ports needs to be restricted for security reasons
PORTS_ALLOWED="9990 9443"

cat> $USER_SCRIPT << "EOF"

# Get the application to be deployed from an Internet URL
mkdir -p /usr/share/java/jboss-ec2-eap-applications
wget https://PATH_TO_MYAPP/MyApp.war -O /usr/share/java/jboss-ec2-eap-applications/MyApp.war

# Create a file of CLI commands to be executed after starting the server
cat> $USER_CLI_COMMANDS << "EOC"
deploy /usr/share/java/jboss-ec2-eap-applications/MyApp.war
EOC

EOF
```

2. Launch the Red Hat AMI instance.

Result

The application has been successfully deployed to JBoss EAP 6.

[Report a bug](#)

24.4.3. Test the Non-clustered JBoss EAP 6 Instance

Summary

This topic covers the steps required to test that the non-clustered JBoss EAP 6 is running correctly.

Procedure 24.2. Test the Non-clustered JBoss EAP 6 Instance is Running Correctly

1. Determine the instance's **Public DNS**, located in the instance's details pane.
2. Navigate to **http://<public-DNS>:8080**.
3. Confirm that the JBoss EAP home page appears, including a link to the Admin console. If the home page is not available, refer here: [Section 24.11.1, "About Troubleshooting Amazon EC2"](#).
4. Click on the **Admin Console** hyperlink.
5. Log in:
 - Username: **admin**
 - Password: Specified in the **User Data** field here: [Section 24.4.1, "Launch a Non-clustered JBoss EAP 6 Instance"](#).
6. **Test the Sample Application**

Navigate to <http://<public-DNS>:8080/hello> to test that the sample application is running successfully. The text **Hello World!** should appear in the browser. If the text is not visible, refer here: [Section 24.11.1, "About Troubleshooting Amazon EC2"](#).

7. Log out of the JBoss EAP 6 Admin Console.

Result

The JBoss EAP 6 instance is running correctly.

[Report a bug](#)

24.5. NON-CLUSTERED MANAGED DOMAINS

24.5.1. Launch an Instance to Serve as a Domain Controller

Summary

This topic covers the steps required to launch a non-clustered JBoss EAP 6 managed domain on a Red Hat AMI (Amazon Machine Image).

Prerequisites

- A suitable Red Hat AMI. Refer to [Section 24.1.6, "Supported Red Hat AMIs"](#).
- [Section 24.6.3, "Create a Virtual Private Cloud \(VPC\)"](#)
- [Section 24.6.4, "Launch an Apache HTTP Server Instance to Serve as a mod_cluster Proxy and a NAT Instance for the VPC"](#)
- [Section 24.6.5, "Configure the VPC Private Subnet Default Route"](#)
- [Section 24.6.7, "Configure IAM Setup"](#)
- [Section 24.6.9, "Configure S3 Bucket Setup"](#)

Procedure 24.3. Launch a non-clustered JBoss EAP 6 managed domain on a Red Hat AMI

1. In the Security Group tab, ensure all traffic is allowed. Red Hat Enterprise Linux's built-in firewall capabilities can be used to restrict access if desired.
2. Set the public subnet of the VPC to *running*.
3. Select a static IP.
4. Configure the **User Data** field. The configurable parameters are available here: [Section 24.10.1, "Permanent Configuration Parameters"](#), [Section 24.10.2, "Custom Script Parameters"](#). For further information on domain controller discovery on Amazon EC2, see [Section 24.5.4, "Configuring Domain Controller Discovery and Failover on Amazon EC2"](#).

Example 24.4. Example User Data Field

The example shows the User Data field for a non-clustered JBoss EAP 6 managed domain. The password for the user **admin** has been set to **admin**.

```
## password that will be used by slave host controllers to connect to the domain controller
```

```

JBOSSAS_ADMIN_PASSWORD=admin

## subnet prefix this machine is connected to
SUBNET=10.0.0.

## S3 domain controller discovery setup
# JBOSS_DOMAIN_S3_SECRET_ACCESS_KEY=<your secret key>
# JBOSS_DOMAIN_S3_ACCESS_KEY=<your access key>
# JBOSS_DOMAIN_S3_BUCKET=<your bucket name>

#### to run the example no modifications below should be needed ####
JBOSS_DOMAIN_CONTROLLER=true
PORTS_ALLOWED="9999 9990 9443"
JBOSS_IP=`hostname | sed -e 's/ip-/' -e 'y/-/.'` #listen on public/private EC2 IP address

cat > $USER_SCRIPT << "EOF"
## Get the application to be deployed from an Internet URL
# mkdir -p /usr/share/java/jboss-ec2-eap-applications
# wget https://<your secure storage hostname>/<path>/<app name>.war -O
/usr/share/java/jboss-ec2-eap-applications/<app name>.war

## Create a file of CLI commands to be executed after starting the server
cat> $USER_CLI_COMMANDS << "EOC"

# Add the modcluster subsystem to the default profile to set up a proxy
/profile=default/subsystem=web/connector=ajp:add(name=ajp,protocol=AJP/1.3,scheme=ht
p,socket-binding=ajp)
/:composite(steps=[ {"operation" => "add", "address" => [ ("profile" => "default"),
("subsystem" => "modcluster") ] }, {"operation" => "add", "address" => [ ("profile" =>
"default"), ("subsystem" => "modcluster"), ("mod-cluster-config" => "configuration") ],
"advertise" => "false", "proxy-list" => "${jboss.modcluster.proxyList}", "connector" =>
"ajp"}, { "operation" => "add", "address" => [ ("profile" => "default"), ("subsystem" =>
"modcluster"), ("mod-cluster-config" => "configuration"), ("dynamic-load-provider" =>
"configuration") ] }, { "operation" => "add", "address" => [ ("profile" => "default"),
("subsystem" => "modcluster"), ("mod-cluster-config" => "configuration"), ("dynamic-load-
provider" => "configuration"), ("load-metric" => "busyness")], "type" => "busyness"} ])

# Deploy the sample application from the local filesystem
deploy /usr/share/java/jboss-ec2-eap-samples/hello.war --server-groups=main-server-
group
EOC

## this will workaround the problem that in a VPC, instance hostnames are not resolvable
echo -e "127.0.0.1\tlocalhost.localdomain localhost" > /etc/hosts
echo -e "::1\tlocalhost6.localdomain6 localhost6" >> /etc/hosts
for (( i=1 ; i<255 ; i++ )); do
    echo -e "$SUBNET$i\tip-{$SUBNET//.-}$i" ;
done >> /etc/hosts

EOF

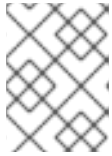
```

5. For Production Instances

For a production instance, add the following line beneath the **USER_SCRIPT** line of the **User Data** field, to ensure security updates are applied on boot.

-

```
yum -y update
```



NOTE

yum -y update should be run regularly, to apply security fixes and enhancements.

6. Launch the Red Hat AMI instance.

Result

A non-clustered JBoss EAP 6 managed domain has been configured, and launched on a Red Hat AMI.

[Report a bug](#)

24.5.2. Launch One or More Instances to Serve as Host Controllers

Summary

This topic covers the steps required to launch one or more instances of JBoss EAP 6 to serve as non-clustered host controllers on a Red Hat AMI (Amazon Machine Image).

Prerequisites

- Configure and launch the non-clustered domain controller. Refer to [Section 24.5.1, “Launch an Instance to Serve as a Domain Controller”](#).
- [Section 24.6.7, “Configure IAM Setup”](#)
- [Section 24.6.9, “Configure S3 Bucket Setup”](#)

Procedure 24.4. Launch Host Controllers

For each instance you would like to create, repeat the following steps:

1. Select an AMI.
2. Define the desired number of instances (the number of slave host controllers).
3. Select the VPC and instance type.
4. Click on Security Group.
5. Ensure that all traffic from the JBoss EAP 6 subnet is allowed.
6. Define other restrictions as desired.
7. Add the following into the User Data field:

```
## mod cluster proxy addresses
MOD_CLUSTER_PROXY_LIST=10.0.0.4:7654

## host controller setup
### static domain controller discovery setup
JBOSS_DOMAIN_MASTER_ADDRESS=10.0.0.5
```

```

### S3 domain controller discovery setup
# JBOSS_DOMAIN_S3_SECRET_ACCESS_KEY=<your secret key>
# JBOSS_DOMAIN_S3_ACCESS_KEY=<your access key>
# JBOSS_DOMAIN_S3_BUCKET=<your bucket name>

JBOSS_HOST_PASSWORD=<password for slave host controllers>

## subnet prefix this machine is connected to
SUBNET=10.0.1.

##### to run the example no modifications below should be needed #####
JBOSS_HOST_USERNAME=admin
PORTS_ALLOWED="1024:65535"
JBOSS_IP=`hostname | sed -e 's/ip-//' -e 'y/-/./'` #listen on public/private EC2 IP address

cat > $USER_SCRIPT << "EOF"
## Server instance configuration
sed -i "s/other-server-group/main-server-group/"
$JBOSS_CONFIG_DIR/$JBOSS_HOST_CONFIG

## this will workaround the problem that in a VPC, instance hostnames are not resolvable
echo -e "127.0.0.1\tlocalhost.localdomain localhost" > /etc/hosts
echo -e ">:::1\tlocalhost6.localdomain6 localhost6" >> /etc/hosts
for (( i=1 ; i<255 ; i++ )); do
    echo -e "$SUBNET$i\tip-{$SUBNET//./-}$i" ;
done >> /etc/hosts

EOF

```

For further information on domain controller discovery on Amazon EC2, see [Section 24.5.4, “Configuring Domain Controller Discovery and Failover on Amazon EC2”](#).

8. For Production Instances

For a production instance, add the following line beneath the **USER_SCRIPT** line of the **User Data** field, to ensure security updates are applied on boot.

```
yum -y update
```



NOTE

yum -y update should be run regularly, to apply security fixes and enhancements.

9. Launch the Red Hat AMI instance.

Result

The JBoss EAP 6 non-clustered host controllers are configured and launched on a Red Hat AMI.

[Report a bug](#)

24.5.3. Test the Non-Clustered JBoss EAP 6 Managed Domain

Summary

This topic covers the steps required to test the non-clustered JBoss EAP 6 managed domain on a Red Hat AMI (Amazon Machine Image).

To test the managed domain you must know the elastic IP addresses of both the Apache HTTP server and JBoss EAP 6 domain controller.

Prerequisites

- Configure and launch the domain controller. See [Section 24.5.1, “Launch an Instance to Serve as a Domain Controller”](#).
- Configure and launch the host controllers. See [Section 24.5.2, “Launch One or More Instances to Serve as Host Controllers”](#).

Procedure 24.5. Test the Web Server

- Navigate to **http://ELASTIC_IP_OF_APACHE_HTTPD** in a browser to confirm the web server is running successfully.

Procedure 24.6. Test the Domain Controller

1. Navigate to **http://ELASTIC_IP_OF_DOMAIN_CONTROLLER:9990/console**
2. Log in using the username of **admin** and the password specified in the User Data field for the domain controller and the admin console landing page for a managed domain should appear (**http://ELASTIC_IP_OF_DOMAIN_CONTROLLER:9990/console/App.html#server-instances**).
3. Click the **Server** label at the top right side of the screen, and select any of the host controllers in the **Host** dropdown menu at the top left side of the screen.
4. Verify that each host controller has two server configurations called **server-one** and **server-two** and that they both belong to the **main-server-group**.
5. Log out of the JBoss EAP 6 Admin Console.

Procedure 24.7. Test the Host Controllers

1. Navigate to **http://ELASTIC_IP_OF_APACHE_HTTPD/hello** to test that the sample application is running successfully. The text **Hello World!** should appear in the browser.

If the text is not visible, refer here: [Section 18.5.1, “About Troubleshooting Amazon EC2”](#).

2. Connect to the Apache HTTP server instance:

```
$ ssh -L7654:localhost:7654 ELASTIC_IP_OF_APACHE_HTTPD
```

3. Navigate to **http://localhost:7654/mod_cluster-manager** to confirm all instances are running correctly.

Result

The JBoss EAP 6 web server, domain controller, and host controllers are running correctly on a Red Hat AMI.

[Report a bug](#)

24.5.4. Configuring Domain Controller Discovery and Failover on Amazon EC2

For a managed domain running on Amazon EC2, in addition to static domain controller discovery, host controllers can dynamically discover a domain controller using the Amazon S3 storage system. In particular, host controllers and the domain controller can be configured with information needed to access an Amazon S3 bucket.

Using this configuration, when a domain controller is started, it writes its contact information to an S3 file in the bucket. Whenever a host controller attempts to contact the domain controller, it gets the domain controller's contact information from the S3 file.

This means that if the domain controller's contact information changes (for example, it is common for an EC2 instance's IP address to change when it is stopped and started), the host controllers do not need to be reconfigured. The host controllers are able to get the domain controller's new contact information from the S3 file.

You can automatically enable domain controller discovery by passing the **JBOSS_DOMAIN_S3_ACCESS_KEY**, **JBOSS_DOMAIN_S3_SECRET_ACCESS_KEY**, and **JBOSS_DOMAIN_S3_BUCKET** parameters to the JBoss EAP 6 instance when launching it. See [Section 24.10.1, "Permanent Configuration Parameters"](#) for configurable parameters. Alternatively, you can manually configure domain discovery using the following configuration.

The manual domain controller discovery configuration is specified using the following properties:

access-key

The Amazon AWS user account access key.

secret-access-key

The Amazon AWS user account secret access key.

location

The Amazon S3 bucket to be used.

The following are example host controller and domain controller configurations. Although one discovery option is shown in the examples below, it is possible to configure any number of static discovery or S3 discovery options. For details on the domain discovery and failover process, see [Section 1.7, "About Domain Controller Discovery and Failover"](#).

Example 24.5. Host Controller Configuration

```
<domain-controller>
  <remote security-realm="ManagementRealm">
    <discovery-options>
      <discovery-option name="s3-discovery"
code="org.jboss.as.host.controller.discovery.S3Discovery" module="org.jboss.as.host-controller">
        <property name="access-key" value="S3_ACCESS_KEY"/>
        <property name="secret-access-key" value="S3_SECRET_ACCESS_KEY"/>
        <property name="location" value="S3_BUCKET_NAME"/>
      </discovery-option>
    </discovery-options>
  </remote>
</domain-controller>
```

Example 24.6. Domain Controller Configuration

```

<domain-controller>
  <local>
    <discovery-options>
      <discovery-option name="s3-discovery"
code="org.jboss.as.host.controller.discovery.S3Discovery" module="org.jboss.as.host-controller">
        <property name="access-key" value="S3_ACCESS_KEY"/>
        <property name="secret-access-key" value="S3_SECRET_ACCESS_KEY"/>
        <property name="location" value="S3_BUCKET_NAME"/>
      </discovery-option>
    </discovery-options>
  </local>
</domain-controller>

```

[Report a bug](#)

24.6. CLUSTERED JBOSS EAP 6

24.6.1. About Clustered Instances

A clustered instance is an Amazon EC2 instance running JBoss EAP 6 with clustering enabled. Another instance running the Apache HTTP server will be acting as the proxy for the instances in the cluster.

The JBoss EAP 6 AMIs include two configuration files for use in clustered instances, **standalone-ec2-ha.xml** and **standalone-mod_cluster-ec2-ha.xml**. Each of these configuration files provides clustering without the use of multicast because Amazon EC2 does not support multicast. This is done by using TCP unicast for cluster communications and S3_PING as the discovery protocol. The **standalone-mod_cluster-ec2-ha.xml** configuration also provides easy registration with mod_cluster proxies.

Similarly, the **domain-ec2.xml** configuration file provides two profiles for use in clustered managed domains: ec2-ha, and mod_cluster-ec2-ha.

[Report a bug](#)

24.6.2. About Virtual Private Clouds

An Amazon Virtual Private Cloud (Amazon VPC) is a feature of Amazon Web Services (AWS) that allows you to isolate a set of AWS resources in a private network. The topology and configuration of this private network can be customized to your needs.

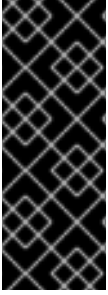
Refer to the Amazon Virtual Private Cloud website for more information <http://aws.amazon.com/vpc/>.

[Report a bug](#)

24.6.3. Create a Virtual Private Cloud (VPC)

Summary

This topic covers the steps required to create a Virtual Private Cloud, using a database external to the VPC as an example. Your security policies may require connection to the database to be encrypted. Please refer to Amazon's *RDS FAQ* for details about encrypting the database connections.



IMPORTANT

VPC is recommended for a JBoss EAP 6 cluster setup as it greatly simplifies secure communication between cluster nodes, a JON Server and the mod_cluster proxy. Without a VPC, these communication channels need to be encrypted and authenticated.

For detailed instructions on configuring SSL, refer to the *Core Management Security Guide*.

1. Go to the VPC tab in the AWS console.
2. Subscribe to the service if needed.
3. Click on "**Create new VPC**".
4. Choose a VPC with one public and one private subnet.
 - a. Set the public subnet to be **10.0.0.0/24**.
 - b. Set the private subnet to be **10.0.1.0/24**.
5. Go to **Elastic IPs**.
6. Create an elastic IP for use by the mod_cluster proxy/NAT instance.
7. Go to **Security groups** and create a security group to allow all traffic in and out.
8. Go to Network ACLs
 - a. Create an ACL to allow all traffic in and out.
 - b. Create an ACL to allow all traffic out and traffic in on only TCP ports **22, 8009, 8080, 8443, 9443, 9990** and **16163**.

Result

The Virtual Private Cloud has been successfully created.

[Report a bug](#)

24.6.4. Launch an Apache HTTP Server Instance to Serve as a mod_cluster Proxy and a NAT Instance for the VPC

Summary

This topic covers the steps required to launch an Apache HTTP server instance to serve as a mod_cluster proxy and a NAT instance for the Virtual Private Cloud.

Prerequisites

- [Section 24.6.3, "Create a Virtual Private Cloud \(VPC\)"](#)

Procedure 24.8. Launch an Apache HTTP server Instance to Serve as a mod_cluster proxy and a NAT Instance for the VPC

1. Create an elastic IP for this instance.

2. Select an AMI.
3. Go to **Security Group** and allow all traffic (use Red Hat Enterprise Linux's built-in firewall capabilities to restrict access if desired).
4. Select "**running**" in the public subnet of the VPC.
5. Select a static IP (e.g. **10.0.0.4**).
6. Put the following in the **User Data** field:

```
JBOSSCONF=disabled

cat > $USER_SCRIPT << "EOS"

echo 1 > /proc/sys/net/ipv4/ip_forward
echo 0 > /proc/sys/net/ipv4/conf/all/rp_filter
echo 0 > /proc/sys/net/ipv4/conf/eth0/rp_filter

iptables -I INPUT 4 -s 10.0.1.0/24 -p tcp --dport 7654 -j ACCEPT
iptables -I INPUT 4 -p tcp --dport 80 -j ACCEPT

iptables -I FORWARD -m state --state RELATED,ESTABLISHED -j ACCEPT
iptables -I FORWARD -s 10.0.1.0/24 -j ACCEPT
iptables -t nat -A POSTROUTING -o eth0 ! -s 10.0.0.4 -j MASQUERADE

# balancer module incompatible with mod_cluster
sed -i -e 's/LoadModule proxy_balancer_module/#0/' /etc/httpd/conf/httpd.conf

cat > /etc/httpd/conf.d/mod_cluster.conf << "EOF"
#LoadModule proxy_module modules/mod_proxy.so
#LoadModule proxy_ajp_module modules/mod_proxy_ajp.so
LoadModule slotmem_module modules/mod_slotmem.so
LoadModule manager_module modules/mod_manager.so
LoadModule proxy_cluster_module modules/mod_proxy_cluster.so
LoadModule advertise_module modules/mod_advertise.so

Listen 7654

# workaround JBPAPP-4557
MemManagerFile /var/cache/mod_proxy/manager

<VirtualHost *:7654>
  <Location /mod_cluster-manager>
    SetHandler mod_cluster-manager
    Order deny,allow
    Deny from all
    Allow from 127.0.0.1
  </Location>

  <Location />
    Order deny,allow
    Deny from all
    Allow from 10.
    Allow from 127.0.0.1
  </Location>
```

```

KeepAliveTimeout 60
MaxKeepAliveRequests 0
ManagerBalancerName mycluster
ServerAdvertise Off
EnableMCPMReceive
</VirtualHost>
EOF

echo "`hostname | sed -e 's/ip-//' -e 'y/-/.'` `hostname`" >> /etc/hosts

semanage port -a -t http_port_t -p tcp 7654 #add port in the apache port list for the below to work
setsebool -P httpd_can_network_relay 1 #for mod_proxy_cluster to work
chcon -t httpd_config_t -u system_u /etc/httpd/conf.d/mod_cluster.conf

#### Uncomment the following line when launching a managed domain ####
# setsebool -P httpd_can_network_connect 1

service httpd start

EOS

```

7. Disable the Amazon EC2 cloud source/destination checking for this instance so it can act as a router.
 - a. Right-click on the running Apache HTTP server instance and choose "**Change Source/Dest check**".
 - b. Click on **Yes, Disable**.
8. Assign the elastic IP to this instance.

Result

The Apache HTTP server instance has been launched successfully.

[Report a bug](#)

24.6.5. Configure the VPC Private Subnet Default Route

Summary

This topic covers the steps required to configure the VPC private subnet default route. JBoss EAP 6 cluster nodes will run in the private subnet of the VPC, but cluster nodes require Internet access for S3 connectivity. A default route needs to be set to go through the NAT instance.

Procedure 24.9. Configure the VPC Private Subnet Default Route

1. Navigate to the Apache HTTP server instance in the Amazon AWS console.
2. Navigate to the **VPC → route tables**.
3. Click on the routing table used by the private subnet.
4. In the field for a new route enter **0.0.0.0/0**.

5. Click on "**Select a target**".
6. Select "**Enter Instance ID**".
7. Choose the ID of the running Apache HTTP server instance.

Result

The default route has been successfully configured for the VPC subnet.

[Report a bug](#)

24.6.6. About Identity and Access Management (IAM)

Identity and Access Management (IAM) provides configurable security for your AWS resources. IAM can be configured to use accounts created in IAM or to provide identity federation between IAM and your own identity services.

Refer to the AWS Identity and Access Management website for more information <http://aws.amazon.com/iam/>.

[Report a bug](#)

24.6.7. Configure IAM Setup

Summary

This topic covers the configuration steps required for setting up IAM for clustered JBoss EAP 6 instances. The **S3_PING** protocol uses an S3 bucket to discover other cluster members. **JGroups** version 3.0.x requires Amazon AWS account access and secret keys to authenticate against the S3 service.

Because S3 domain controller discovery makes use of an S3 bucket, it requires Amazon AWS account access and secret keys to authenticate against the S3 service (similar to the **S3_PING** protocol used by JGroups). The IAM user and S3 bucket used for S3 discovery must be different from the IAM user and S3 bucket used for clustering.

It is a security risk to enter your main account credentials in the user-data field, store them online or in an AMI. To circumvent this, a separate account can be created using the Amazon IAM feature which would be only granted access to a single S3 bucket.

Procedure 24.10. Configure IAM Setup

1. Go to the IAM tab in the AWS console.
2. Click on **users**.
3. Select **Create New Users**.
4. Choose a name, and ensure the **Generate an access key for each User** option is checked.
5. Select **Download credentials**, and save them in a secure location.
6. Close the window.
7. Click on the newly created user.

8. Make note of the **User ARM** value. This value is required to set up the S3 bucket, documented here: [Section 24.6.9, "Configure S3 Bucket Setup"](#).

Result

The IAM user account has been successfully created.

[Report a bug](#)

24.6.8. About the S3 Bucket

S3 Buckets are the basic organization store unit in the Amazon Simple Storage System (Amazon S3). A bucket can store any number of arbitrary objects and must have a unique name to identify it with Amazon S3..

Refer to the Amazon S3 website for more information, <http://aws.amazon.com/s3/>.

[Report a bug](#)

24.6.9. Configure S3 Bucket Setup

Summary

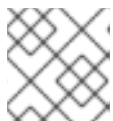
This topic covers the steps required to configure a new S3 bucket.

Prerequisites

- [Section 24.6.7, "Configure IAM Setup"](#).

Procedure 24.11. Configure S3 Bucket Setup

1. Open the **S3** tab in the AWS console.
2. Click on **Create Bucket**.
3. Choose a name for the bucket and click **Create**.



NOTE

Bucket names are unique across the entire S3. Names cannot be reused.

4. Right click on the new bucket and select **Properties**.
5. Click **Add bucket policy** in the permissions tab.
6. Click **New policy** to open the policy creation wizard.
 - a. Copy the following content into the new policy, replacing **arn:aws:iam::055555555555:user/jbosscluster*** with the value defined here: [Section 24.6.7, "Configure IAM Setup"](#). Change both instances of **clusterbucket123** to the name of the bucket defined in step 3 of this procedure.

```
{
  "Version": "2008-10-17",
  "Id": "Policy1312228794320",
```



```

"Statement": [
  {
    "Sid": "Stmnt1312228781799",
    "Effect": "Allow",
    "Principal": {
      "AWS": [
        "arn:aws:iam::055555555555:user/jbosscluster"
      ]
    },
    "Action": [
      "s3:ListBucketVersions",
      "s3:GetObjectVersion",
      "s3:ListBucket",
      "s3:PutBucketVersioning",
      "s3>DeleteObject",
      "s3>DeleteObjectVersion",
      "s3:GetObject",
      "s3:ListBucketMultipartUploads",
      "s3:ListMultipartUploadParts",
      "s3:PutObject",
      "s3:GetBucketVersioning"
    ],
    "Resource": [
      "arn:aws:s3:::clusterbucket123/*",
      "arn:aws:s3:::clusterbucket123"
    ]
  }
]
}

```

Result

A new S3 bucket has been created, and configured successfully.

[Report a bug](#)

24.7. CLUSTERED INSTANCES

24.7.1. Launch Clustered JBoss EAP 6 AMIs

Summary

This topic covers the steps required to launch clustered JBoss EAP 6 AMIs.

Prerequisites

- [Section 24.6.3, "Create a Virtual Private Cloud \(VPC\)"](#).
- [Section 24.6.4, "Launch an Apache HTTP Server Instance to Serve as a mod_cluster Proxy and a NAT Instance for the VPC"](#).
- [Section 24.6.5, "Configure the VPC Private Subnet Default Route"](#).
- [Section 24.6.7, "Configure IAM Setup"](#).
- [Section 24.6.9, "Configure S3 Bucket Setup"](#).

**WARNING**

Running a JBoss EAP 6 cluster in a subnet with network mask smaller than 24 bits or spanning multiple subnets complicates acquiring a unique server peer ID for each cluster member.

Refer to the ***JBOSS_CLUSTER_ID*** variable for information on how to make such a configuration work reliably: [Section 24.10.1, "Permanent Configuration Parameters"](#).

**IMPORTANT**

The auto-scaling Amazon EC2 feature can be used with JBoss EAP 6 cluster nodes. However, ensure it is tested **before** deployment. You should ensure that your particular workloads scale to the desired number of nodes, and that the performance meets your needs for the instance type you are planning to use (different instance types receive a different share of the EC2 cloud resources).

Furthermore, instance locality and current network/storage/host machine/RDS utilization can affect performance of a cluster. Test with your expected real-life loads and try to account for unexpected conditions.

**WARNING**

The Amazon EC2 *scale-down* action terminates the nodes without any chance to gracefully shut down, and, as some transactions might be interrupted, other cluster nodes (and load balancers) will need time to fail over. This is likely to impact your application users' experience.

It is recommended that you either scale down the application cluster manually by disabling the server from the `mod_cluster` management interface until processed sessions are completed, or shut down the JBoss EAP 6 instance gracefully (SSH access to the instance or JON can be used).

Test that your chosen procedure for scaling-down does not lead to adverse effects on your users' experience. Additional measures might be required for particular workloads, load balancers and setups.

Procedure 24.12. Launch Clustered JBoss EAP 6 AMIs

1. Select an AMI.
2. Define the desired number of instances (the cluster size).
3. Select the VPC and instance type.
4. Click on **Security Group**.

5. Ensure that all traffic from the JBoss EAP 6 cluster subnet is allowed.
6. Define other restrictions as desired.
7. Add the following into the **User Data** field:

Example 24.7. Example User Data Field

```
## mod cluster proxy addresses
MOD_CLUSTER_PROXY_LIST=10.0.0.4:7654

## clustering setup
JBOSS_JGROUPS_S3_PING_SECRET_ACCESS_KEY=<your secret key>
JBOSS_JGROUPS_S3_PING_ACCESS_KEY=<your access key>
JBOSS_JGROUPS_S3_PING_BUCKET=<your bucket name>

## password to access admin console
JBOSSAS_ADMIN_PASSWORD=<your password for opening admin console>

## database credentials configuration
JAVA_OPTS="$JAVA_OPTS -Ddb.host=instancename.something.rds.amazonaws.com -
Ddb.database=mydatabase -Ddb.user=<user> -Ddb.passwd=<pass>"

## subnet prefix this machine is connected to
SUBNET=10.0.1.

#### to run the example no modifications below should be needed ####
PORTS_ALLOWED="1024:65535"
JBOSS_IP=`hostname | sed -e 's/ip://' -e 'y/-./'` #listen on public/private EC2 IP address

cat > $USER_SCRIPT << "EOF"
## Get the application to be deployed from an Internet URL
# mkdir -p /usr/share/java/jboss-ec2-eap-applications
# wget https://<your secure storage hostname>/<path>/<app name>.war -O
/usr/share/java/jboss-ec2-eap-applications/<app name>.war

## install the JDBC driver as a core module
yum -y install mysql-connector-java
mkdir -p /usr/share/jbossas/modules/com/mysql/main
cp -v /usr/share/java/mysql-connector-java-*.jar
/usr/share/jbossas/modules/com/mysql/main/mysql-connector-java.jar

cat > /usr/share/jbossas/modules/com/mysql/main/module.xml <<"EOM"
<?xml version="1.0" encoding="UTF-8"?>
<module xmlns="urn:jboss:module:1.0" name="com.mysql">
  <resources>
    <resource-root path="mysql-connector-java.jar"/>
  </resources>
  <dependencies>
    <module name="javax.api"/>
  </dependencies>
</module>
EOM

cat > $USER_CLI_COMMANDS << "EOC"
## Deploy sample application from local filesystem
```

```

deploy --force /usr/share/java/jboss-ec2-eap-samples/cluster-demo.war

## ExampleDS configuration for MySQL database
data-source remove --name=ExampleDS
/subsystem=datasources/jdbc-driver=mysql:add(driver-name="mysql",driver-module-
name="com.mysql")
data-source add --name=ExampleDS --connection-
url="jdbc:mysql://${db.host}:3306/${db.database}" --jndi-
name=java:jboss/datasources/ExampleDS --driver-name=mysql --user-
name="${db.user}" --password="${db.passwd}"
/subsystem=datasources/data-source=ExampleDS:enable
/subsystem=datasources/data-source=ExampleDS:test-connection-in-pool
EOF

## this will workaround the problem that in a VPC, instance hostnames are not resolvable
echo -e "127.0.0.1\tlocalhost.localdomain localhost" > /etc/hosts
echo -e ":::1\tlocalhost6.localdomain6 localhost6" >> /etc/hosts
for (( i=1 ; i<255 ; i++ )); do
    echo -e "$SUBNET$i\tip-${SUBNET//./-}$i" ;
done >> /etc/hosts

EOF

```

Result

The clustered JBoss EAP 6 AMIs have been configured and launched successfully.

[Report a bug](#)

24.7.2. Test the Clustered JBoss EAP 6 Instance

Summary

This topic covers the steps to confirm that the clustered JBoss EAP 6 instances are running correctly.

Procedure 24.13. Testing the Clustered Instance

1. Navigate to http://ELASTIC_IP_OF_APACHE_HTTPD in a browser to confirm the web server is running successfully.
2. **Test the Clustered Nodes**
 - a. Navigate to http://ELASTIC_IP_OF_APACHE_HTTPD/cluster-demo/put.jsp in a browser.
 - b. Verify that one of the cluster nodes logs the following message:

```
Putting date now
```
 - c. Stop the cluster node that logged the message in the previous step.
 - d. Navigate to http://ELASTIC_IP_OF_APACHE_HTTPD/cluster-demo/get.jsp in a browser.
 - e. Verify that the time shown is the same as the time that was PUT using **put.jsp** in Step 2-a.
 - f. Verify that one of the running cluster nodes logs the following message:

Getting date now

- g. Restart the stopped clustered node.
- h. Connect to the Apache HTTP server instance:

```
ssh -L7654:localhost:7654 <ELASTIC_IP_OF_APACHE_HTTPD>
```

- i. Navigate to http://localhost:7654/mod_cluster-manager to confirm all instances are running correctly.

Result

The clustered JBoss EAP 6 instances have been tested, and confirmed to be working correctly.

[Report a bug](#)

24.8. CLUSTERED MANAGED DOMAINS

24.8.1. Launch an Instance to Serve as a Cluster Domain Controller

Summary

This topic covers the steps required to launch a clustered JBoss EAP 6 managed domain on a Red Hat AMI (Amazon Machine Image).

Prerequisites

- A suitable Red Hat AMI. Refer to [Section 24.1.6, "Supported Red Hat AMIs"](#) .
- [Section 24.6.3, "Create a Virtual Private Cloud \(VPC\)"](#)
- [Section 24.6.4, "Launch an Apache HTTP Server Instance to Serve as a mod_cluster Proxy and a NAT Instance for the VPC"](#)
- [Section 24.6.5, "Configure the VPC Private Subnet Default Route"](#)
- [Section 24.6.7, "Configure IAM Setup"](#)
- [Section 24.6.9, "Configure S3 Bucket Setup"](#)

Procedure 24.14. Launch a Cluster Domain Controller

1. Create an elastic IP for this instance.
2. Select an AMI.
3. Go to Security Group and allow all traffic (use Red Hat Enterprise Linux's built-in firewall capabilities to restrict access if desired).
4. Choose "running" in the public subnet of the VPC.
5. Choose a static IP (e.g. **10.0.0.5**).
6. Put the following in the User Data: field:

```

## mod cluster proxy addresses
MOD_CLUSTER_PROXY_LIST=10.0.0.4:7654

## password that will be used by slave host controllers to connect to the domain controller
JBOSSAS_ADMIN_PASSWORD=<password for slave host controllers>

## subnet prefix this machine is connected to
SUBNET=10.0.0.

## S3 domain controller discovery setup
# JBOSS_DOMAIN_S3_SECRET_ACCESS_KEY=<your secret key>
# JBOSS_DOMAIN_S3_ACCESS_KEY=<your access key>
# JBOSS_DOMAIN_S3_BUCKET=<your bucket name>

#### to run the example no modifications below should be needed ####
JBOSS_DOMAIN_CONTROLLER=true
PORTS_ALLOWED="9999 9990 9443"
JBOSS_IP=`hostname | sed -e 's/ip-//' -e 'y/-./.'` #listen on public/private EC2 IP address

cat > $USER_SCRIPT << "EOF"
## Get the application to be deployed from an Internet URL
# mkdir -p /usr/share/java/jboss-ec2-eap-applications
# wget https://<your secure storage hostname>/<path>/<app name>.war -O
/usr/share/java/jboss-ec2-eap-applications/<app name>.war

## Install the JDBC driver as a core module
yum -y install mysql-connector-java
mkdir -p /usr/share/jbossas/modules/com/mysql/main
cp -v /usr/share/java/mysql-connector-java-*.jar
/usr/share/jbossas/modules/com/mysql/main/mysql-connector-java.jar

cat > /usr/share/jbossas/modules/com/mysql/main/module.xml <<"EOM"
<?xml version="1.0" encoding="UTF-8"?>
<module xmlns="urn:jboss:module:1.0" name="com.mysql">
  <resources>
    <resource-root path="mysql-connector-java.jar"/>
  </resources>
  <dependencies>
    <module name="javax.api"/>
  </dependencies>
</module>
EOM

cat > $USER_CLI_COMMANDS << "EOC"
## Deploy the sample application from the local filesystem
deploy /usr/share/java/jboss-ec2-eap-samples/cluster-demo.war --server-groups=other-
server-group

## ExampleDS configuration for MySQL database
data-source --profile=mod_cluster-ec2-ha remove --name=ExampleDS
/profile=mod_cluster-ec2-ha/subsystem=datasources/jdbc-driver=mysql:add(driver-
name="mysql",driver-module-name="com.mysql")
data-source --profile=mod_cluster-ec2-ha add --name=ExampleDS --connection-
url="jdbc:mysql://${db.host}:3306/${db.database}" --jndi-
name=java:jboss/datasources/ExampleDS --driver-name=mysql --user-name="${db.user}" --
password="${db.passwd}"

```

```
/profile=mod_cluster-ec2-ha/subsystem=datasources/data-source=ExampleDS:enable
EOC
```

```
## this will workaround the problem that in a VPC, instance hostnames are not resolvable
echo -e "127.0.0.1\tlocalhost.localdomain localhost" > /etc/hosts
echo -e ">:::1\tlocalhost6.localdomain6 localhost6" >> /etc/hosts
for (( i=1 ; i<255 ; i++ )); do
    echo -e "$SUBNET$\t\tip-{$SUBNET//./-}$i" ;
done >> /etc/hosts
```

```
EOF
```

7. For Production Instances

For a production instance, add the following line beneath the **USER_SCRIPT** line of the **User Data** field, to ensure security updates are applied on boot.

```
yum -y update
```



NOTE

yum -y update should be run regularly, to apply security fixes and enhancements.

8. Launch the Red Hat AMI instance.

Result

A clustered JBoss EAP 6 managed domain is configured and launched on a Red Hat AMI.

[Report a bug](#)

24.8.2. Launch One or More Instances to Serve as Cluster Host Controllers

Summary

This topic covers the steps required to launch one or more instances of JBoss EAP 6 to serve as cluster host controllers on a Red Hat AMI (Amazon Machine Image).

Prerequisites

- Configure and launch the cluster domain controller. Refer to [Section 24.8.1, “Launch an Instance to Serve as a Cluster Domain Controller”](#).
- [Section 24.6.7, “Configure IAM Setup”](#)
- [Section 24.6.9, “Configure S3 Bucket Setup”](#)

Procedure 24.15. Launch Host Controllers

For each instance you would like to create, repeat the following steps:

1. Select an AMI.
2. Define the desired number of instances (the number of slave host controllers).

3. Select the VPC and instance type.
4. Click on Security Group.
5. Ensure that all traffic from the JBoss EAP 6 cluster subnet is allowed.
6. Define other restrictions as desired.
7. Add the following into the User Data field:

```

## mod cluster proxy addresses
MOD_CLUSTER_PROXY_LIST=10.0.0.4:7654

## clustering setup
JBOSS_JGROUPS_S3_PING_SECRET_ACCESS_KEY=<your secret key>
JBOSS_JGROUPS_S3_PING_ACCESS_KEY=<your access key>
JBOSS_JGROUPS_S3_PING_BUCKET=<your bucket name>

## host controller setup
### static domain controller discovery setup
JBOSS_DOMAIN_MASTER_ADDRESS=10.0.0.5
### S3 domain controller discovery setup
# JBOSS_DOMAIN_S3_SECRET_ACCESS_KEY=<your secret key>
# JBOSS_DOMAIN_S3_ACCESS_KEY=<your access key>
# JBOSS_DOMAIN_S3_BUCKET=<your bucket name>

JBOSS_HOST_PASSWORD=<password for slave host controllers>

## database credentials configuration
JAVA_OPTS="$JAVA_OPTS -Ddb.host=instancename.something.rds.amazonaws.com -
Ddb.database=mydatabase -Ddb.user=<user> -Ddb.passwd=<pass>"

## subnet prefix this machine is connected to
SUBNET=10.0.1.

#### to run the example no modifications below should be needed ####
JBOSS_HOST_USERNAME=admin
PORTS_ALLOWED="1024:65535"
JBOSS_IP=`hostname | sed -e 's/ip-//' -e 'y/-/./'` #listen on public/private EC2 IP address

cat > $USER_SCRIPT << "EOF"
## Server instance configuration
sed -i "s/main-server-group/other-server-group/"
$JBOSS_CONFIG_DIR/$JBOSS_HOST_CONFIG

## install the JDBC driver as a core module
yum -y install mysql-connector-java
mkdir -p /usr/share/jbossas/modules/com/mysql/main
cp -v /usr/share/java/mysql-connector-java-*.jar
/usr/share/jbossas/modules/com/mysql/main/mysql-connector-java.jar

cat > /usr/share/jbossas/modules/com/mysql/main/module.xml <<"EOM"
<?xml version="1.0" encoding="UTF-8"?>
<module xmlns="urn:jboss:module:1.0" name="com.mysql">
  <resources>
    <resource-root path="mysql-connector-java.jar"/>

```



```

</resources>
<dependencies>
  <module name="javax.api"/>
</dependencies>
</module>
EOM

```

```

## this will workaround the problem that in a VPC, instance hostnames are not resolvable
echo -e "127.0.0.1\tlocalhost.localdomain localhost" > /etc/hosts
echo -e "::1\tlocalhost6.localdomain6 localhost6" >> /etc/hosts
for (( i=1 ; i<255 ; i++ )); do
  echo -e "$SUBNET$\t\tip-${SUBNET//./-}$i" ;
done >> /etc/hosts

EOF

```

8. For Production Instances

For a production instance, add the following line beneath the **USER_SCRIPT** line of the **User Data** field, to ensure security updates are applied on boot.

```
yum -y update
```



NOTE

yum -y update should be run regularly, to apply security fixes and enhancements.

9. Launch the Red Hat AMI instance.

Result

The JBoss EAP 6 cluster host controllers are configured and launched on a Red Hat AMI.

[Report a bug](#)

24.8.3. Test the Clustered JBoss EAP 6 Managed Domain

Summary

This topic covers the steps required to test the clustered JBoss EAP 6 managed domain on a Red Hat AMI (Amazon Machine Image).

To test the Managed Domain you must know the elastic IP addresses of both the Apache HTTP server and JBoss EAP 6 Domain Controller.

Prerequisites

- Configure and launch the cluster domain controller. See [Section 24.8.1, “Launch an Instance to Serve as a Cluster Domain Controller”](#) .
- Configure and launch the cluster host controllers. See [Section 24.8.2, “Launch One or More Instances to Serve as Cluster Host Controllers”](#) .

Procedure 24.16. Test the Apache HTTP server instance

- Navigate to **http://ELASTIC_IP_OF_APACHE_HTTP_SERVER** in a browser to confirm the web server is running successfully.

Procedure 24.17. Test the Domain Controller

1. Navigate to **http://ELASTIC_IP_OF_DOMAIN_CONTROLLER:9990/console**
2. Log in using the username **admin** and the password specified in the User Data field for the domain controller. Once logged in, the administration console landing page for a managed domain should appear (**http://ELASTIC_IP_OF_DOMAIN_CONTROLLER:9990/console/App.html#server-instances**).
3. Click the **Server** label at the top right side of the screen. Select any of the host controllers in the **Host** dropdown menu at the top left side of the screen.
4. Verify that this host controller has two server configurations called **server-one** and **server-two** and verify that they both belong to the **other-server-group**.

Procedure 24.18. Test the Host Controllers

1. Navigate to **http://ELASTIC_IP_OF_APACHE_HTTP_SERVER/cluster-demo/put.jsp** in a browser.
2. Verify that one of the host controllers logs the following message: **Putting date now**.
3. Stop the server instance that logged the message in the previous step (see *Stop a Server Using the Management Console*).
4. Navigate to **http://ELASTIC_IP_OF_APACHE_HTTP_SERVER/cluster-demo/get.jsp** in a browser.
5. Verify that the time shown is the same as the time that was **PUT** using **put.jsp** in Step 2.
6. Verify that one of the running server instances logs the following message: **Getting date now**.
7. Restart the stopped server instance (see [Section 2.3.2, "Start a Server Using the Management Console"](#))
8. Connect to the Apache HTTP server instance.

```
$ ssh -L7654:localhost:7654 ELASTIC_IP_OF_APACHE_HTTP_SERVER
```

9. Navigate to **http://localhost:7654/mod_cluster-manager** to confirm all instances are running correctly.

Result

The JBoss EAP 6 web server, domain controller, and host controllers are running correctly on a Red Hat AMI.

[Report a bug](#)

24.9. ESTABLISHING MONITORING WITH JBOSS OPERATIONS NETWORK (JON)

24.9.1. About AMI Monitoring

With your business application deployed to a correctly-configured AMI instance, the next step is to establish monitoring of the platform with JBoss Operations Network (JON).

The JON server is commonly located inside a corporate network, so it's necessary to establish a secure connection between the server and each of its agents. Establishing a VPN between the two points is the most common solution but this complicates the required networking configuration. This chapter provides network configuration guidelines for enabling communication between the JON agent and JON server. For more extensive information on configuration, management, and usage please refer to the official Red Hat documentation for JBoss Operations Network (JON).

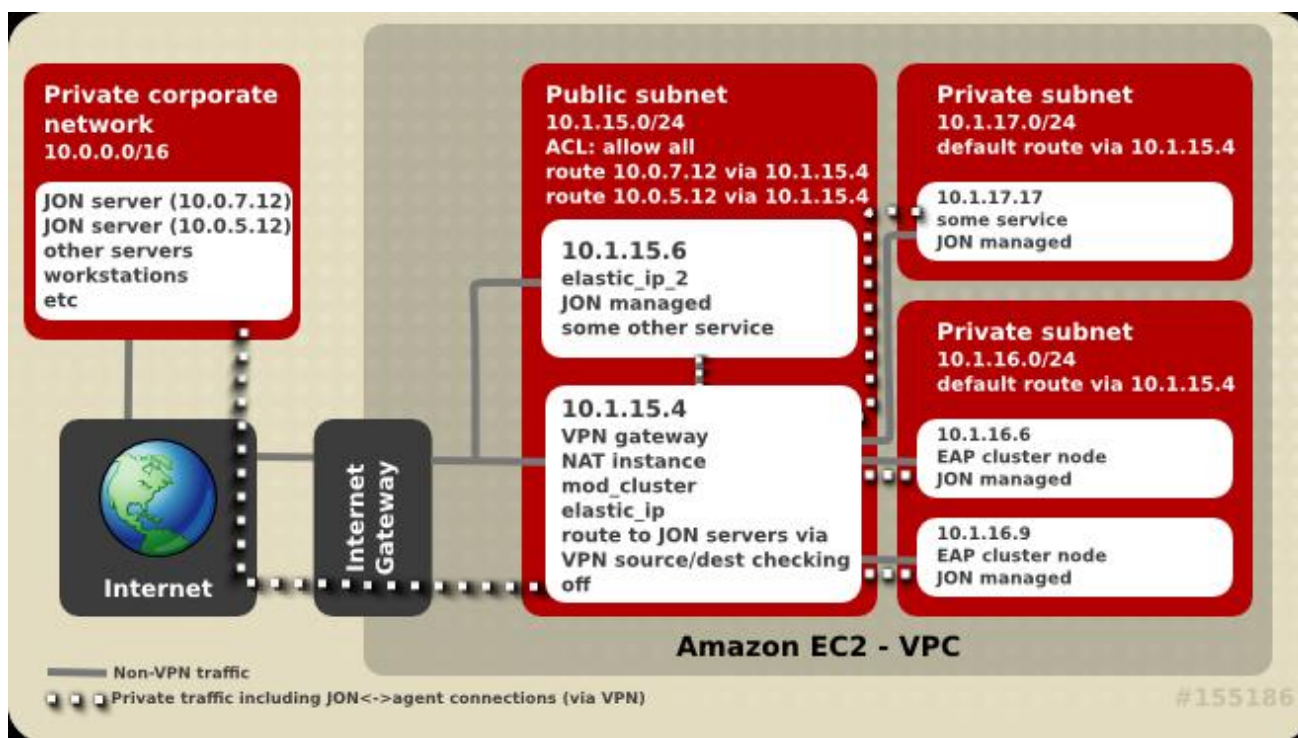


Figure 24.1. JON Server connectivity

[Report a bug](#)

24.9.2. About Connectivity Requirements

Registering a JON agent with its servers requires two-way communication between agent and servers. The JON Agent needs access to port **7080** on all JON servers, except in the case of SSL where port **7443** is used. Each JON server must be able to access each of the connected agents on a unique host and port pairing. The agent port is usually **16163**.

If there are multiple clustered JON servers, make sure each agent can communicate with all servers in the JON cluster via the IP and hostname pairs as configured through the JON server administration console. The JON server used by the agent to register may not be the server it tries to use after initialization.

[Report a bug](#)

24.9.3. About Network Address Translation (NAT)

A corporate VPN gateway acting in routed mode greatly simplifies network configuration. If your corporate VPN gateway is acting in NAT mode however, the JON server does not have direct visibility of agents. In this case, port forwarding needs to be configured for each agent.

NAT VPN configurations require a port on the gateway to be forwarded to the JON agent's address of port on the managed machine. The JON agent also needs to be configured to tell the server the forwarded port number and IP address. You can find further information in the **rhq.communications.connector.*** description for the **agent-configuration.xml** configuration file.

[Report a bug](#)

24.9.4. About Amazon EC2 and DNS

JON servers and JON agents need to be able to resolve each others' hostnames. The DNS resolution is more complicated in the case of a VPN configuration. Connected servers have multiple possible options. One option is to use either the Amazon EC2 or the corporate network's DNS servers. Another option is to use a split DNS configuration where the corporate DNS servers are used for resolving names in particular domains, and the Amazon EC2 DNS servers are used for resolving all other names.

[Report a bug](#)

24.9.5. About Routing in EC2

All Amazon EC2 servers have a **source/destination checking** routing feature activated by default. This feature drops any packets being sent to the server which have a destination different from the machine's IP address. If the VPN solution selected for connecting agents to the JON server includes a router, this feature needs to be turned off for the server or servers acting as routers or VPN gateways. This configuration setting can be accessed via the Amazon AWS console. Disabled **source/destination checking** is also required in a Virtual Private Cloud (VPC).

Some VPN configurations route general Internet traffic through the corporate VPN by default. It is recommended that you avoid this as it may be a slower and less efficient configuration for your particular needs.

While the use of a proper addressing schema is not a concern specific to JON, poor schemas can affect it. Amazon EC2 assigns IP addresses from the **10.0.0.0/8** network. Instances usually have a public IP address also, but only network traffic on the internal IP address within the same availability zone is free. To avoid using the **10.0.0.0/8** network in private addressing, there are a few things to consider.

- When creating a VPC, avoid allocating addresses already in use in the private network to avoid connectivity problems.
- If an instance needs access to availability zone local resources, make sure Amazon EC2 private addresses are used and traffic is not routed through the VPN.
- If an Amazon EC2 instance will access a small subset of corporate private network addresses (for example only JON servers), only these addresses should be routed through the VPN. This increases security and lowers the chance of Amazon EC2 or private network address space collisions.

[Report a bug](#)

24.9.6. About Terminating and Restarting with JON

One of the benefits of a cloud environment is the ease by which you can terminate and launch a machine instance. You can also launch an instance identical to the initial one. This may cause issues if the new

instance tries to register with JON servers using the same agent name as the previously running agent. If this happens the JON server will not allow an agent to reconnect with a missing or non-matching identification token.

To avoid this, ensure that terminated agents are removed from the JON inventory before trying to connect an agent with the same name or specify the correct identification token when starting new agent.

Another issue that you might encounter is when an agent machine is assigned a new VPN IP address that no longer matches the address recorded in the JON configuration. An example might include a machine that is restarted or where a VPN connection is terminated. In this case, it is recommended that you bind the JON agent's life cycle to the VPN connection's life cycle. If the connection drops, you can stop the agent. When the connection is restored again, update **JON_AGENT_ADDR** in **/etc/sysconfig/jon-agent-ec2** to reflect the new IP address and restart the agent.

Information on how to change the agent's IP address can be found in the Configuring JON Servers and Agents Guide available at https://access.redhat.com/documentation/en-US/JBoss_Operations_Network.

If there are a high number of instances launched and/or terminated it can become impractical to add and remove them manually from the JON inventory. JON's scripting capabilities can be used for automate these steps. Refer to the JON documentation for further information.

[Report a bug](#)

24.9.7. Configure an Instance to Register with JBoss Operations Network

Use the following procedure to register a JBoss EAP 6 instance with JBoss Operations Network.

- For JBoss EAP 6, add this to the User Data field.

```
JON_SERVER_ADDR=jon2.it.example.com
## if instance not already configured to resolve its hostname
JON_AGENT_ADDR=`ip addr show dev eth0 primary to 0/0 | sed -n 's#.*inet \([0-9.]\+\)/.*#\1#p`
PORTS_ALLOWED=16163
# insert other JON options when necessary.
```

See [Section 24.10.1, "Permanent Configuration Parameters"](#), parameters starting with **JON_** for the format of JON options.

[Report a bug](#)

24.10. USER SCRIPT CONFIGURATION

24.10.1. Permanent Configuration Parameters

Summary

The following parameters can be used to influence the configuration and operation of JBoss EAP 6. Their contents are written to **/etc/sysconfig/jbossas** and **/etc/sysconfig/jon-agent-ec2**.

Table 24.2. Configurable Parameters

Name	Description	Default
JBOSS_JGROUPS_S3_PING_ACCESS_KEY	Amazon AWS user account access key for S3_PING discovery if clustering is used.	N/A
JBOSS_JGROUPS_S3_PING_SECRET_ACCESS_KEY	Amazon AWS user account secret access key.	N/A
JBOSS_JGROUPS_S3_PING_BUCKET	Amazon S3 bucket to be used for S3_PING discovery.	N/A
JBOSS_CLUSTER_ID	<p>ID of cluster member nodes. Only used for clustering. Accepted values are (in order):</p> <ul style="list-style-type: none"> • A valid cluster ID number in the range 0 - 1023. • A network interface name, where the last octet of the IP is used as the value. • "S3" as a value would coordinate ID usage through the S3 bucket used by jgroups' S3_PING. <p>It is recommended to use the last octet of the IP (the default) when all cluster nodes are located in the same 24 or more bit subnet (for example, in a VPC subnet).</p>	Last octet of eth0's IP address
MOD_CLUSTER_PROXY_LIST	Comma-delimited list of IPs/hostnames of mod_cluster proxies if mod_cluster is to be used.	N/A
PORTS_ALLOWED	List of incoming ports to be allowed by firewall in addition to the default ones.	N/A
JBOSSAS_ADMIN_PASSWORD	Password for the admin user.	N/A
JON_SERVER_ADDR	JON server hostname or IP with which to register. This is only used for registration, after that the agent may communicate with other servers in the JON cluster.	N/A

Name	Description	Default
JON_SERVER_PORT	Port used by the agent to communicate with the server.	7080
JON_AGENT_NAME	Name of JON agent, must be unique.	Instance's ID
JON_AGENT_PORT	Port that the agent listens on.	16163
JON_AGENT_ADDR	IP address that the JON agent is to be bound to. This is used when the server has more than one public address, (e.g. VPN).	JON agent chooses the IP of local hostname by default.
JON_AGENT_OPTS	Additional JON agent system properties which can be used for configuring SSL, NAT and other advanced settings.	N/A
JBOSS_SERVER_CONFIG	Name of JBoss EAP 6 server configuration file to use. If JBOSS_DOMAIN_CONTROLLER=true, then domain-ec2.xml is used. Otherwise: <ul style="list-style-type: none"> • If S3 config is present, then standalone-ec2-ha.xml is used. • If MOD_CLUSTER_PROXY_LIST is specified, then standalone-mod_cluster-ec2-ha.xml is selected. • If neither of the first two options are used, then the standalone.xml file is used. • Can also be set to standalone-full.xml. 	standalone.xml, standalone-full.xml, standalone-ec2-ha.xml, standalone-mod_cluster-ec2-ha.xml, domain-ec2.xml depending on the other parameters.
JAVA_OPTS	Custom values to be added to the variable before JBoss EAP 6 starts.	JAVA_OPTS is built from the values of other parameters.
JBOSS_IP	IP address that the server is to be bound to.	127.0.0.1

Name	Description	Default
JBOSSCONF	The name of the JBoss EAP 6 profile to start. To prevent JBoss EAP 6 from starting, JBOSSCONF can be set to disabled	standalone
JBOSS_DOMAIN_CONTROLLER	Sets whether or not this instance will run as a domain controller.	false
JBOSS_DOMAIN_MASTER_ADDRESS	IP address of remote domain controller.	N/A
JBOSS_HOST_NAME	The logical host name (within the domain). This needs to be distinct.	The value of the HOSTNAME environment variable.
JBOSS_HOST_USERNAME	The username the host controller should use when registering with the domain controller. If not provided, the JBOSS_HOST_NAME is used instead.	JBOSS_HOST_NAME
JBOSS_HOST_PASSWORD	The password the host controller should use when registering with the domain controller.	N/A
JBOSS_HOST_CONFIG	If JBOSS_DOMAIN_CONTROLLER=true, then host-master.xml is used. If JBOSS_DOMAIN_MASTER_ADDRESS is present, then host-slave.xml is used.	host-master.xml or host-slave.xml , depending on the other parameters.
JBOSS_DOMAIN_S3_ACCESS_KEY	Amazon AWS user account access key for S3 domain controller discovery.	N/A
JBOSS_DOMAIN_S3_SECRET_ACCESS_KEY	Amazon AWS user account secret access key for S3 domain controller discovery.	N/A
JBOSS_DOMAIN_S3_BUCKET	Amazon S3 bucket to be used for S3 domain controller discovery.	N/A

[Report a bug](#)

24.10.2. Custom Script Parameters

Summary

The following parameters can be used in the user customization section of the **User Data:** field.

Table 24.3. Configurable Parameters

Name	Description
JBOSS_DEPLOY_DIR	Deploy directory of the active profile (for example, /var/lib/jbossas/standalone/deployments/). Deployable archives placed in this directory will be deployed. Red Hat recommends using the Management Console or CLI tool to manage deployments instead of using the deploy directory.
JBOSS_CONFIG_DIR	Config directory of the active profile (for example, /var/lib/jbossas/standalone/configuration/).
JBOSS_HOST_CONFIG	Name of the active host configuration file (for example, host-master.xml). Red Hat recommends using the Management Console or CLI tools to configure the server instead of editing the configuration file.
JBOSS_SERVER_CONFIG	Name of the active server configuration file (for example, standalone-ec2-ha.xml). Red Hat recommends using the Management Console or CLI tools to configure the server instead of editing the configuration file.
USER_SCRIPT	Path to the custom configuration script, which is available prior to sourcing user-data configuration.
USER_CLI_COMMANDS	Path to a custom file of CLI commands, which is available prior to sourcing user-data configuration.

[Report a bug](#)

24.11. TROUBLESHOOTING

24.11.1. About Troubleshooting Amazon EC2

EC2 provides an Alarm Status for each instance, indicating severe instance malfunction but the absence of such an alarm is no guarantee that the instance has started correctly and services are running properly. It is possible to use Amazon CloudWatch with its custom metric functionality to monitor instance services' health but use of an enterprise management solution is recommended.

To simplify troubleshooting, Red Hat recommends managing the EC2 instance using JBoss Operations Network (JON) which can automatically discover, monitor and manage many services on an EC2 instance with the JON agent installed, including: JBoss EAP 6, Tomcat, Apache HTTP Server, and PostgreSQL. For details of monitoring JBoss EAP with JON, see [Section 24.9.1, "About AMI Monitoring"](#).

[Report a bug](#)

24.11.2. Diagnostic Information

In case of a problem being detected by the JBoss Operations Network, Amazon CloudWatch or manual inspection, common sources of diagnostic information are:

- **/var/log/jboss_user-data.out** is the output of the `jboss-ec2-eap` init script and user custom configuration script.

- **/var/cache/jboss-ec2-eap/** contains the actual user data, custom script, and custom CLI commands used at instance start-up.
- **/var/log** also contains all the logs collected from machine start up, JBoss EAP 6, httpd and most other services.

Access to these files is only available via an SSH session. Refer to the Amazon EC Getting Started Guide for details on how to configure and establish an SSH session with an Amazon EC2 instance.

[Report a bug](#)

CHAPTER 25. EXTERNALIZE SESSIONS

25.1. EXTERNALIZE HTTP SESSION FROM JBOSS EAP TO JBOSS DATA GRID

You can use Red Hat JBoss Data Grid as an external cache container for application specific data, such as HTTP sessions, in Red Hat JBoss Enterprise Application Platform (JBoss EAP) 6.4 and later. This allows scaling of the data layer independent of the application, and enables different EAP clusters, that may reside in various domains, to access data from the same JBoss Data Grid cluster. Additionally, other applications can interface with the caches presented by Red Hat JBoss Data Grid.

The below procedure applies for both standalone and domain mode of EAP; however, in domain mode each server group requires a unique remote cache configured. While multiple server groups can utilize the same Red Hat JBoss Data Grid cluster the respective remote caches will be unique to the EAP server group.



NOTE

For each distributable application, an entirely new cache must be created. It can be created in an existing cache container, for example, web.

Procedure 25.1. Externalize HTTP Sessions

1. Ensure the remote cache containers are defined in EAP's **infinispan** subsystem; in the example below the **cache** attribute in the **remote-store** element defines the cache name on the remote JBoss Data Grid server:

```
<subsystem xmlns="urn:jboss:domain:infinispan:4.0">
  [...]
  <cache-container name="web" default-cache="dist"
    module="org.jboss.as.clustering.web.infinispan" statistics-enabled="true">
    <transport lock-timeout="60000"/>
    <invalidation-cache name="jdg" mode="SYNC">
      <locking isolation="REPEATABLE_READ"/>
      <transaction mode="BATCH"/>
      <remote-store remote-servers="remote-jdg-server1 remote-jdg-server2"
        cache="default" socket-timeout="60000"
        preload="true" passivation="false" purge="false" shared="true"/>
    </replicated-cache>
  </cache-container>
</subsystem>
```

2. Define the location of the remote Red Hat JBoss Data Grid server by adding the networking information to the **socket-binding-group**:

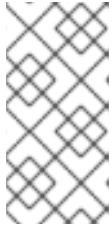
```
<socket-binding-group ...>
  <outbound-socket-binding name="remote-jdg-server1">
    <remote-destination host="JDGHostName1" port="11222"/>
  </outbound-socket-binding>
  <outbound-socket-binding name="remote-jdg-server2">
    <remote-destination host="JDGHostName2" port="11222"/>
  </outbound-socket-binding>
</socket-binding-group>
```

3. Repeat the above steps for each cache-container and each Red Hat JBoss Data Grid server. Each server defined must have a separate **<outbound-socket-binding>** element defined.
4. Add passivation and cache information into the application's **jboss-web.xml**. In the following example **web** is the name of the cache container, and **jdg** is the name of the default cache located in this container. An example file is shown below:

```
<?xml version="1.0" encoding="UTF-8"?>
<jboss-web xmlns="http://www.jboss.com/xml/ns/javaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.jboss.com/xml/ns/javaee
http://www.jboss.org/j2ee/schema/jboss-web_10_0.xsd"
  version="10.0">

  <replication-config>
    <replication-granularity>SESSION</replication-granularity>
    <cache-name>web.jdg</cache-name>
  </replication-config>

</jboss-web>
```



NOTE

The passivation timeouts above are provided assuming that a typical session is abandoned within 15 minutes and uses the default HTTP session timeout in JBoss EAP of 30 minutes. These values may need to be adjusted based on each application's workload.

[Report a bug](#)

APPENDIX A. SUPPLEMENTAL REFERENCES

A.1. DOWNLOAD FILES FROM THE RED HAT CUSTOMER PORTAL

Prerequisites

- Before you begin this task, you need a Customer Portal account. Browse to <https://access.redhat.com> and click the **Register** link in the upper right corner to create an account.

Procedure A.1. Log in and Download Files from the Red Hat Customer Portal

1. Browse to <https://access.redhat.com> and click the **Log in** link in the top right corner. Enter your credentials and click **Log In**.

Result

You are logged into RHN and you are returned to the main web page at <https://access.redhat.com>.

2. **Navigate to the Downloads page.**

Use one of the following options to navigate to the **Downloads** page.

- Click the **Downloads** link in the top navigation bar.
- Navigate directly to <https://access.redhat.com/downloads/>.

3. **Select the product and version to download.**

Use one of the following ways to choose the correct product and version to download.

- Step through the navigation one level at a time.
- Search for your product using the search area at the top right-hand side of the screen.

4. **Download the appropriate file for your operating system and installation method of choice.**

Depending on the product you choose, you may have the choice of a Zip archive, RPM, or native installer for a specific operating system and architecture. Click either the file name or the **Download** link to the right of the file you want to download.

Result

The file is downloaded to your computer.

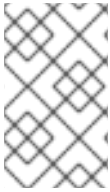
[Report a bug](#)

A.2. CONFIGURE THE DEFAULT JAVA DEVELOPMENT KIT ON RED HAT ENTERPRISE LINUX

It is possible to have multiple Java development kits installed on your Red Hat Enterprise Linux system. This task shows you how to specify which one your current environment uses. It uses the **alternatives** command.

**IMPORTANT**

This task only applies to Red Hat Enterprise Linux.

**NOTE**

It may not be necessary to do this step. Red Hat Enterprise Linux uses OpenJDK 1.6 as the default option. If this is what you want, and your system is working properly, you do not need to manually specify which Java development kit to use.

Prerequisites

- In order to complete this task, you need to have superuser access, either through direct login or by means of the **sudo** command.

Procedure A.2. Configure the Default Java Development Kit

1. **Determine the paths for your preferred `java` and `javac` binaries.**

You can use the command `rpm -ql packagename |grep bin` to find the locations of binaries installed from RPMs. The default locations of the `java` and `javac` binaries on Red Hat Enterprise Linux 32-bit systems are as follows.

Table A.1. Default locations for `java` and `javac` binaries

Java Development Kit	Path
OpenJDK 1.6	<code>/usr/lib/jvm/jre-1.6.0-openjdk/bin/java</code> <code>/usr/lib/jvm/java-1.6.0-openjdk/bin/javac</code>
Oracle JDK 1.6	<code>/usr/lib/jvm/jre-1.6.0-sun/bin/java</code> <code>/usr/lib/jvm/java-1.6.0-sun/bin/javac</code>

2. **Set the alternative you wish to use for each.**

Run the following commands to set your system to use a specific `java` and `javac`:
`/usr/sbin/alternatives --config java` or `/usr/sbin/alternatives --config javac`. Follow the on-screen instructions.

3. **Optional: Set the `java_sdk_1.6.0` alternative choice.**

If you want to use Oracle JDK, you need to set the alternative for `java_sdk_1.6.0`. as well. Use the following command: `/usr/sbin/alternatives --config java_sdk_1.6.0`. The correct path is usually `/usr/lib/jvm/java-1.6.0-sun`. You can do a file listing to verify it.

Result:

The alternative Java Development Kit is selected and active.

[Report a bug](#)

A.3. MANAGEMENT INTERFACE AUDIT LOGGING REFERENCE**Logger Attributes Reference**

In addition to enabling or disabling management interface audit logging, the following **logger** configuration attributes are available.

log-boot

If set to **true**, management operations when booting the server are included in the audit log, **false** otherwise. Default: **true**.

log-read-only

If set to **true**, all operations will be audit logged. If set to **false** only operations that change the model will be logged. Default: **false**.

Log Formatter Attributes Reference

The formatter specifies the format of the log entries. Only one formatter is available, which outputs log entries in JSON format.

Example A.1. Include the timestamp in the log records

```
/core-service=management/access=audit/json-formatter=json-formatter:write-attribute(name=include-date,value=true)
```

Log Formatter Attributes

include-date

A boolean value which defines whether or not the timestamp is included in the formatted log records. Default: **true**.

date-separator

A string containing characters to be used to separate the date and the rest of the formatted log message. This is ignored if **include-date=false**. Default: `–` (This is a space, followed by a hyphen, then a space).

date-format

The date format to use for the timestamp as understood by `java.text.SimpleDateFormat`. Ignored if **include-date=false**. Default: **yyyy-MM-dd HH:mm:ss**.

compact

If **true** it will format the JSON on one line. There may still be values containing new lines, so if having the whole record on one line is important, set **escape-new-line** or **escape-control-characters** to **true**. Default: **false**.

escape-control-characters

If **true** it will escape all control characters (ASCII entries with a decimal value < 32) with the ASCII code in octal; for example, a new line becomes **#012**. If this is **true**, it will override **escape-new-line=false**. Default: **false**.

escape-new-line

If **true** it will escape all new lines with the ASCII code in octal; for example **#012**. Default: **false**.

Management Interface Audit Log File Handler Attributes Reference

A file handler specifies the parameters by which audit log records are output to a file. Specifically it defines the formatter, file name and path for the file.

File Handler Attributes

formatter

The name of a JSON formatter to use to format the log records. Default: **json-formatter**.

path

The path of the audit log file. Default: **audit-log.log**.

relative-to

The name of another previously named path, or of one of the standard paths provided by the system. If **relative-to** is provided, the value of the path attribute is treated as relative to the path specified by this attribute. Default: **jboss.server.data.dir**.

failure-count

The number of logging failures since the handler was initialized. Default: 0.

max-failure-count

The maximum number of logging failures before disabling this handler. Default: 10.

disabled-due-to-failure

Takes the value **true** if this handler was disabled due to logging failures. Default: **false**.

Management Interface Syslog Handler Attributes Reference

A syslog handler specifies the parameters by which audit log entries are sent to a syslog server, specifically the syslog server's hostname and the port on which the syslog server is listening.

Sending audit logging to a syslog server provides more security options than logging to a local file or local syslog server. Multiple syslog handlers can be defined and be active at the same time.

Syslog servers vary in their implementation, so not all settings are applicable to all syslog servers. Testing has been conducted using the *rsyslog* syslog implementation.

The *Syslog Handler Attributes* lists only the high-level attributes. Each attribute has configuration parameters, and some have child configuration parameters. To detail of a syslog handler's attributes, run the following command.

```
/core-service=management/access=audit/syslog-handler=mysyslog:read-resource-description(recursive=true)
```

Syslog Handler Attributes

app-name

The application name to add to the syslog records as defined in section 6.2.5 of RFC-5424. If not specified it will default to the name of the product.

disabled-due-to-failure

Takes the value **true** if this handler was disabled due to logging failures. Default: **false**.

facility

The facility to use for syslog logging as defined in section 6.2.1 of RFC-5424, and section 4.1.1 of RFC-3164.

failure-count

The number of logging failures since the handler was initialized. Default: **0**.

formatter

The name of the formatter to use to format the log records. Default: **json-formatter**.

max-failure-count

The maximum number of logging failures before disabling this handler. Default: **10**.

max-length

The maximum length of a log message (in bytes), including the header. If undefined, it will default to **1024** bytes if the **syslog-format** is **RFC3164**, or **2048** bytes if the **syslog-format** is **RFC5424**.

protocol

The protocol to use for the syslog handler. Must be one and only one of **udp**, **tcp** or **tls**.

reconnect-timeout

Available from JBoss EAP 6.4. The number of seconds to wait before attempting to reconnect to the syslog server, in the event connectivity is lost. Default: **-1** (Disabled).

syslog-format

Syslog format: *RFC-5424* or *RFC-3164*. Default: **RFC-5424**.

truncate

Whether or not a message, including the header, should be truncated if the length in bytes is greater than the value of the **max-length** attribute. If set to **false** messages will be split and sent with the same header values. Default: **false**.

[Report a bug](#)

APPENDIX B. REVISION HISTORY

Revision 6.4.0-47

Thursday November 16 2017

Red Hat Customer Content Services

Red Hat JBoss Enterprise Application Platform 6.4.0.GA Continuous Release