Red Hat JBoss Enterprise Application Platform 7.0

Configuration Guide

For Use with Red Hat JBoss Enterprise Application Platform 7.0

Last Updated: 2018-02-08

Red Hat JBoss Enterprise Application Platform 7.0 Configuration Guide

For Use with Red Hat JBoss Enterprise Application Platform 7.0

Legal Notice

Abstract

This document provides a practical guide for administrators to configure Red Hat JBoss Enterprise Application Platform 7.0.

Table of Contents

CHAPTER 1. OVERVIEW

The purpose of this guide is to cover many of the configuration tasks needed for setting up and maintaining JBoss EAP as well as running applications and other services on it. Before using this guide to configure JBoss EAP, it is assumed that the latest version of JBoss EAP has been downloaded and installed. For installation instructions, see the JBoss EAP *Installation Guide*.

IMPORTANT

Since the installation location of JBoss EAP will vary between host machines, this guide refers to the installation location as **EAP_HOME**. The actual location of the JBoss EAP installation should be used instead of **EAP_HOME** when performing administrative tasks.

CHAPTER 2. STARTING AND STOPPING JBOSS EAP

2.1. STARTING JBOSS EAP

JBoss EAP runs in one of two operating modes: as a standalone server or in a managed domain, and is supported on several platforms: Red Hat Enterprise Linux, Windows Server, Oracle Solaris, and Hewlett-Packard HP-UX.

The specific command to start JBoss EAP depends on the underlying platform and the desired operating mode.

Start JBoss EAP as a Standalone Server

```
$ EAP_HOME/bin/standalone.sh
```

NOTE

For Windows Server, use the **`EAP_HOME\bin\standalone.bat`** script.

This startup script uses the **`EAP_HOME/bin/standalone.conf`** file (or **`standalone.conf.bat`** for Windows Server) to set some default preferences, such as JVM options. You can customize the settings in this file.

JBoss EAP uses the **`standalone.xml`** configuration file by default, but can be started using a different one. For details on the available standalone configuration files and how to use them, see the Standalone Server Configuration Files section.

For a complete listing of all available startup script arguments and their purposes, use the **`--help`** argument or see the Server Runtime Arguments section.

Start JBoss EAP in a Managed Domain

The domain controller must be started before the servers in any of the server groups in the domain. Use this script to first start the domain controller, and then for each associated host controller.

```
$ EAP_HOME/bin/domain.sh
```

NOTE

For Windows Server, use the **`EAP_HOME\bin\domain.bat`** script.

This startup script uses the **`EAP_HOME/bin/domain.conf`** file (or **`domain.conf.bat`** for Windows Server) to set some default preferences, such as JVM options. You can customize the settings in this file.

JBoss EAP uses the **`host.xml`** host configuration file by default, but can be started using a different one. For details on the available managed domain configuration files and how to use them, see the Managed Domain Configuration Files section.

When setting up a managed domain, additional arguments will need to be passed into the startup script. For a complete listing of all available startup script arguments and their purposes, use the **`--help`** argument or see the Server Runtime Arguments section.

2.2. STOPPING JBOSS EAP

The way that you stop JBoss EAP depends on how it was started.

Stop an Interactive Instance of JBoss EAP

Press **Ctrl+C** in the terminal where JBoss EAP was started.

Stop a Background Instance of JBoss EAP

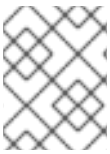
Use the management CLI to connect to the running instance and shut down the server.

1. Launch the management CLI.

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

2. Issue the **shutdown** command.

```
shutdown
```



NOTE

When running in a managed domain, you must specify the host name to shut down by using the **--host** argument with the **shutdown** command.

2.3. RUNNING JBOSS EAP IN ADMIN-ONLY MODE

JBoss EAP has the ability to be started in *admin-only* mode. This enables JBoss EAP to run and accept management requests but not start other runtime services or accept end user requests. Admin-only mode is available in both standalone servers as well as managed domains. In a managed domain, if a domain controller is started in admin-only mode, it will not accept incoming connections from slave host controllers.

To start a JBoss EAP instance in admin-only mode, use the **--admin-only** runtime switch when starting the JBoss EAP instance.

NOTE

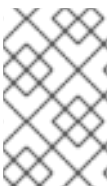
The management CLI commands shown assume that you are running a JBoss EAP standalone server. For more details on using the management CLI for a JBoss EAP managed domain, please see the JBoss EAP *Management CLI Guide*.

Start JBoss EAP in Admin-Only Mode

```
$ EAP_HOME/bin/standalone.sh --admin-only
```

Check If JBoss EAP is Running in Admin-Only Mode

To determine if a JBoss EAP instance is running in admin-only mode:



NOTE

The management CLI commands shown assume that you are running a JBoss EAP standalone server. For more details on using the management CLI for a JBoss EAP managed domain, please see the JBoss EAP *Management CLI Guide*.

```
:read-attribute(name=running-mode)
```

If the JBoss EAP instance is running in admin-only mode, the result will be:

```
{
  "outcome" => "success",
  "result" => "ADMIN_ONLY"
}
```

otherwise, the result will be:

```
{
  "outcome" => "success",
  "result" => "NORMAL"
}
```

Restart in a Different Mode from the Management CLI

In addition to stopping and starting a JBoss EAP instance with a different runtime switch, the management CLI may also be used to reload the server and start it in a different mode. To reload a JBoss EAP instance to start in admin-only mode:

NOTE

The management CLI commands shown assume that you are running a JBoss EAP standalone server. For more details on using the management CLI for a JBoss EAP managed domain, please see the JBoss EAP *Management CLI Guide*.

```
reload --admin-only=true
```

To reload a JBoss EAP instance to start in normal mode:

```
reload --admin-only=false
```

NOTE

Separate from the current running mode, the initial running mode may also be checked with the following command: **`/core-service=server-environment:read-attribute(name=initial-running-mode)`**. This command differs from **`:read-attribute(name=running-mode)`** by displaying the running mode in which JBoss EAP was launched and *NOT* its current running mode.

2.4. SUSPEND AND SHUT DOWN JBOSS EAP GRACEFULLY

JBoss EAP can be suspended or shut down gracefully. This allows active requests to complete normally, without accepting any new requests. A timeout value specifies how long that the suspend or shut down operation will wait for active requests to complete. While the server is suspended, management requests are still processed.

Graceful shutdown is coordinated at a server-wide level, mostly focused on the entry points at which a request enters the server. The following subsystems support graceful shutdown:

Undertow

The **undertow** subsystem will wait for all requests to finish.

mod_cluster

The **modcluster** subsystem will notify the load balancer that the server is suspending in the **PRE_SUSPEND** phase.

EJB

The **ejb3** subsystem will wait for all remote EJB requests and MDB message deliveries to finish. Delivery to MDBs is stopped in the **PRE_SUSPEND** phase. EJB timers are suspended, and missed timers will be activated when the server is resumed.

EE Concurrency

The server will wait for all active jobs to finish. All queued jobs will be skipped. Currently, since EE Concurrency does not have persistence, those queued jobs that were skipped will be lost.
While the server is in a suspended state, scheduled tasks will continue to execute at their scheduled times but will throw a **java.lang.IllegalStateException**. Once the server is resumed, scheduled tasks will continue to execute normally and in most cases, tasks will not need to be rescheduled.

Batch

The server will stop all running jobs within the timeout period and defer all scheduled jobs.

NOTE

Graceful shutdown currently will not reject inbound remote distributed transactions or new inbound JMS messages. EE batch jobs and EE concurrency tasks scheduled by inflight activity are currently allowed to proceed. However, EE concurrency tasks submitted that pass the timeout window currently error when executed.

Requests are tracked by the **request-controller** subsystem. Without this subsystem, suspend and resume capabilities are limited and the server will not wait for requests to complete before suspending or shutting down. However, if you do not need this capability, the **request-controller** subsystem can be removed for a small performance improvement.

2.4.1. Suspend Servers

JBoss EAP 7 introduced a *suspend* mode, which suspends server operations gracefully. This allows all active requests to complete normally, but will not accept any new requests. Once the server has been suspended, it can be shut down, returned back to a running state, or left in a suspended state to perform maintenance.

NOTE

The management interfaces are not impacted by suspending the server.

The server can be suspended and resumed using the management console or the management CLI.

Check the Server Suspend State
The server suspend state can be viewed using the following management CLI commands. The resulting value will be one of **RUNNING**, **PRE_SUSPEND**, **SUSPENDING**, or **SUSPENDED**.

- Check the suspend state for a standalone server.

```
:read-attribute(name=suspend-state)
```

–

- Check the suspend state for a server in a managed domain.

```
/host=master/server=server-one:read-attribute(name=suspend-state)
```

Suspend
Use the following management CLI commands to suspend the server, specifying the timeout value, in seconds, for the server to wait for active requests to complete. The default is **0**, which will suspend immediately. A value of **-1** will cause the server to wait indefinitely for all active requests to complete.

Each example waits up to 60 seconds for requests to complete before suspending.

- Suspend a standalone server.

```
:suspend(timeout=60)
```

- Suspend all servers in a managed domain.

```
:suspend-servers(timeout=60)
```

- Suspend a single server in a managed domain.

```
/host=master/server-config=server-one:suspend(timeout=60)
```

- Suspend all servers in a server group.

```
/server-group=main-server-group:suspend-servers(timeout=60)
```

Resume
The server can be returned back to a normal running state to accept new requests by using the **resume** command at the appropriate level (server, server group, entire domain). For example:

```
:resume
```

2.4.2. Shut Down Servers Gracefully

A server will be shut down gracefully if an appropriate timeout value is specified when stopping the server. Once the command is issued, the server will be suspended and will wait up to the specified timeout for all requests to finish before shutting down.

Use the following management CLI commands to shut down the server gracefully. Specify the timeout value, in seconds, for the server to wait for active requests to complete. The default is **0**, which will shut down the server immediately. A value of **-1** will cause the server to wait indefinitely for all active requests to complete before shutting down.

Each example waits up to 60 seconds for requests to complete before shutting down.

- Shut down a standalone server gracefully.

```
:shutdown(timeout=60)
```

- Stop all servers in a managed domain gracefully.

```
:stop-servers(timeout=60)
```

- Stop a single server in a managed domain gracefully.

```
/host=master/server-config=server-one:stop(timeout=60)
```

- Stop all servers in a server group gracefully.

```
/server-group=main-server-group:stop-servers(timeout=60)
```

2.5. STARTING AND STOPPING JBOSS EAP (RPM INSTALLATION)

Starting and stopping JBoss EAP is different for an RPM installation compared to a ZIP or installer installation.

2.5.1. Starting JBoss EAP (RPM Installation)

The command for starting an RPM installation of JBoss EAP depends on which operating mode you want to start (a standalone server or a managed domain), and which Red Hat Enterprise Linux version you are running.

Start JBoss EAP as a Standalone Server (RPM Installation)

- For Red Hat Enterprise Linux 6:

```
$ service eap7-standalone start
```

- For Red Hat Enterprise Linux 7:

```
$ systemctl start eap7-standalone.service
```

This will start JBoss EAP using the **`standalone.xml`** configuration file by default. You can start JBoss EAP with a different standalone server configuration file by setting a property in the RPM service configuration file. For more information, see the Configure RPM Service Properties section below.

Start JBoss EAP in a Managed Domain (RPM Installation)

- For Red Hat Enterprise Linux 6:

```
$ service eap7-domain start
```

- For Red Hat Enterprise Linux 7:

```
$ systemctl start eap7-domain.service
```

This will start JBoss EAP using the **`host.xml`** configuration file by default. You can start JBoss EAP with a different managed domain configuration file by setting a property in the RPM service configuration file. For more information, see the Configure RPM Service Properties section below.

Configure RPM Service Properties
This section shows you how to configure the RPM service properties and other startup options for your JBoss EAP installation. Note that it is recommended to back up your configuration files before making modifications.

For a listing of all available startup options for an RPM installation, see the RPM Service Configuration Properties section.

IMPORTANT

For Red Hat Enterprise Linux 7, RPM service configuration files are loaded using **systemd**, so variable expressions are not expanded.

- Specify the server configuration file.
  When starting a standalone server, the **standalone.xml** file is used by default. When running in a managed domain, the **host.xml** file is used by default. You can start JBoss EAP with a different configuration file by setting the **WILDFLY_SERVER_CONFIG** property in the appropriate RPM configuration file, for example, **eap7-standalone.conf**.

```
WILDFLY_SERVER_CONFIG=standalone-full.xml
```

- Bind to a specific IP address.
  By default, a JBoss EAP RPM installation binds to **0.0.0.0**. You can bind JBoss EAP to a specific IP address by setting the **WILDFLY_BIND** property in the appropriate RPM configuration file, for example, **eap7-standalone.conf**.

```
WILDFLY_BIND=192.168.0.1
```

NOTE

If you want to bind the management interfaces to a specific IP address, this can be configured in the JBoss EAP startup configuration file as shown in the next example.

- Set JVM options or Java properties.
  You can specify JVM options or Java properties to pass into the JBoss EAP startup script by editing the startup configuration file. This file is **EAP_HOME/bin/standalone.conf** for a standalone server or **EAP_HOME/bin/domain.conf** for a managed domain. The below example configures the heap size and binds the JBoss EAP management interfaces to an IP address.

```
JAVA_OPTS="$JAVA_OPTS -Xms2048m -Xmx2048m"
JAVA_OPTS="$JAVA_OPTS -Djboss.bind.address.management=192.168.0.1"
```

NOTE

If required, the JBoss EAP bind address must be configured using the **WILDFLY_BIND** property and not using the **jboss.bind.address** standard property here.

NOTE

If a property has the same name in both the RPM service configuration file (for example, **/etc/sysconfig/eap7-standalone**) and in the JBoss EAP startup configuration file (for example, **EAP_HOME/bin/standalone.conf**), the value that takes precedence is the one in the JBoss EAP startup configuration file. One such property is **JAVA_HOME**.

2.5.2. Stopping JBoss EAP (RPM Installation)

The command for stopping an RPM installation of JBoss EAP depends on which operating mode that was started (a standalone server or a managed domain), and which Red Hat Enterprise Linux version you are running.

Stop JBoss EAP as a Standalone Server (RPM Installation)

- For Red Hat Enterprise Linux 6:

```
$ service eap7-standalone stop
```

- For Red Hat Enterprise Linux 7:

```
$ systemctl stop eap7-standalone.service
```

Stop JBoss EAP in a Managed Domain (RPM Installation)

- For Red Hat Enterprise Linux 6:

```
$ service eap7-domain stop
```

- For Red Hat Enterprise Linux 7:

```
$ systemctl stop eap7-domain.service
```

For a listing of all available startup options for an RPM installation, see the RPM Service Configuration Files section.

2.6. POWERSHELL SCRIPTS (WINDOWS SERVER)

IMPORTANT

This feature is provided as Technology Preview only. It is not supported for use in a production environment, and it may be subject to significant future changes. See Technology Preview Features Support Scope on the Red Hat Customer Portal for information about the support scope for Technology Preview features.

JBoss EAP includes PowerShell script equivalents for most of the JBoss EAP management scripts. This includes a PowerShell script to start JBoss EAP on Microsoft Windows Server.

The JBoss EAP PowerShell scripts are designed to work with PowerShell version 2 and newer running on tested versions of Windows Server.

The JBoss EAP PowerShell scripts are located in ***EAP_HOME*\bin**, and are used in mostly the same way as the JBoss EAP batch scripts.

For example, to start a standalone JBoss EAP server with the **standalone-full.xml** configuration file, use the following PowerShell command:

```
.\standalone.ps1 "-c=standalone-full.xml"
```

NOTE

Arguments of the JBoss EAP PowerShell scripts must be in quotes.

CHAPTER 3. JBOSS EAP MANAGEMENT

JBoss EAP uses a simplified configuration, with one configuration file per standalone server or managed domain. Default configuration for a standalone server is stored in the **`EAP_HOME/standalone/configuration/standalone.xml`** file and default configuration for a managed domain is stored in the **`EAP_HOME/domain/configuration/domain.xml`** file. Additionally, the default configuration for a host controller is stored in the **`EAP_HOME/domain/configuration/host.xml`** file.

JBoss EAP can be configured using the command-line management CLI, web-based management console, Java API, or HTTP API. Changes made using these management interfaces persist automatically and the XML configuration files are overwritten by the Management API. The management CLI and management console are the preferred methods, and it is not recommended to edit the XML configuration files manually.

3.1. ABOUT SUBSYSTEMS, EXTENSIONS, AND PROFILES

Different aspects of JBoss EAP functionality are configured in different subsystems. For example, application and server logging are configured in the **`logging`** subsystem.

A *subsystem* provides configuration options for a particular extension. An *extension* is a module that extends the core functionality of the server. Extensions are loaded as they are needed by deployments, and are unloaded when they are no longer needed.

A collection of subsystem configurations makes up a *profile*, which is configured to satisfy the needs of the server. A standalone server has a single, unnamed profile. A managed domain can define many profiles for use by server groups in the domain.

For more information on the available subsystems, see Overview of JBoss EAP Subsystems.

Using the Management Console or the Management CLI

Both the management console and the management CLI are valid, supported ways of updating the configuration of a JBoss EAP instance. Deciding between the two is a matter of preference. Those who prefer to use a graphical, web-based interface should use the management console. Those who prefer a command-line interface should use the management CLI.

3.2. MANAGEMENT USERS

The default JBoss EAP configuration provides local authentication so that a user can access the management CLI on the local host without requiring authentication.

However, you must add a management user if you want to access the management CLI remotely or use the management console, which is considered remote access even if the traffic originates on the local host. If you attempt to access the management console before adding a management user, you will receive an error message.

If JBoss EAP is installed using the graphical installer, then a management user is created during the installation process.

This guide covers simple user management for JBoss EAP using the **`add-user`** script, which is a utility for adding new users to the properties files for out-of-the-box authentication. 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* guide.

3.2.1. Adding a Management User

1. Run the **add-user** utility script and follow the prompts.

```
$ EAP_HOME/bin/add-user.sh
```

NOTE

For Windows Server, use the **EAP_HOME\bin\add-user.bat** script.

2. Press **ENTER** to select the default option **a** to add a management user.
   This user will be added to the *ManagementRealm* and will be authorized to perform management operations using the management console or management CLI. The other choice (**b**) adds a user to the *ApplicationRealm*, which is used for applications and provides no particular permissions.

3. Enter the desired username and password. You will be prompted to confirm the password.
   By default, JBoss EAP allows weak passwords but will issue a warning. See Setting Add-User Utility Password Restrictions for details on changing this default behavior.

4. Enter a comma-separated list of groups to which the user belongs. If you do not want the user to belong to any groups, press **ENTER** to leave it blank.

5. Review the information and enter **yes** to confirm.

6. Determine whether this user represents a remote JBoss EAP server instance. For a basic management user, enter **no**.
   One type of user that may need to be added to the *ManagementRealm* is a user representing another instance of JBoss EAP, which must be able to authenticate to join as a member of a cluster. If this is the case, then answer **yes** to this prompt and you will be given a hashed secret value representing the user's password, which will need to be added to a different configuration file.

Users can also be created non-interactively by passing parameters to the **add-user** script. This approach is not recommended on shared systems, because the passwords will be visible in log and history files. For more information, see Running the Add-User Utility Non-Interactively.

3.2.2. Running the Add-User Utility Non-Interactively

You can run the **add-user** script non-interactively by passing in arguments on the command line. At a minimum, the username and password must be provided.

WARNING

This approach is not recommended on shared systems, because the passwords will be visible in log and history files.

Create a User Belonging to Multiple Groups

The following command adds a management user (**mgmtuser1**) with the **guest** and **mgmtgroup** groups.

```
$ EAP_HOME/bin/add-user.sh -u 'mgmtuser1' -p 'password1!' -g
'guest,mgmtgroup'
```

Specify an Alternative Properties File
By default, user and group information created using the **add-user** script are stored in properties files located in the server configuration directory.

User information is stored in the following properties files:

- **EAP_HOME/standalone/configuration/mgmt-users.properties**
- **EAP_HOME/domain/configuration/mgmt-users.properties**

Group information is stored in the following properties files:

- **EAP_HOME/standalone/configuration/mgmt-groups.properties**
- **EAP_HOME/domain/configuration/mgmt-groups.properties**

These default directories and properties file names can be overridden. The following command adds a new user, specifying a different name and location for the user properties files.

```
$ EAP_HOME/bin/add-user.sh -u 'mgmtuser2' -p 'password1!' -sc
'/path/to/standaloneconfig/' -dc '/path/to/domainconfig/' -up
'newname.properties'
```

The new user was added to the user properties files now located at **/path/to/standaloneconfig/newname.properties** and **/path/to/domainconfig/newname.properties**. Note that these files must already exist or you will see an error.

For a complete listing of all available **add-user** arguments and their purposes, use the **--help** argument or see the Add-User Utility Arguments section.

3.2.3. Setting Add-User Utility Password Restrictions

The password restrictions for the **add-user** utility script can be configured using the **EAP_HOME/bin/add-user.properties** file.

By default, JBoss EAP allows weak passwords but will issue a warning. To reject passwords that do not meet the minimum requirements specified, set the **password.restriction** property to **REJECT**.

Additional password requirements that can be configured in the **EAP_HOME/bin/add-user.properties** file:

- Minimum length
- Minimum alphabetic characters
- Minimum digits
- Minimum symbols

- List of forbidden passwords (such as **admin**)
- Whether to allow a password that matches the username

3.3. MANAGEMENT INTERFACES

3.3.1. Management CLI

The management command-line interface (CLI) is a command-line administration tool for JBoss EAP.

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.

Many common terminal commands are available, such as **ls**, **cd**, and **pwd**. The management CLI also supports tab completion.

For detailed information on using the management CLI, including commands and operations, syntax, and running in batch mode, see the JBoss EAP *Management CLI Guide*.

Launch the Management CLI

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

NOTE

For Windows Server, use the **EAP_HOME\bin\jboss-cli.bat** script.

Connect to a Running Server

```
connect
```

Or you can launch the management CLI and connect in one step by using the **EAP_HOME/bin/jboss-cli.sh --connect** command.

Display Help
Use the following command for general help.

```
help
```

Use the following command for help on a specific command.

```
deploy --help
```

Quit the Management CLI

```
quit
```

View System Settings
The following command uses the **read-attribute** operation to display whether the example datasource is enabled.

```
/subsystem=datasources/data-source=ExampleDS:read-attribute(name=enabled)
{
    "outcome" => "success",
    "result" => true
}
```

When running in a managed domain, you must specify which profile to update by preceding the command with **`/profile=PROFILE_NAME`**.

```
/profile=default/subsystem=datasources/data-source=ExampleDS:read-
attribute(name=enabled)
```

Update System Settings
The following command uses the **`write-attribute`** operation to disable the example datasource.

```
/subsystem=datasources/data-source=ExampleDS:write-
attribute(name=enabled,value=false)
```

Start Servers
The management CLI can also be used to start and stop servers when running in a managed domain.

```
/host=HOST_NAME/server-config=server-one:start
```

3.3.2. Management Console

The management console is a web-based administration tool for JBoss EAP.

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 performed by the current user 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.

For a JBoss EAP instance running on the local host using the default management port, the management console can be accessed through a web browser at http://localhost:9990/console/App.html. You will need to authenticate with a user that has permissions to access the management console.

The management console provides the following tabs for navigating and managing your JBoss EAP standalone server or managed domain.

Home

Learn how to accomplish several common configuration and management tasks. Take a tour to become familiar with the JBoss EAP management console.

Deployments

Add, remove, and enable deployments. In a managed domain, assign deployments to server groups.

Configuration

Configure available subsystems, which provide capabilities such as web services, messaging, or high availability. In a managed domain, manage the profiles that contain different subsystem configurations.

Runtime

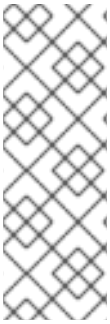
View runtime information, such as server status, JVM usage, and server logs. In a managed domain, manage your hosts, server groups, and servers.

Access Control

Assign roles to users and groups when using Role-Based Access Control.

Patching

Apply patches to your JBoss EAP instances.



NOTE

To take a tour of the updated management console, click the **Take a Tour** link on the management console home page.

To view details about the form fields, click the **Need Help?** link.

To view the message history of configuration actions you have performed, click the **Messages** link in the top-right of the management console.

3.3.2.1. Enable/Disable Management Console

You can enable or disable the management console by setting the **`console-enabled`** boolean attribute of **`/core-service=management/management-interface=http-interface`** resource. For master host in domain mode, **`/host=master/core-service=management/management-interface=http-interface`**.

For example, to enable:

```
/core-service=management/management-interface=http-interface:write-
attribute(name=console-enabled,value=true)
```

For example, to disable:

```
/core-service=management/management-interface=http-interface:write-
attribute(name=console-enabled,value=false)
```

3.3.2.2. Change the Language of the Management Console

By default, the language settings of the management console is English. You can choose to use one of the following languages instead:

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

To Change the Language of the Management Console

1. Log in to the management console.
2. Click the **Settings** link in the lower-right corner of the management console.
3. Select the required language from the **Locale** selection box.
4. Select **Save**. A confirmation box informs you that you need to reload the application.
5. Click **Confirm**. The system refreshes your web browser automatically to use the selected locale.

3.4. MANAGEMENT APIS

3.4.1. HTTP API

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

The HTTP API is used by the JBoss EAP management console but offers integration capabilities for other clients as well. By default, the HTTP API is accessible at **`http://HOST_NAME:9990/management`**. This URL will display the raw attributes and values exposed to the API.

Read Resources
While you can read, write, or perform other operations using the HTTP **POST** method, you can perform some read operations using a **GET** request. The HTTP **GET** method uses the following URL format.

```
http://HOST_NAME:9990/management/PATH_TO_RESOURCE?
operation=OPERATION&PARAMETER=VALUE
```

Be sure to replace all of the replaceable values with those that are appropriate for your request. The following values are the available options for the ***OPERATION*** replaceable value:

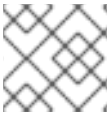
Value	Description
attribute	Performs the **`read-attribute`** operation.
operation-description	Performs the **`read-operation-description`** operation.
operation-names	Performs the **`read-operation-names`** operation.
resource	Performs the **`read-resource`** operation.
resource-description	Performs the **`read-resource-description`** operation.
snapshots	Performs the **`list-snapshots`** operation.

The following example URLs show how to perform read operations using the HTTP API.

Example: Read All Attributes and Values for a Resource

```
http://HOST_NAME:9990/management/subsystem/undertow/server/default-
server/http-listener/default
```

This displays all attributes and their values for the **default** HTTP listener.



NOTE

The default operation is **read-resource**.

Example: Read the Value of an Attribute for a Resource

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

This reads the value of the **enabled** attribute for the **ExampleDS** datasource.

Update Resources
You can use the HTTP **POST** method to update configuration values or perform other operations using the HTTP API. You must provide authentication for these operations.

The following examples show how to update resources using the HTTP API.

Example: Update the Value of an Attribute for a Resource

```
$ curl --digest http://HOST_NAME:9990/management --header "Content-Type:
application/json" -u USERNAME:PASSWORD -d '{"operation":"write-attribute",
"address":["subsystem","datasources","data-source","ExampleDS"],
"name":"enabled", "value":"false", "json.pretty":"1"}'
```

This updates the value of the **enabled** attribute for the **ExampleDS** datasource to **false**.

Example: Issue an Operation to the Server

```
$ curl --digest http://localhost:9990/management --header "Content-Type:
application/json" -u USERNAME:PASSWORD -d '{"operation":"reload"}'
```

This reloads the server.

See Deploying Applications Using the HTTP API for information on how to deploy applications to JBoss EAP using the HTTP API.

3.4.2. Native API

The native API endpoint is the entry point for management clients that rely on the native protocol to integrate with the JBoss EAP management layer. The native API is used by the JBoss EAP management CLI but offers integration capabilities for other clients as well.

The following Java code shows an example of how to execute management operations from Java code using the native API.

NOTE

You must add the required JBoss EAP libraries, found in the ***EAP_HOME*/bin/client/jboss-cli-client.jar** file, to your class path.

Example: Using the Native API to Read Resources

```
// Create the management client
ModelControllerClient client =
ModelControllerClient.Factory.create("localhost", 9990);

// Create the operation request
ModelNode op = new ModelNode();

// Set the operation
op.get("operation").set("read-resource");

// Set the address
ModelNode address = op.get("address");
address.add("subsystem", "undertow");
address.add("server", "default-server");
address.add("http-listener", "default");

// Execute the operation and manipulate the result
ModelNode returnVal = client.execute(op);
System.out.println("Outcome: " + returnVal.get("outcome").toString());
System.out.println("Result: " + returnVal.get("result").toString());

// Close the client
client.close();
```

3.5. CONFIGURATION DATA

3.5.1. Standalone Server Configuration Files

The standalone configuration files are located in the **EAP_HOME/standalone/configuration/** directory. A separate file exists for each of the four predefined profiles (*default*, *ha*, *full*, *full-ha*).

Table 3.1. Standalone Configuration Files

Configuration File	Purpose
standalone.xml	This standalone configuration file is the default configuration that is used when you start your standalone server. It contains all information about the server, including subsystems, networking, deployments, socket bindings, and other configurable details. It does not provide the subsystems necessary for messaging or high availability.
standalone-ha.xml	This standalone configuration file includes all of the default subsystems and adds the **modcluster** and **jgroups** subsystems for high availability. It does not provide the subsystems necessary for messaging.

Configuration File	Purpose
standalone-full.xml	This standalone configuration file includes all of the default subsystems and adds the **messaging-activemq** and **iiop-openjdk** subsystems. It does not provide the subsystems necessary for high availability.
standalone-full-ha.xml	This standalone configuration file includes support for every possible subsystem, including those for messaging and high availability.

By default, starting JBoss EAP as a standalone server uses the **standalone.xml** file. To start JBoss EAP with a different configuration, use the **--server-config** argument. For example,

```
$ EAP_HOME/bin/standalone.sh --server-config=standalone-full.xml
```

3.5.2. Managed Domain Configuration Files

The managed domain configuration files are located in the **EAP_HOME/domain/configuration/** directory.

Table 3.2. Managed Domain Configuration Files

Configuration File	Purpose
domain.xml	This is the main configuration file for a managed domain. Only the domain master reads this file. This file contains the configurations for all of the profiles (*default*, *ha*, *full*, *full-ha*).
host.xml	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.
host-master.xml	This file includes only the configuration details necessary to run a server as the master domain controller.
host-slave.xml	This file includes only the configuration details necessary to run a server as a managed domain host controller.

By default, starting JBoss EAP in a managed domain uses the **host.xml** file. To start JBoss EAP with a different configuration, use the **--host-config** argument. For example,

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

3.5.3. Backing Up Configuration Data

In order to later restore the JBoss EAP server configuration, items in the following locations should be backed up:

- **`EAP_HOME/standalone/configuration/`**
  - Back up the entire directory to save user data, server configuration, and logging settings for standalone servers.
- **`EAP_HOME/domain/configuration/`**
  - Back up the entire directory to save user and profile data, domain and host configuration, and logging settings for managed domains.
- **`EAP_HOME/modules/`**
  - Back up any custom modules.
- **`EAP_HOME/welcome-content/`**
  - Back up any custom welcome content.
- **`EAP_HOME/bin/`**
  - Back up any custom scripts or startup configuration files.

3.5.4. Configuration File Snapshots

To assist in the maintenance and management of the server, JBoss EAP creates a timestamped version of the original configuration file at the time of startup. Any additional configuration changes made by management operations will result in the original file being automatically backed up, and a working copy of the instance being preserved for reference and rollback. Additionally, configuration snapshots can be taken, which are point-in-time copies of the current server configuration. These snapshots can be saved and loaded by an administrator.

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

Take a Snapshot
Use the management CLI to take a snapshot of the current configurations.

```
:take-snapshot
{
    "outcome" => "success",
    "result" =>
"EAP_HOME/standalone/configuration/standalone_xml_history/snapshot/2015102
2-133109702standalone.xml"
}
```

List Snapshots
Use the management CLI to list all snapshots that have been taken.

```
:list-snapshots
{
    "outcome" => "success",
    "result" => {
        "directory" =>
"EAP_HOME/standalone/configuration/standalone_xml_history/snapshot",
        "names" => [
            "20151022-133109702standalone.xml",
```

```
            "20151022-132715958standalone.xml"
        ]
    }
}
```

Delete a Snapshot
Use the management CLI to delete a snapshot.

```
:delete-snapshot(name=20151022-133109702standalone.xml)
```

Start the Server with a Snapshot
The server can be started using a snapshot or an automatically-saved version of the configuration.

1. Navigate to the **`EAP_HOME/standalone/configuration/standalone_xml_history`** directory and identify the snapshot or saved configuration file to be loaded.
2. Start the server and point to the selected configuration file. Pass in the file path relative to the configuration directory, **`EAP_HOME/standalone/configuration/`**.

```
$ EAP_HOME/bin/standalone.sh --server-
config=standalone_xml_history/snapshot/20151022-
133109702standalone.xml
```

NOTE

When running in a managed domain, use the **`--host-config`** argument instead to specify the configuration file.

3.5.5. View Configuration Changes

JBoss EAP 7 provides the ability to track configuration changes made to the running system. This allows administrators to view a history of configuration changes made by other authorized users.

IMPORTANT

Changes are stored in memory and are not persisted between server restarts. This feature is not a replacement for management audit logging.

To enable to tracking of configuration changes, use the following management CLI command. You can specify how many entries to store using the **`max-history`** attribute.

```
/core-service=management/service=configuration-changes:add(max-history=10)
```

To view the list of most recent configuration changes, use the following management CLI command.

```
/core-service=management/service=configuration-changes:list-changes
```

This will list each configuration change made, with the date, origin, outcome, and operation details. For example, the below output from the **`list-changes`** command shows configuration changes, with the most recent displayed first.

```
{
```

```
    "outcome" => "success",
    "result" => [
        {
            "operation-date" => "2016-02-12T18:37:00.354Z",
            "access-mechanism" => "NATIVE",
            "remote-address" => "127.0.0.1/127.0.0.1",
            "outcome" => "success",
            "operations" => [{
                "address" => [],
                "operation" => "reload",
                "operation-headers" => {
                    "caller-type" => "user",
                    "access-mechanism" => "NATIVE"
                }
            }]
        },
        {
            "operation-date" => "2016-02-12T18:34:16.859Z",
            "access-mechanism" => "NATIVE",
            "remote-address" => "127.0.0.1/127.0.0.1",
            "outcome" => "success",
            "operations" => [{
                "address" => [
                    ("subsystem" => "datasources"),
                    ("data-source" => "ExampleDS")
                ],
                "operation" => "write-attribute",
                "name" => "enabled",
                "value" => false,
                "operation-headers" => {
                    "caller-type" => "user",
                    "access-mechanism" => "NATIVE"
                }
            }]
        },
        {
            "operation-date" => "2016-02-12T18:24:11.670Z",
            "access-mechanism" => "HTTP",
            "remote-address" => "127.0.0.1/127.0.0.1",
            "outcome" => "success",
            "operations" => [{
                "operation" => "remove",
                "address" => [
                    ("subsystem" => "messaging-activemq"),
                    ("server" => "default"),
                    ("jms-queue" => "ExpiryQueue")
                ],
                "operation-headers" => {"access-mechanism" => "HTTP"}
            }]
        }
    ]
}
```

This example lists the details of three operations performed that impacted the configuration:

- Reloading the server from the management CLI.

- Disabling the **ExampleDS** datasource from the management CLI.
- Removing the **ExpiryQueue** queue from the management console.

3.5.6. Property Replacement

JBoss EAP allows you to use expressions to define replaceable properties in place of literal values in the configuration. Expressions use the format **${PARAMETER:DEFAULT_VALUE}**. If the specified parameter is set, then the parameter's value will be used. Otherwise, the default value provided will be used.

The supported sources for resolving 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. For deployment types that support subdeployments, the resolution is scoped to all subdeployments if the properties file is in the outer deployment, for example the EAR. If the properties file is in the subdeployment, then the resolution is scoped just to that subdeployment.

The example below from the **standalone.xml** configuration file sets the **inet-address** for the **public** interface to **127.0.0.1** unless the **jboss.bind.address** parameter is set.

```
<interface name="public">
    <inet-address value="${jboss.bind.address:127.0.0.1}"/>
</interface>
```

The **jboss.bind.address** parameter can be set when starting EAP as a standalone server with the following command:

```
$ EAP_HOME/bin/standalone.sh -Djboss.bind.address=IP_ADDRESS
```

Nested Expressions
Expressions can 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:

```
${SYSTEM_VALUE_1${SYSTEM_VALUE_2}}
```

Nested expressions are evaluated recursively, so the *inner* expression is first evaluated, then the *outer* expression is evaluated. Expressions may also be recursive, where an expression resolves to another expression, which is then resolved. Nested expressions are permitted anywhere that expressions are permitted, with the exception of management CLI commands.

An example of where a nested expression might be used is if the password used in a datasource definition is masked. The configuration for the datasource might have the following line:

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

The value of **ds_ExampleDS** could be replaced with a system property (**datasource_name**) using a nested expression. The configuration for the datasource could instead have the following line:

```
<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.

Descriptor-Based Property Replacement
Application configuration, such as datasource connection parameters, typically varies between development, testing, and production environments. 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, 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.

There are several flags in the **ee** subsystem that control whether property replacement is applied.

JBoss-specific descriptor replacement is controlled by the **jboss-descriptor-property-replacement** flag and is *enabled* by default. When enabled, properties can be replaced in the following deployment descriptors:

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

The following management CLI command can be used to enable or disable property replacement in JBoss-specific descriptors:

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

Java EE descriptor replacement controlled by the **spec-descriptor-property-replacement** flag and is *disabled* by default. When enabled, properties can be replaced in the following deployment descriptors:

- **ejb-jar.xml**
- **persistence.xml**
- **application.xml**
- **web.xml**

The following management CLI command can be used to enable or disable property replacement in Java EE descriptors:

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

3.6. FILE SYSTEM PATHS

JBoss EAP uses logical names for file system paths. Other areas of the configuration 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.

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

Example: Relative Path Example for the Server Log Directory

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

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

Table 3.3. Standard Paths

Property	Description
java.ext.dirs	The Java development kit extension directory paths.
java.home	The Java installation directory
jboss.controller.temp.dir	A common alias for standalone servers and managed domains. The directory to be used for temporary file storage. Equivalent to **jboss.domain.temp.dir** in a managed domain, and **jboss.server.temp.dir** on a standalone server.
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.
jboss.home.dir	The root directory of the JBoss EAP distribution.
jboss.server.base.dir	The base directory for standalone server content.
jboss.server.config.dir	The directory that contains the standalone server configuration.

Property	Description
jboss.server.data.dir	The directory the standalone server will use for persistent data file storage.
jboss.server.log.dir	The directory the standalone server will use for log file storage.
jboss.server.temp.dir	The directory the standalone server will use for temporary file storage.
jboss.server.deploy.dir	The directory that the standalone server will use for storing deployed content.
user.dir	The user's current working directory.
user.home	The user home directory.

You can override a standard path or add a custom path.

3.6.1. Override a Standard Path

You can override the default locations of the standard paths that begin with **`jboss.server.*`** or **`jboss.domain.*`**. This can be done in one of two ways:

- Pass in the command-line argument when you start the server. For example:

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

- Modify the **JAVA_OPTS** variable in the server configuration file (**`standalone.conf`** or **`domain.conf`**). For example:

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

Overriding a Managed Domain's Standard Paths
In this example, the objective is to store domain files in the **`/opt/jboss_eap/domain_data`** directory, and give each top-level directory a custom name. The default directory grouping, **`by-server`**, is used.

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

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

```
$ EAP_HOME/bin/domain.sh -
Djboss.domain.temp.dir=/opt/jboss_eap/domain_data/all_temp -
Djboss.domain.log.dir=/opt/jboss_eap/domain_data/all_logs -
```

```
Djboss.domain.data.dir=/opt/jboss_eap/domain_data/all_data -
Djboss.domain.servers.dir=/opt/jboss_eap/domain_data/all_servers
```

The resulting path structure will be as follows:

```
/opt/jboss_eap/domain_data/
                ├── all_data
                ├── all_logs
                ├── all_servers
                │   ├── server-one
                │   │   ├── data
                │   │   ├── log
                │   │   └── tmp
                │   └── server-two
                │       ├── data
                │       ├── log
                │       └── tmp
                └── all_temp
```

3.6.2. Add a Custom Path

You can add a custom file system path using the management CLI or the management console.

- From the management CLI, you can add a new path using the following management CLI command.

  ```
  /path=my.custom.path:add(path=/my/custom/path)
  ```

- From the management console, you can configure file system paths by navigating to the **Configuration** tab and selecting **Paths**. From there, you can add, modify, and remove paths.

You can then use this custom path in your configuration. For example, the below log handler uses a custom path for its relative path.

```
<subsystem xmlns="urn:jboss:domain:logging:3.0">
  ...
  <periodic-rotating-file-handler name="FILE" autoflush="true">
    <formatter>
      <named-formatter name="PATTERN"/>
    </formatter>
    <file relative-to="my.custom.path" path="server.log"/>
    <suffix value=".yyyy-MM-dd"/>
    <append value="true"/>
  </periodic-rotating-file-handler>
  ...
</subsystem>
```

3.6.3. Directory Grouping

In a managed domain, each server's files are stored in the **`EAP_HOME/domain`** directory. You can specify how to organize the subdirectories for servers using the host controller's **`directory-grouping`** attribute. Directories can be grouped either by *server* or by *type*. By default, directories are grouped by *server*.

Directory Grouping by Server
By default, directories are grouped 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.

If JBoss EAP is installed using the ZIP installation method, the default directory structure (grouped by server) will be as follows.

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

To group domain directories by server, enter the following management CLI command:

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

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

```
<servers directory-grouping="by-server">
  <server name="server-one" group="main-server-group"/>
  <server name="server-two" group="main-server-group" auto-start="true">
    <socket-bindings port-offset="150"/>
  </server>
</servers>
```

Directory Grouping by Type
Instead of grouping directories by server, you can instead group by file type. If your administration is *file type-centric*, this configuration is recommended. For example, this would allow you to easily back up only `data` files.

If JBoss EAP is installed using the ZIP installation method and the domain's files are grouped by type, the directory structure 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
```

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

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

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

```
<servers directory-grouping="by-type">
  <server name="server-one" group="main-server-group"/>
  <server name="server-two" group="main-server-group" auto-start="true">
    <socket-bindings port-offset="150"/>
  </server>
</servers>
```

3.7. SYSTEM PROPERTIES

You can use Java system properties to configure many JBoss EAP options, as well as set any name-value pair for use within the application server.

System properties can be used to override default values in the JBoss EAP configuration. For example, the following XML configuration for the public interface bind address shows that it can be set by the **jboss.bind.address** system property, but if the system property is not provided, it will default to **127.0.0.1**.

```
<inet-address value="${jboss.bind.address:127.0.0.1}"/>
```

There are a few ways you can set system properties in JBoss EAP, including:

- passing the system property to the JBoss EAP startup script
- using the management CLI
- using the management console
- using the **JAVA_OPTS** environment variable

If you use a JBoss EAP managed domain, system properties can be applied to either the whole domain, a specific server group, a specific host and all its server instances, or just to one specific server instance. As with most other JBoss EAP domain settings, a system property set at a more specific level will override a more abstract one. See the Domain Management chapter for more information.

Passing a System Property to the Startup Script

You can pass a system property to the JBoss EAP startup script by using the **-D** argument. For example:

```
$ EAP_HOME/bin/standalone.sh -Djboss.bind.address=192.168.1.2
```

This method of setting the system property is especially useful for JBoss EAP options that need to be set before JBoss EAP starts.

Setting a System Property Using the Management CLI

Using the management CLI, you can set a system property using the following syntax:

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

For example:

```
/system-property=jboss.bind.address:add(value=192.168.1.2)
```

When setting system properties using the management CLI, some JBoss EAP options, including the above example of **`jboss.bind.address`**, will only take effect after the next server restart.

For a managed domain, the above example configures a system property for the entire domain, but you can also set or override system properties at more specific levels of the domain configuration.

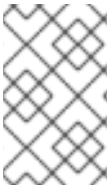
Setting a System Property Using the Management Console

- For a standalone JBoss EAP server, you can configure system properties in the management console under the **Configuration** tab. Select **System Properties**, and click the **View** button.
- For a managed domain:
  - Domain-level system properties can be set in the **Configuration** tab. Select **System Properties**, and click the **View** button.
  - Server group, and server-level system properties can be set in the **Runtime** tab. Select the server group or server you want to configure, click the **View** button next to the server group or server name, and select the **System Properties** tab.
  - Host-level system properties can be set in the **Runtime** tab. Select the host you want to configure, then using the drop-down menu next to the host name, select **Properties**.

Setting a System Property Using JAVA_OPTS

System properties can also be configured using the **JAVA_OPTS** environment variable. There are many ways to modify **JAVA_OPTS**, but JBoss EAP provides a configuration file for setting **JAVA_OPTS** that is used by the JBoss EAP process.

For a standalone server, this file is **`EAP_HOME/bin/standalone.conf`**, or for a managed domain, it is **`EAP_HOME/bin/domain.conf`**. For Microsoft Windows systems these files have a **`.bat`** extension.



NOTE

For an RPM installation, the RPM service configuration file is the preferred location to modify **JAVA_OPTS** to configure system properties. For more information, see Configure RPM Service Properties.

Add your system property definition to **JAVA_OPTS** in the relevant configuration file. The examples below demonstrate setting the bind address on a Red Hat Enterprise Linux system.

- For **`standalone.conf`**, add your **JAVA_OPTS** system property definition at the end of the file. For example:

```
...
# Set the bind address
JAVA_OPTS="$JAVA_OPTS -Djboss.bind.address=192.168.1.2"
```

- For **`domain.conf`**, **JAVA_OPTS** must be set before the process controller **JAVA_OPTS** setting. For example:

```
...
# Set the bind address
JAVA_OPTS="$JAVA_OPTS -Djboss.bind.address=192.168.1.2"
```

```
# The ProcessController process uses its own set of java options
if [ "x$PROCESS_CONTROLLER_JAVA_OPTS" = "x" ]; then
...
```

3.8. MANAGEMENT AUDIT LOGGING

You can enable audit logging for the management interfaces, which will log all operations performed using the management console, management CLI, or custom application that uses the Management API. Audit log entries are stored in JSON format. By default, audit logging is disabled.

You can configure audit logging to output to a file or to a syslog server.

NOTE

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

Standalone Server Audit Logging

Though disabled by default, the default audit logging configuration writes to a file.

```
<audit-log>
    <formatters>
        <json-formatter name="json-formatter"/>
    </formatters>
    <handlers>
        <file-handler name="file" formatter="json-formatter" path="audit-
log.log" relative-to="jboss.server.data.dir"/>
    </handlers>
    <logger log-boot="true" log-read-only="false" enabled="false">
        <handlers>
            <handler name="file"/>
        </handlers>
    </logger>
</audit-log>
```

This configuration can be read using the following management CLI command.

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

See Enable Audit Logging to enable audit logging for a standalone server.

Managed Domain Audit Logging

Though disabled by default, the default audit logging configuration writes a file for each host and for each server.

```
<audit-log>
    <formatters>
        <json-formatter name="json-formatter"/>
    </formatters>
    <handlers>
        <file-handler name="host-file" formatter="json-formatter"
relative-to="jboss.domain.data.dir" path="audit-log.log"/>
        <file-handler name="server-file" formatter="json-formatter"
```

```
relative-to="jboss.server.data.dir" path="audit-log.log"/>
    </handlers>
    <logger log-boot="true" log-read-only="false" enabled="false">
        <handlers>
            <handler name="host-file"/>
        </handlers>
    </logger>
    <server-logger log-boot="true" log-read-only="false" enabled="false">
        <handlers>
            <handler name="server-file"/>
        </handlers>
    </server-logger>
</audit-log>
```

This configuration can be read using the following management CLI command.

```
/host=HOST_NAME/core-service=management/access=audit:read-
resource(recursive=true)
```

See Enable Audit Logging to enable audit logging for a managed domain.

3.8.1. Enable Management Audit Logging

JBoss EAP is preconfigured with file handlers for audit logging, though audit logging is disabled by default. The management CLI command to enable audit logging depends on whether you are running as a standalone server or in a managed domain. See Management Audit Logging Attributes for file handler attributes.

To set up syslog audit logging, see Set Up Management Audit Logging to a Syslog Server .

Enable Standalone Server Audit Logging
Audit logging can be enabled using the following command.

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

By default, this will write the audit log to **`EAP_HOME/standalone/data/audit-log.log`**.

Enable Managed Domain Audit Logging
The default audit logging configuration for a managed domain is preconfigured to write an audit log for each host and each server.

Audit logging for each host can be enabled using the following command.

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

By default, this will write the audit logs to **`EAP_HOME/domain/data/audit-log.log`**.

Audit logging for each server can be enabled using the following command.

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

By default, this will write the audit logs to
EAP_HOME/domain/servers/SERVER_NAME/data/audit-log.log

3.8.2. Send Management Audit Logging to a Syslog Server

A syslog handler specifies the parameters by which audit log entries are sent to a syslog server, specifically the syslog server's host name 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.

By default, audit logging is preconfigured to output to a file when enabled. Use the following steps to set up and enable audit logging to a syslog server. See Management Audit Logging Attributes for syslog handler attributes.

1. Add a syslog handler.
   Create the syslog handler, specifying the host and port of the syslog server. In a managed domain, you must precede the **/core-service** commands with **/host=HOST_NAME**.

   ```
   batch
   /core-service=management/access=audit/syslog-
   handler=SYSLOG_HANDLER_NAME:add(formatter=json-formatter)
   /core-service=management/access=audit/syslog-
   handler=SYSLOG_HANDLER_NAME/protocol=udp:add(host=HOST_NAME,port=POR
   T)
   run-batch
   ```

   NOTE

   The parameters to pass in differ depending on the protocol specified.

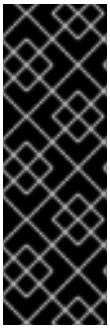
   To configure the handler to use TLS to communicate securely with the syslog server, you must also configure the authentication, for example:

   ```
   /core-service=management/access=audit/syslog-
   handler=SYSLOG_HANDLER_NAME/protocol=tls/authentication=t
   ruststore:add(keystore-path=PATH_TO_TRUSTSTORE,keystore-
   password=TRUSTSTORE_PASSWORD)
   ```

2. Add a reference to the syslog handler.
   In a managed domain, you must precede this command with **/host=HOST_NAME**.

   ```
   /core-service=management/access=audit/logger=audit-
   log/handler=SYSLOG_HANDLER_NAME:add
   ```

3. Enable audit logging.
   See Enable Management Audit Logging to enable audit logging.



IMPORTANT

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, see the *Basic Configuration of Rsyslog* section of the *System Administrator's Guide* for Red Hat Enterprise Linux at https://access.redhat.com/documentation/en/red-hat-enterprise-linux/

3.8.3. Read Audit Log Entries

Audit log entries output to files 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 audit log entries are stored in JSON format. Each log entry begins with an optional timestamp, followed by the fields in the below table.

Table 3.4. Management Audit Log Fields

Field Name	Description
access	This can have one of the following values: • **`NATIVE`** - The operation came in through the native management interface. • **`HTTP`** - The operation came in through the domain HTTP interface. • **`JMX`** - The operation came in through the `jmx` subsystem.
booting	Has the value `true` if the operation was executed during the bootup process, or `false` if it was executed once the server is up and running.
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.
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.
r/o	Has the value `true` if the operation does not change the management model, or `false` if it does not.
remote-address	The address of the client executing this operation.

Field Name	Description
success	Has the value `true` if the operation was successful, or `false` if it was rolled back.
type	This can have the value `core`, meaning it is a management operation, or `jmx`, meaning it comes from the `jmx` subsystem.
user	The username of the authenticated user. If the operation occurred using the management CLI on the same machine as the running server, the special user `$local` is used.
version	The version number of the JBoss EAP instance.

CHAPTER 4. NETWORK AND PORT CONFIGURATION

4.1. INTERFACES

JBoss EAP references named interfaces throughout the configuration. This allows the configuration to reference individual interface declarations with logical names, rather than requiring the full details of the interface at each use.

This also allows for easier configuration in a managed domain, where network interface details can vary across multiple machines. Each server instance can correspond to a logical name group.

The **standalone.xml**, **domain.xml**, and **host.xml** files all include interface declarations. There are several preconfigured interface names, depending on which default configuration is used. The **management** interface can be used for all components and services that require the management layer, including the HTTP management endpoint. The **public** interface can be used for all application-related network communications. The **unsecure** interface is used for IIOP sockets in the standard configuration. The **private** interface is used for JGroups sockets in the standard configuration.

4.1.1. 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="private">
    <inet-address value="${jboss.bind.address.private:127.0.0.1}"/>
  </interface>
  <interface name="unsecure">
    <inet-address value="${jboss.bind.address.unsecure:127.0.0.1}"/>
  </interface>
</interfaces>
```

By default, JBoss EAP binds these interfaces to **127.0.0.1**, but these values can be overridden at runtime by setting the appropriate property. For example, the **inet-address** of the **public** interface can be set when starting JBoss EAP as a standalone server with the following command.

```
$ EAP_HOME/bin/standalone.sh -Djboss.bind.address=IP_ADDRESS
```

Alternatively, you can use the **-b** switch on the server start command line. For more information about server start options, see Server Runtime Arguments.

IMPORTANT

If you modify the default network interfaces or ports that JBoss EAP uses, you must also remember to change any scripts that use the modified interfaces or ports. These include JBoss EAP service scripts, as well as remembering to specify the correct interface and port when accessing the management console or management CLI.

4.1.2. Configuring Interfaces

Network interfaces are declared by specifying a logical name and selection criteria for the physical interface. The selection 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. For a listing of all available interface selection criteria, see the Interface Attributes section.

Interfaces can be configured using the management console or the management CLI. Below are several examples of adding and updating interfaces. The management CLI command is shown first, followed by the corresponding configuration XML.

Add an Interface with a NIC Value
Add a new interface with a NIC value of **`eth0`**.

```
/interface=external:add(nic=eth0)
```

```
<interface name="external">
   <nic name="eth0"/>
</interface>
```

Add an Interface with Several Conditional Values
Add a new interface that matches any interface/address on the correct subnet if it is up, supports multicast, and is not point-to-point.

```
/interface=default:add(subnet-
match=192.168.0.0/16,up=true,multicast=true,not={point-to-point=true})
```

```
<interface name="default">
   <subnet-match value="192.168.0.0/16"/>
   <up/>
   <multicast/>
   <not>
      <point-to-point/>
   </not>
</interface>
```

Update an Interface Attribute
Update the **`public`** interface's default **`inet-address`** value, keeping the **`jboss.bind.address`** property to allow for this value to be set at runtime.

```
/interface=public:write-attribute(name=inet-
address,value="${jboss.bind.address:192.168.0.0}")
```

```
<interface name="public">
    <inet-address value="${jboss.bind.address:192.168.0.0}"/>
</interface>
```

Add an Interface to a Server in a Managed Domain

```
/host=master/server-config=SERVER_NAME/interface=INTERFACE_NAME:add(inet-
address=127.0.0.1)
```

```
<servers>
   <server name="SERVER_NAME" group="main-server-group">
      <interfaces>
```

```
        <interface name="INTERFACE_NAME">
           <inet-address value="127.0.0.1"/>
        </interface>
      </interfaces>
   </server>
</servers>
```

4.2. SOCKET BINDINGS

Socket bindings and socket binding groups allow you to define network ports and their relationship to the networking interfaces required for your JBoss EAP configuration. A socket binding is a named configuration for a socket. A socket binding group is a collection of socket binding declarations that are grouped under a logical name.

This allows other sections of the configuration to reference socket bindings by their logical name, rather than requiring the full details of the socket configuration at each use.

The declarations for these named configurations can be found in the **`standalone.xml`** and **`domain.xml`** configuration files. A standalone server contains only one socket binding group, while a managed domain 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 ports used by JBoss EAP by default depend on which socket binding groups are used and the requirements of your individual deployments.

4.2.1. Management Ports

Management ports were consolidated in JBoss EAP 7. By default, JBoss EAP 7 uses port **9990** for both native management, used by the management CLI, and HTTP management, used by the web-based management console. Port **9999**, which was used as the native management port in JBoss EAP 6, is no longer used but can still be enabled if desired.

If HTTPS is enabled for the management console, then port **9993** is used by default.

4.2.2. Default Socket Bindings

JBoss EAP ships with a socket binding group for each of the four predefined profiles (*default*, *ha*, *full*, *full-ha*).

For detailed information about the default socket bindings, such as default ports and descriptions, see the Default Socket Bindings section.

IMPORTANT

If you modify the default network interfaces or ports that JBoss EAP uses, you must also remember to change any scripts that use the modified interfaces or ports. These include JBoss EAP service scripts, as well as remembering to specify the correct interface and port when accessing the management console or management CLI.

Standalone Server
When running as a standalone server, only one socket binding group is defined per configuration file. Each standalone configuration file (**`standalone.xml`**, **`standalone-ha.xml`**, **`standalone-full.xml`**, **`standalone-full-ha.xml`**) defines socket bindings for the technologies used by its corresponding profile.

For example, the default standalone configuration file (`standalone.xml`) specifies the below socket bindings.

```
<socket-binding-group name="standard-sockets" default-interface="public"
port-offset="${jboss.socket.binding.port-offset:0}">
    <socket-binding name="management-http" interface="management"
port="${jboss.management.http.port:9990}"/>
    <socket-binding name="management-https" interface="management"
port="${jboss.management.https.port:9993}"/>
    <socket-binding name="ajp" port="${jboss.ajp.port:8009}"/>
    <socket-binding name="http" port="${jboss.http.port:8080}"/>
    <socket-binding name="https" port="${jboss.https.port:8443}"/>
    <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>
```

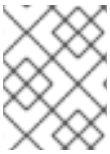
Managed Domain
When running in a managed domain, all socket binding groups are defined in the `domain.xml` file. There are four predefined socket binding groups:

- `standard-sockets`
- `ha-sockets`
- `full-sockets`
- `full-ha-sockets`

Each socket binding group specifies socket bindings for the technologies used by its corresponding profile. For example, the `full-ha-sockets` socket binding group defines several `jgroups` socket bindings, which are used by the *full-ha* profile for high availability.

```
<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="${jboss.ajp.port:8009}"/>
        <socket-binding name="http" port="${jboss.http.port:8080}"/>
        <socket-binding name="https" port="${jboss.https.port:8443}"/>
        <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-group>
    <socket-binding-group name="full-sockets" default-interface="public">
        <!-- Needed for server groups using the 'full' profile  -->
    ...
```

```
    </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="${jboss.ajp.port:8009}"/>
        <socket-binding name="http" port="${jboss.http.port:8080}"/>
        <socket-binding name="https" port="${jboss.https.port:8443}"/>
        <socket-binding name="iiop" interface="unsecure" port="3528"/>
        <socket-binding name="iiop-ssl" interface="unsecure" port="3529"/>
        <socket-binding name="jgroups-mping" interface="private" port="0"
multicast-address="${jboss.default.multicast.address:230.0.0.4}"
multicast-port="45700"/>
        <socket-binding name="jgroups-tcp" interface="private"
port="7600"/>
        <socket-binding name="jgroups-tcp-fd" interface="private"
port="57600"/>
        <socket-binding name="jgroups-udp" interface="private"
port="55200" multicast-
address="${jboss.default.multicast.address:230.0.0.4}" multicast-
port="45688"/>
        <socket-binding name="jgroups-udp-fd" interface="private"
port="54200"/>
        <socket-binding name="modcluster" port="0" multicast-
address="224.0.1.105" multicast-port="23364"/>
        <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>
```



NOTE

The socket configuration for the management interfaces is defined in the domain controller's **host.xml** file.

4.2.3. Configuring Socket Bindings

When defining a socket binding, you can configure the **port** and **interface** attributes, as well as multicast settings such as **multicast-address** and **multicast-port**. For details on all available socket bindings attributes, see the Socket Binding Attributes section.

Socket bindings can be configured using the management console or the management CLI. The following steps go through adding a socket binding group, adding a socket binding, and configuring socket binding settings using the management CLI.

1. Add a new socket binding group. Note that this step cannot be performed when running as a standalone server.

```
/socket-binding-group=new-sockets:add(default-interface=public)
```

2. Add a socket binding.

```
/socket-binding-group=new-sockets/socket-binding=new-socket-
```

```
binding:add(port=1234)
```

3. Change the socket binding to use an interface other than the default, which is set by the socket binding group.

```
/socket-binding-group=new-sockets/socket-binding=new-socket-
binding:write-attribute(name=interface,value=unsecure)
```

The following example shows how the XML configuration may look after the above steps have been completed.

```
<socket-binding-groups>
    ...
    <socket-binding-group name="new-sockets" default-interface="public">
        <socket-binding name="new-socket-binding" interface="unsecure"
port="1234"/>
    </socket-binding-group>
</socket-binding-groups>
```

4.2.4. Port Offsets

A port offset is a numeric offset value added to all port values specified in the socket binding group for that server. This allows the server to inherit the port values defined in its socket binding group, with an offset to ensure that it does not conflict with any other servers on the same host. For instance, if the HTTP port of the socket binding group is **8080**, and a server uses a port offset of **100**, then its HTTP port is **8180**.

Below is an example of setting a port offset of **250** for a server in a managed domain using the management CLI.

```
/host=master/server-config=server-two/:write-attribute(name=socket-
binding-port-offset,value=250)
```

Port offsets can be used for servers in a managed domain and for running multiple standalone servers on the same host.

You can pass in a port offset when starting a standalone server using the **jboss.socket.binding.port-offset** property.

```
$ EAP_HOME/bin/standalone.sh -Djboss.socket.binding.port-offset=100
```

4.3. IPV6 ADDRESSES

By default, JBoss EAP is configured to run using IPv4 addresses. The steps below show how to configure JBoss EAP to run using IPv6 addresses.

Configure the JVM Stack for IPv6 Addresses
Update the startup configuration to prefer IPv6 addresses.

1. Open the startup configuration file.
    - When running as a standalone server, edit the **EAP_HOME/bin/standalone.conf** file (or **standalone.conf.bat** for Windows Server).

- When running in a managed domain, edit the **EAP_HOME/bin/domain.conf** file (or **domain.conf.bat** for Windows Server).

2. Set the **java.net.preferIPv4Stack** property to **false**.

```
-Djava.net.preferIPv4Stack=false
```

3. Append the **java.net.preferIPv6Addresses** property and set it to **true**.

```
-Djava.net.preferIPv6Addresses=true
```

The following example shows how the JVM options in the startup configuration file may look after making the above changes.

```
# Specify options to pass to the Java VM.
#
if [ "x$JAVA_OPTS" = "x" ]; then
   JAVA_OPTS="-Xms1303m -Xmx1303m -Djava.net.preferIPv4Stack=false"
   JAVA_OPTS="$JAVA_OPTS -
Djboss.modules.system.pkgs=$JBOSS_MODULES_SYSTEM_PKGS -
Djava.awt.headless=true"
   JAVA_OPTS="$JAVA_OPTS -Djava.net.preferIPv6Addresses=true"
else
```

Update Interface Declarations for IPv6 Addresses

The default interface values in the configuration can be changed to IPv6 addresses. For example, the below management CLI command sets the **management** interface to the IPv6 loopback address (**::1**).

```
/interface=management:write-attribute(name=inet-
address,value="${jboss.bind.address.management:[::1]}")
```

The following example shows how the XML configuration may look after running the above command.

```
<interfaces>
    <interface name="management">
        <inet-address value="${jboss.bind.address.management:[::1]}"/>
    </interface>
    ....
</interfaces>
```

CHAPTER 5. JBOSS EAP SECURITY

JBoss EAP offers the ability to configure security for its own interfaces and services as well as provide security for applications that are running on it.

- See the *Security Architecture* guide for an overview of general security concepts as well as JBoss EAP-specific security concepts.
- See *How to Configure Server Security* for information on securing JBoss EAP itself.
- See *How to Configure Identity Management* for information on providing security for applications deployed to JBoss EAP.
- See *How to Set Up SSO with Kerberos* for information on configuring single sign-on for JBoss EAP using Kerberos.
- See *How To Set Up SSO with SAML v2* for information on configuring single sign-on for JBoss EAP using SAML v2.

CHAPTER 6. JBOSS EAP CLASS LOADING

JBoss EAP uses a modular class loading system for controlling the class paths of deployed applications. This system provides more flexibility and control than the traditional system of hierarchical class loaders. Developers have fine-grained control of the classes available to their applications, and can configure a deployment to ignore classes provided by the application server in favor of their own.

The modular class loader separates all Java classes into logical groups called modules. Each module can define dependencies on other modules in order to have the classes from that module added to its own class path. Because each deployed JAR and WAR file is treated as a module, developers can control the contents of their application's class path by adding module configuration to their application.

6.1. MODULES

A module is a logical grouping of classes used for class loading and dependency management. JBoss EAP identifies two different types of modules: *static* and *dynamic*. The main difference between the two is how they are packaged.

Static Modules

Static modules are defined in the **`EAP_HOME/modules/`** directory of the application server. Each module exists as a subdirectory, for example **`EAP_HOME/modules/com/mysql/`**. Each module directory then contains a slot subdirectory, which defaults to **`main`** and contains the **`module.xml`** configuration file and any required JAR files. All the application server-provided APIs are provided as static modules, including the Java EE APIs as well as other APIs.

Example MySQL JDBC Driver `module.xml` File

```
<?xml version="1.0" ?>
<module xmlns="urn:jboss:module:1.1" name="com.mysql">
  <resources>
    <resource-root path="mysql-connector-java-5.1.36-bin.jar"/>
  </resources>
  <dependencies>
    <module name="javax.api"/>
    <module name="javax.transaction.api"/>
  </dependencies>
</module>
```

The module name (**`com.mysql`**) must match the directory structure for the module, excluding the slot name (**`main`**).

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 an administrator. The applications can then declare an explicit dependency on the custom static modules.

The modules provided in JBoss EAP distributions are located in the **`system`** directory within the **`EAP_HOME/modules`** directory. This keeps them separate from any modules provided by third parties. Any Red Hat provided products that layer on top of JBoss EAP also install their modules within the **`system`** directory.

Users must ensure that custom modules are installed into the **`EAP_HOME/modules`** directory, using one directory per module. 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.

6.2. MODULE DEPENDENCIES

A module dependency is a declaration that one module requires the classes of one or more other modules in order to function. When JBoss EAP 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.

NOTE

See the Modules section for complete details about modules and the modular class loading system.

Deployed applications (a JAR or WAR, for example) are loaded as dynamic modules and make use of dependencies to access the APIs provided by JBoss EAP.

There are two types of dependencies: *explicit* and *implicit*.

Explicit Dependencies

Explicit dependencies are declared by the developer in a configuration file. A static module can declare dependencies in its `module.xml` file. A dynamic module can declare dependencies in the deployment's `MANIFEST.MF` or `jboss-deployment-structure.xml` deployment descriptor.

Implicit Dependencies

Implicit dependencies are added automatically by JBoss EAP when certain conditions or meta-data are found in a deployment. The Java EE 7 APIs supplied with JBoss EAP are examples of modules that are added by detection of implicit dependencies in deployments.
Deployments can also be configured to exclude specific implicit dependencies by using the `jboss-deployment-structure.xml` deployment descriptor file. This can be useful when an application bundles a specific version of a library that JBoss EAP will attempt to add as an implicit dependency.

Optional Dependencies

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.

Export a 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.

For example, 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*.

Global Modules

A global module is a module that JBoss EAP provides as a dependency to every application. Any module can be made global by adding it to JBoss EAP's list of global modules. It does not require changes to the module.

See the Define Global Modules section for details.

6.3. CREATE A CUSTOM MODULE

Custom static modules can be added to make resources available for deployments running on JBoss EAP. You can create the module manually or by using the management CLI.

Once you create the module, you must add the module as a dependency if its resources need to be made available to applications.

Create a Custom Module Manually

You can create a custom module manually using the following steps.

1. Create the appropriate directory structure in the **`EAP_HOME/modules/`** directory.

   Example: Create MySQL JDBC Driver Directory Structure

   ```
   $ cd EAP_HOME/modules/
   $ mkdir -p com/mysql/main
   ```

2. Copy the JAR files or other necessary resources to the **`main/`** subdirectory.

   Example: Copy MySQL JDBC Driver JAR

   ```
   $ cp /path/to/mysql-connector-java-5.1.36-bin.jar
   EAP_HOME/modules/com/mysql/main/
   ```

3. Create a **`module.xml`** file in the **`main/`** subdirectory, specifying the appropriate resources and dependencies in the file.

   Example: MySQL JDBC Driver `module.xml` File

   ```
   <?xml version="1.0" ?>
   <module xmlns="urn:jboss:module:1.1" name="com.mysql">
     <resources>
       <resource-root path="mysql-connector-java-5.1.36-bin.jar"/>
   ```

```
  </resources>
  <dependencies>
    <module name="javax.api"/>
    <module name="javax.transaction.api"/>
  </dependencies>
</module>
```

Create a Custom Module Using the Management CLI

You can create a custom module using the **`module add`** management CLI command.

IMPORTANT

Using the **`module`** management CLI command to add and remove modules is provided as technology preview only. This command is not appropriate for use in a managed domain or when connecting to the management CLI remotely. Modules should be added and removed manually in a production environment.

1. Start the JBoss EAP server.
2. Launch the management CLI, but do not use the **`--connect`** or **`-c`** argument to connect to the running instance.

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

3. Use the **`module add`** management CLI command to add the new core module.

```
module add --name=MODULE_NAME --resources=PATH_TO_RESOURCE --
dependencies=DEPENDENCIES
```

Execute **`module --help`** for more details on using this command to add and remove modules.

Add the Module as a Dependency

In order for your application to be able to access this module's resources, you will need to add the module as a dependency.

- See the Add an Explicit Module Dependency to a Deployment section of the JBoss EAP *Development Guide* for adding application-specific dependencies using deployment descriptors.
- See the Define Global Modules section for instructions on adding modules as dependencies to all applications.

As an example, the following steps add a JAR file containing several properties files as a module and define a global module, so that an application can then load these properties.

1. Add the JAR file as a core module.

```
module add --name=myprops --resources=/path/to/properties.jar
```

2. Define this module as a global module so that it is made available to all deployments.

```
/subsystem=ee:list-add(name=global-modules,value={name=myprops})
```

3. The application could then retrieve the properties from one of the properties files contained within the JAR.

```
Thread.currentThread().getContextClassLoader().getResource("my.properties");
```

6.4. REMOVE A CUSTOM MODULE

Custom static modules can be removed manually or by using the management CLI.

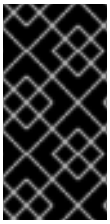
Remove a Custom Module Manually

Before manually removing a module, ensure that it is not required by deployed applications or elsewhere in the server configuration, such as by a datasource.

To remove a custom module, remove the module's directory under **`EAP_HOME/modules/`**, which includes its **`module.xml`** file and associated JAR files or other resources. For example, remove the **`EAP_HOME/modules/com/mysql/main/`** directory to remove a custom MySQL JDBC driver module in the **`main`** slot.

Remove a Custom Module Using the Management CLI

You can remove a custom module using the **`module remove`** management CLI command.



IMPORTANT

Using the **`module`** management CLI command to add and remove modules is provided as technology preview only. This command is not appropriate for use in a managed domain or when connecting to the management CLI remotely. Modules should be added and removed manually in a production environment.

1. Start the JBoss EAP server.
2. Launch the management CLI, but do not use the **`--connect`** or **`-c`** argument to connect to the running instance.

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

3. Use the **`module remove`** management CLI command to remove the custom module.

```
module remove --name=MODULE_NAME
```

- Use the **`--slot`** argument if the module to remove is in a slot other than **`main`**.

Example: Remove a MySQL Module

```
module remove --name=com.mysql
```

Execute **`module --help`** for more details on using this command to add and remove modules.

6.5. DEFINE GLOBAL MODULES

A list of global modules can be defined for JBoss EAP, which will add the modules as dependencies to all deployments.

NOTE

You must know the name of the modules that are to be configured as global modules. For the complete listing of the included modules and whether they are supported, see Red Hat JBoss Enterprise Application Platform 7 Included Modules on the Red Hat Customer Portal. See the Dynamic Module Naming section for naming conventions for modules in deployments.

Use the following management CLI command to define the list of global modules.

```
/subsystem=ee:write-attribute(name=global-modules,value=
[{name=MODULE_NAME_1},{name=MODULE_NAME_2}]
```

Use the following management CLI command to add a single module the list of existing global modules.

```
/subsystem=ee:list-add(name=global-modules,value={name=MODULE_NAME})
```

Global modules can also be added and removed using the management console by navigating to the **EE** subsystem from the **Configuration** tab and selecting the **Global Modules** section.

6.6. CONFIGURE SUBDEPLOYMENT ISOLATION

Each subdeployment in an Enterprise Archive (EAR) is a dynamic module with its own class loader. Subdeployments always have an implicit dependency on the parent module, which gives them access to classes in **EAR/lib**. By default, a subdeployment can access the resources of other subdeployments within that EAR.

If you do not want a subdeployment to be allowed to access classes belonging to other subdeployments, then strict subdeployment isolation can be enabled in JBoss EAP. This setting will affect all deployments.

Enable Subdeployment Module Isolation for All Deployments

Subdeployment isolation can be enabled or disabled using the management console or the management CLI from the **ee** subsystem. By default, subdeployment isolation is set to false, which allows the subdeployments to access resources of other subdeployments within an EAR deployment.

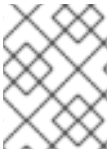
Use the following management CLI to enable EAR subdeployment isolation.

```
/subsystem=ee:write-attribute(name=ear-subdeployments-isolated,value=true)
```

Subdeployments in an EAR will no longer be able to access resources from other subdeployments.

6.7. DEFINE AN EXTERNAL JBOSS EAP MODULE DIRECTORY

The default directory for JBoss EAP modules is **EAP_HOME/modules**. You can specify a different directory for JBoss EAP modules using the **JBOSS_MODULEPATH** variable. Follow the below steps to set this variable in the JBoss EAP startup configuration file.



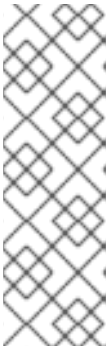
NOTE

You can also set **JBOSS_MODULEPATH** as an environment variable instead of setting this in the JBoss EAP startup configuration files.

1. Edit the startup configuration file.
   - When running as a standalone server, edit the **EAP_HOME/bin/standalone.conf** file (or **standalone.conf.bat** for Windows Server).
   - When running in a managed domain, edit the **EAP_HOME/bin/domain.conf** file (or **domain.conf.bat** for Windows Server).
2. Set the **JBOSS_MODULEPATH** variable, for example:

```
JBOSS_MODULEPATH="/path/to/modules/directory/"
```

To specify a list of directories use a colon (**:**) to delimit the list of directories.



NOTE

For Windows Server, use the following syntax to set the **JBOSS_MODULEPATH** variable:

```
set "JBOSS_MODULEPATH /path/to/modules/directory/"
```

To specify a list of directories use a semicolon (**;**) to delimit the list of directories.

6.8. DYNAMIC MODULE NAMING CONVENTIONS

JBoss EAP loads all deployments as modules, which are named according to the following conventions.

- Deployments of WAR and JAR files are named using 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 (EAR) are named using 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**.

CHAPTER 7. DEPLOYING APPLICATIONS

JBoss EAP features a range of application deployment and configuration options to cater to both administrators and developers. For administrators, the management console and the management CLI offer ideal graphical and command-line interfaces to manage application deployment in a production environment. For developers, the range of application deployment testing options include a configurable file system deployment scanner, the HTTP API, an IDE such as Red Hat JBoss Developer Studio, and Maven.

When deploying applications, you may want to enable validation for deployment descriptors by setting the **`org.jboss.metadata.parser.validate`** system property to **`true`**. This can be done one of the following ways:

- While starting the server.

```
$ EAP_HOME/bin/standalone.sh -
Dorg.jboss.metadata.parser.validate=true
```

- By adding it to the server configuration with the following management CLI command.

```
/system-property=org.jboss.metadata.parser.validate:add(value=true)
```

7.1. DEPLOYING APPLICATIONS USING THE MANAGEMENT CLI

Deploying applications using 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 deployments for a single server when running as a standalone server, or an entire network of servers when running in a managed domain.

7.1.1. Deploy an Application to a Standalone Server Using the Management CLI

Deploy an Application
From the management CLI, use the **`deploy`** command and specify the path to the application deployment.

```
deploy /path/to/test-application.war
```

A successful deployment does not produce any output to the management CLI, but the server log displays deployment messages.

```
WFLYSRV0027: Starting deployment of "test-application.war" (runtime-name:
"test-application.war")
WFLYUT0021: Registered web context: /test-application
WFLYSRV0010: Deployed "test-application.war" (runtime-name : "test-
application.war")
```

The application has been successfully deployed.

Undeploy an Application
From the management CLI, use the **`undeploy`** command and specify the deployment name.

- Undeploy the application and delete the deployment content.

```
undeploy test-application.war
```

- Undeploy the application without removing the deployment content from the repository.

  ```
  undeploy test-application.war --keep-content
  ```

  This is the same as disabling the deployment from the management console.

A successful undeployment does not produce any output to the management CLI, but the server log displays undeployment messages.

```
WFLYUT0022: Unregistered web context: /test-application
WFLYSRV0028: Stopped deployment test-application.war (runtime-name: test-
application.war) in 62ms
WFLYSRV0009: Undeployed "test-application.war" (runtime-name: "test-
application.war")
```

The application has been successfully undeployed.

List Deployments
From the management CLI, use the **`deployment-info`** command to list deployment information.

```
deployment-info
```

The output will show details about each deployment, such as the runtime name, status, and whether it is enabled.

```
NAME                 RUNTIME-NAME         PERSISTENT ENABLED STATUS
jboss-helloworld.war jboss-helloworld.war true       true    OK
test-application.war test-application.war true       true    OK
```

You can also filter the deployments to display by name using the **`--name`** argument.

7.1.2. Deploy an Application in a Managed Domain Using the Management CLI

Deploy an Application
From the management CLI, use the **`deploy`** command and specify the path to the application deployment. You must also specify the server groups to which the application should be deployed.

- To deploy the application to all server groups.

  ```
  deploy /path/to/test-application.war --all-server-groups
  ```

- To deploy the application to specific server groups.

  ```
  deploy /path/to/test-application.war --server-groups=main-server-
  group,other-server-group
  ```

A successful deployment does not produce any output to the management CLI, but the server log displays deployment messages for each affected server.

```
[Server:server-one] WFLYSRV0027: Starting deployment of "test-
application.war" (runtime-name: "test-application.war")
```

```
[Server:server-one] WFLYUT0021: Registered web context: /test-application
[Server:server-one] WFLYSRV0010: Deployed "test-application.war" (runtime-
name : "test-application.war")
```

The application has been successfully deployed to the appropriate server groups in your managed domain.

Undeploy an Application
From the management CLI, use the **undeploy** command and specify the deployment name. You must also specify the server groups from which the application should be undeployed.

- Undeploy the application from all server groups with that deployment.

  ```
  undeploy test-application.war --all-relevant-server-groups
  ```

- Undeploy the application from specific server groups. Note that the **--keep-content** parameter is required, as the content must remain in the repository for other server groups with that deployment.

  ```
  undeploy test-application.war --server-groups=other-server-group --
  keep-content
  ```

  This is the same as disabling the deployment from the management console.

A successful undeployment does not produce any output to the management CLI, but the server log displays undeployment messages for each affected server.

```
[Server:server-one] WFLYUT0022: Unregistered web context: /test-
application
[Server:server-one] WFLYSRV0028: Stopped deployment test-application.war
(runtime-name: test-application.war) in 74ms
[Server:server-one] WFLYSRV0009: Undeployed "test-application.war"
(runtime-name: "test-application.war")
```

The application has been successfully undeployed.

List Deployments
From the management CLI, use the **deployment-info** command to list deployment information. You can list deployment information by deployment name or by server group.

To display deployment information by name:

```
deployment-info --name=jboss-helloworld.war
```

The output will list the deployment and its state in each server group.

```
NAME                 RUNTIME-NAME
jboss-helloworld.war jboss-helloworld.war

SERVER-GROUP       STATE
main-server-group  enabled
other-server-group added
```

To display deployment information by server group:

```
deployment-info --server-group=other-server-group
```

The output will list the deployments and their state for the specified server group.

```
NAME                 RUNTIME-NAME         STATE
jboss-helloworld.war jboss-helloworld.war added
test-application.war test-application.war enabled
```

You can also list all deployments in the domain using the **`deploy -l`** command.

7.2. DEPLOYING APPLICATIONS USING THE MANAGEMENT CONSOLE

Deploying applications using the management console gives you the benefit of a graphical interface that is easy to use. You can see at a glance which applications are deployed to your server or server groups, and you can enable, disable or remove applications from the content repository as required.

7.2.1. Deploy an Application to a Standalone Server Using the Management Console

Deployments can be viewed and managed from the **Deployments** tab of the JBoss EAP management console.

Deploy an Application
Click the **Add** button and use the **New Deployment** wizard to deploy an application. You can choose to deploy an application by *uploading a deployment* or *creating an unmanaged deployment*. Deployments are enabled by default.

- Upload a deployment
  Upload an application that will be copied to the server's content repository and managed by JBoss EAP.

- Create an unmanaged deployment
  Specify the location of a deployment. This deployment will not be copied to the server's content repository and will not be managed by JBoss EAP. Note that exploded deployments are only supported as unmanaged.

Undeploy an Application
Select the deployment and choose the **Remove** option to undeploy the application. This undeploys the deployment and removes it from the content repository.

Disable an Application
Select the deployment and choose the **Disable** option to disable the application. This undeploys the deployment, but does not remove it from the content repository.

Replace an Application
Select the deployment and choose the **Replace** option. Select the new version of the deployment, which must have the same name as the original, and click **Finish**. This undeploys and removes the original version of the deployment, and then deploys the new version.

7.2.2. Deploy an Application in a Managed Domain Using the Management Console

From the **Deployments** tab of the JBoss EAP management console, deployments can be viewed and managed by:

- Content Repository

All managed and unmanaged deployments are listed in the **Content Repository** section. Deployments can be added and assigned to server groups here.

- Unassigned Content
  Deployments that have not been assigned to any server groups are listed in the **Unassigned Content** section. Deployments can be assigned to server groups or removed here.

- Server Groups
  Deployments that have been assigned to one or more server groups are listed in the **Server Groups** section. Deployments can be enabled and added directly to a server group here.

Deploy an Application

1. From **Content Repository**, click the **Add** button.

2. Choose to deploy an application by *uploading a deployment* or *creating an unmanaged deployment*.

3. Follow the prompts to deploy the application.
   Note that a deployment must be assigned to a server group before it can be enabled.

Deployments can also be added, assigned to a server group, and enabled in one step by adding the deployment from **Server Groups**.

Assign an Application to a Server Group

1. From **Unassigned Content**, select a deployment and click the **Assign** button.

2. Select one or more server groups to which this deployment should be assigned.

3. Optionally, select the option to enable the deployment on the selected server groups.

Unassign an Application from a Server Group

1. From **Server Groups**, select the appropriate server group.

2. Select the desired deployment and click the **Unassign** button.

Deployments can also be unassigned from multiple server groups at once by selecting the **Unassign** button for the deployment in **Content Repository**.

Undeploy an Application

1. If the deployment is still assigned to any server groups, be sure to unassign the deployment.

2. From **Content Repository**, select the deployment and click the **Remove** button.

This undeploys the deployment and removes it from the content repository.

Disable an Application

1. From **Server Groups**, select the appropriate server group.

2. Select the desired deployment and click the **Disable** button.

This undeploys the deployment, but does not remove it from the content repository.

Replace an Application

1. From **Content Repository**, select the deployment and click the **Replace** button.
2. Select the new version of the deployment, which must have the same name as the original, and click **Finish**.

This undeploys and removes the original version of the deployment, and then deploys the new version.

7.3. DEPLOYING APPLICATIONS USING THE DEPLOYMENT SCANNER

The deployment scanner monitors the deployment directory for applications to deploy. By default, the deployment scanner scans the **`EAP_HOME/standalone/deployments/`** directory every five seconds for changes. Marker files are used to indicate the status of a deployment and to trigger actions against deployments, such as undeploying or redeploying.

While it is recommended to use the management console or management CLI for application deployment in a production environment, deploying using the deployment scanner is provided for the convenience of developers. This allows users build and test applications in a manner suited for rapid development cycles. Additionally, the deployment scanner should not be used in conjunction with other deployment methods.

The deployment scanner is only available when running JBoss EAP as a standalone server.

7.3.1. Deploy an Application to a Standalone Server Using the Deployment Scanner

The deployment scanner can be configured to allow or disallow automatic deployment of XML, zipped, and exploded content. If automatic deployment is disabled, you must manually create marker files to trigger deployment actions. For more information about the available marker file types and their purposes, see the Deployment Scanner Marker Files section.

By default, automatic deployment for XML and zipped content is enabled. For details on configuring automatic deployment for each content type, see Configure the Deployment Scanner.

WARNING

Deploying using the deployment scanner is provided for the convenience of developers and is not recommended for use in a production environment. It should also not be used in conjunction with other deployment methods.

Deploy an Application
Copy the content to the deployment folder.

```
$ cp /path/to/test-application.war EAP_HOME/standalone/deployments/
```

If auto-deployment is enabled, this file will be picked up automatically, deployed, and a **`.deployed`** marker file will be created. If auto-deployment is not enabled, then you will need to manually add a **`.dodeploy`** marker file to trigger deployment.

```
$ touch EAP_HOME/standalone/deployments/test-application.war.dodeploy
```

Undeploy an Application
Trigger an undeployment by removing the **.deployed** marker file.

```
$ rm EAP_HOME/standalone/deployments/test-application.war.deployed
```

If auto-deployment is enabled, you can also remove the **test-application.war** file, which will trigger the undeployment. Note that this does not apply for exploded deployments.

Redeploy an Application
Create a **.dodeploy** marker file to initiate redeployment.

```
$ touch EAP_HOME/standalone/deployments/test-application.war.dodeploy
```

7.3.2. Configure the Deployment Scanner

The deployment scanner can be configured using the management console or the management CLI. You can configure the deployment scanner's behavior, such as the scan interval, deployment folder location, and autodeployment of certain application file types. You can also disable the deployment scanner entirely.

For details on all available deployment scanner attributes, see the Deployment Scanner Attributes section.

Use the below management CLI commands to configure the default deployment scanner.

Disable the Deployment Scanner

```
/subsystem=deployment-scanner/scanner=default:write-attribute(name=scan-
enabled,value=false)
```

This disables the **default** deployment scanner.

Change the Scan Interval

```
/subsystem=deployment-scanner/scanner=default:write-attribute(name=scan-
interval,value=10000)
```

This updates the scan interval time from **5000** milliseconds (five seconds) to **10000** milliseconds (ten seconds).

Change the Deployment Folder

```
/subsystem=deployment-scanner/scanner=default:write-
attribute(name=path,value=/path/to/deployments)
```

This changes the location of the deployment folder from the default location of **EAP_HOME/standalone/deployments** to **/path/to/deployments**.

The **path** value will be treated as an absolute path unless the **relative-to** attribute is specified, in which case it will be relative to that path.

Enable the Automatic Deployment of Exploded Content

```
/subsystem=deployment-scanner/scanner=default:write-attribute(name=auto-
deploy-exploded,value=true)
```

This enables the automatic deployment of exploded content, which is disabled by default.

Disable the Automatic Deployment of Zipped Content

```
/subsystem=deployment-scanner/scanner=default:write-attribute(name=auto-
deploy-zipped,value=false)
```

This disables the automatic deployment of zipped content, which is enabled by default.

Disable the Automatic Deployment of XML Content

```
/subsystem=deployment-scanner/scanner=default:write-attribute(name=auto-
deploy-xml,value=false)
```

This disables the automatic deployment of XML content, which is enabled by default.

7.3.3. Define a Custom Deployment Scanner

A new deployment scanner can be added using the management CLI or by navigating to the **Deployment Scanners** subsystem from the **Configuration** tab in the management console. This will define a new directory to scan for deployments. The default deployment scanner monitors **`EAP_HOME/standalone/deployments`**. See Configure the Deployment Scanner for details on configuring an existing deployment scanner.

The following management CLI command adds a new deployment scanner that will check **`EAP_HOME/standalone/new_deployment_dir`** every five seconds for deployments.

```
/subsystem=deployment-scanner/scanner=new-
scanner:add(path=new_deployment_dir,relative-
to=jboss.server.base.dir,scan-interval=5000)
```

NOTE

The specified directory must already exist or this command will fail with an error.

A new deployment scanner has been defined and the specified directory will be monitored for deployments.

7.4. DEPLOYING APPLICATIONS USING MAVEN

Deploying applications using Apache Maven allows you to easily incorporate deployment to JBoss EAP into your existing development workflow.

You can use Maven to deploy applications to JBoss EAP using the WildFly Maven Plugin, which provides simple operations to deploy and undeploy applications to the application server.

7.4.1. Deploy an Application to a Standalone Server Using Maven

The following instructions show how to deploy and undeploy the JBoss EAP **`helloworld`** quickstart to a standalone server using Maven.

See Using the Quickstart Examples in the JBoss EAP *Getting Started Guide* for more information on the JBoss EAP quickstarts.

Deploy an Application
Initialize the WildFly Maven Plugin in your Maven **pom.xml** file. This should already be configured in the JBoss EAP quickstart **pom.xml** files.

```
<plugin>
  <groupId>org.wildfly.plugins</groupId>
  <artifactId>wildfly-maven-plugin</artifactId>
  <version>${version.wildfly.maven.plugin}</version>
</plugin>
```

From the **helloworld** quickstart directory, execute the following Maven command.

```
$ mvn clean install wildfly:deploy
```

After issuing the Maven command to deploy, the terminal window shows the following output indicating a successful deployment.

```
[INFO] ------------------------------------------------------------------
---------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------
---------
[INFO] Total time: 2.981 s
[INFO] Finished at: 2015-12-23T15:06:13-05:00
[INFO] Final Memory: 21M/231M
[INFO] ------------------------------------------------------------------
---------
```

The deployment can also be confirmed by viewing the server log of the active server instance.

```
WFLYSRV0027: Starting deployment of "jboss-helloworld.war" (runtime-name:
"jboss-helloworld.war")
WFLYUT0021: Registered web context: /jboss-helloworld
WFLYSRV0010: Deployed "jboss-helloworld.war" (runtime-name : "jboss-
helloworld.war")
```

Undeploy an Application
From the **helloworld** quickstart directory, execute the following Maven command.

```
$ mvn wildfly:undeploy
```

After issuing the Maven command to undeploy, the terminal window shows the following output indicating a successful undeployment.

```
[INFO] ------------------------------------------------------------------
---------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------
---------
[INFO] Total time: 1.237 s
[INFO] Finished at: 2015-12-23T15:09:10-05:00
```

```
[INFO] Final Memory: 10M/183M
[INFO] ------------------------------------------------------------------------
---------
```

The undeployment can also be confirmed by viewing the server log of the active server instance.

```
WFLYUT0022: Unregistered web context: /jboss-helloworld
WFLYSRV0028: Stopped deployment jboss-helloworld.war (runtime-name: jboss-
helloworld.war) in 27ms
WFLYSRV0009: Undeployed "jboss-helloworld.war" (runtime-name: "jboss-
helloworld.war")
```

7.4.2. Deploy an Application in a Managed Domain Using Maven

The following instructions show how to deploy and undeploy the JBoss EAP **`helloworld`** quickstart in a managed domain using Maven.

See Using the Quickstart Examples in the JBoss EAP *Getting Started Guide* for more information on the JBoss EAP quickstarts.

Deploy an Application
When deploying an application in a managed domain, you must specify the server groups to which the application should be deployed. This is configured in the Maven **`pom.xml`** file.

The following configuration in the **`pom.xml`** initializes the WildFly Maven Plugin and specifies **`main-server-group`** as the server group to which the application should be deployed.

```
<plugin>
  <groupId>org.wildfly.plugins</groupId>
  <artifactId>wildfly-maven-plugin</artifactId>
  <version>${version.wildfly.maven.plugin}</version>
  <configuration>
    <domain>
      <server-groups>
        <server-group>main-server-group</server-group>
      </server-groups>
    </domain>
  </configuration>
</plugin>
```

From the **`helloworld`** quickstart directory, execute the following Maven command.

```
$ mvn clean install wildfly:deploy
```

After issuing the Maven command to deploy, the terminal window shows the following output indicating a successful deployment.

```
[INFO] ------------------------------------------------------------------------
---------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
---------
[INFO] Total time: 4.005 s
[INFO] Finished at: 2016-09-02T14:36:17-04:00
```

```
[INFO] Final Memory: 21M/226M
[INFO] ------------------------------------------------------------------------
---------
```

The deployment can also be confirmed by viewing the server log of the active server instance.

```
WFLYSRV0027: Starting deployment of "jboss-helloworld.war" (runtime-name:
"jboss-helloworld.war")
WFLYUT0021: Registered web context: /jboss-helloworld
WFLYSRV0010: Deployed "jboss-helloworld.war" (runtime-name : "jboss-
helloworld.war")
```

Undeploy an Application
From the **helloworld** quickstart directory, execute the following Maven command.

```
$ mvn wildfly:undeploy
```

After issuing the Maven command to undeploy, the terminal window shows the following output indicating a successful undeployment.

```
[INFO] ------------------------------------------------------------------------
---------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
---------
[INFO] Total time: 1.750 s
[INFO] Finished at: 2016-09-02T14:45:10-04:00
[INFO] Final Memory: 10M/184M
[INFO] ------------------------------------------------------------------------
---------
```

The undeployment can also be confirmed by viewing the server log of the active server instance.

```
WFLYUT0022: Unregistered web context: /jboss-helloworld
WFLYSRV0028: Stopped deployment jboss-helloworld.war (runtime-name: jboss-
helloworld.war) in 106ms
WFLYSRV0009: Undeployed "jboss-helloworld.war" (runtime-name: "jboss-
helloworld.war")
```

7.5. DEPLOYING APPLICATIONS USING THE HTTP API

Applications can be deployed to JBoss EAP using the HTTP API with the **curl** command. For more information on using the HTTP API, see the HTTP API section.

7.5.1. Deploy an Application to a Standalone Server Using the HTTP API

By default, the HTTP API is accessible at **http://*HOST*:*PORT*/management**, for example, **http://localhost:9990/management**.

Deploy an Application

```
$ curl --digest -L -D - http://HOST:PORT/management --header "Content-
Type: application/json" -u USER:PASSWORD -d '{"operation" : "composite",
```

```
"address" : [], "steps" : [{"operation" : "add", "address" : {"deployment"
: "test-application.war"}, "content" : [{"url" : "file:/path/to/test-
application.war"}]},{"operation" : "deploy", "address" : {"deployment" :
"test-application.war"}}],"json.pretty":1}'
```

Undeploy an Application

```
$ curl --digest -L -D - http://HOST:PORT/management --header "Content-
Type: application/json" -u USER:PASSWORD -d '{"operation" : "composite",
"address" : [], "steps" : [{"operation" : "undeploy", "address" :
{"deployment" : "test-application.war"}},{"operation" : "remove",
"address" : {"deployment" : "test-application.war"}}],"json.pretty":1}'
```

See this Red Hat Knowledgebase article to learn more about programmatically generating the JSON requests.

7.5.2. Deploy an Application in a Managed Domain Using the HTTP API

By default, the HTTP API is accessible at **`http://HOST:PORT/management`**, for example, **`http://localhost:9990/management`**.

Deploy an Application

1. Add the deployment to the content repository.

```
$ curl --digest -L -D - http://HOST:PORT/management --header
"Content-Type: application/json" -u USER:PASSWORD -d '{"operation" :
"add", "address" : {"deployment" : "test-application.war"},
"content" : [{"url" : "file:/path/to/test-
application.war"}],"json.pretty":1}'
```

2. Add the deployment to the desired server group.

```
$ curl --digest -L -D - http://HOST:PORT/management --header
"Content-Type: application/json" -u USER:PASSWORD -d '{"operation" :
"add", "address" : {"server-group" : "main-server-
group","deployment":"test-application.war"},"json.pretty":1}'
```

3. Deploy the application to the server group.

```
$ curl --digest -L -D - http://HOST:PORT/management --header
"Content-Type: application/json" -u USER:PASSWORD -d '{"operation" :
"deploy", "address" : {"server-group" : "main-server-
group","deployment":"test-application.war"},"json.pretty":1}'
```

Undeploy an Application

1. Remove the deployment from all server groups to which it is assigned.

```
$ curl --digest -L -D - http://HOST:PORT/management --header
"Content-Type: application/json" -u USER:PASSWORD -d '{"operation" :
"remove", "address" : {"server-group" : "main-server-
group","deployment":"test-application.war"},"json.pretty":1}'
```

2. Remove the deployment from the content repository.

```
$ curl --digest -L -D - http://HOST:PORT/management --header
"Content-Type: application/json" -u USER:PASSWORD -d '{"operation" :
"remove", "address" : {"deployment" : "test-application.war"},
"json.pretty":1}'
```

7.6. CUSTOMIZING DEPLOYMENT BEHAVIOR

7.6.1. Define a Custom Directory for Deployment Content

You can define a custom location for JBoss EAP to store deployed content.

Define a Custom Directory for a Standalone Server
By default, deployed content for a standalone server 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.

```
$ EAP_HOME/bin/standalone.sh -
Djboss.server.deploy.dir=/path/to/new_deployed_content
```

The chosen location should be unique among JBoss EAP instances.

NOTE

The **`jboss.server.deploy.dir`** property specifies the directory to be used for storing content that has been deployed using the management console or management CLI. To define a custom deployment directory to be monitored by the deployment scanner, see Configure the Deployment Scanner.

Define a Custom Directory for a Managed Domain
By default, deployed content for a managed domain 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.

```
$ EAP_HOME/bin/domain.sh -
Djboss.domain.deployment.dir=/path/to/new_deployed_content
```

The chosen location should be unique among JBoss EAP instances.

7.6.2. Control the Order of Deployments

JBoss EAP offers fine-grained control over the order of deployments when the server is started. Strict order of the deployment of applications present in multiple EAR files can be specified along with persistence of the order after a restart.

You can use the **`jboss-all.xml`** deployment descriptor to declare dependencies between top-level deployments.

For example, if you have an **`app.ear`** that depends on **`framework.ear`** being deployed first, then you can create an **`app.ear/META-INF/jboss-all.xml`** file as shown below.

```
<jboss umlns="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.

This ensures that **`framework.ear`** is deployed before **`app.ear`**.

IMPORTANT

Although the **`jboss-all.xml`** file and other deployment descriptors allow 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 deployment descriptor have already been deployed or are available. If there are missing dependencies, JBoss EAP does not automatically deploy them, and the deployment fails.

7.6.3. Override Deployment Content

A *deployment overlay* can be used to overlay content into an existing deployment without physically modifying the contents of the deployment archive. It allows you to override deployment descriptors, JAR files, classes, JSP pages, and other files at runtime without rebuilding the archive.

This can be useful if you need to adapt a deployment for different environments that need different configurations or settings. For example, when moving a deployment through the application lifecycle from development, to testing, to stage, and finally into production, you might want to swap deployment descriptors, modify static web resources to change the branding of the application, or even replace JAR libraries with different versions depending on the target environment. It is also a useful feature for installations that need to change a configuration but can not modify or crack an archive due to policy or security restrictions.

When defining a deployment overlay, you specify the file on a file system that will replace the file in the deployment archive. You must also specify which deployments should be affected by the deployment overlay. Any affected deployments must be redeployed in order for the changes to take effect.

Use the **`deployment-overlay add`** management CLI command to add a deployment overlay.

```
deployment-overlay add --name=new-deployment-overlay --content=WEB-
INF/web.xml=/path/to/other/web.xml --deployments=test-application.war --
redeploy-affected
```

NOTE

In a managed domain, specify the applicable server groups by using **`--server-groups`** or specify all server groups with **`--all-server-groups`**.

Once created, you can add content to an existing overlay, link the overlay to a deployment, or remove the overlay. For full usage details, execute **`deployment-overlay --help`**.

Parameters

`name`

The name of the deployment overlay.

`content`

Comma-separated list that maps the file on the file system to the file in the archive that it will replace. The format for each entry is **`ARCHIVE_PATH=FILESYSTEM_PATH`**.

`deployments`

Comma-separated list of deployments to which this overlay will be linked.

`redeploy-affected`

Redeploys all affected deployments.

7.6.4. Using Rollout Plans

About Rollout Plans
In a managed domain, 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. If no rollout plan is specified, the default rollout plan is used.

Below is an example rollout plan involving five server groups. Operations can be applied to server groups serially (**`in-series`**) or concurrently (**`concurrent-groups`**). The syntax is described in more detail in Rollout Plan Syntax.

```
{"my-rollout-plan" => {"rollout-plan" => {
    "in-series" => [
        {"concurrent-groups" => {
            "group-A" => {
                "max-failure-percentage" => "20",
                "rolling-to-servers" => "true"
            },
            "group-B" => undefined
        }},
        {"server-group" => {"group-C" => {
            "rolling-to-servers" => "false",
            "max-failed-servers" => "1"
        }}},
        {"concurrent-groups" => {
            "group-D" => {
                "max-failure-percentage" => "20",
                "rolling-to-servers" => "true"
            },
            "group-E" => undefined
        }}
    ],
    "rollback-across-groups" => "true"
}}}
```

Looking at the example above, applying the operation to the servers in the domain is done in three 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.

1. Server groups *group-A* and *group-B* will have the operation applied concurrently. The operation will be applied to the servers in *group-A* in series, while all servers in *group-B* will handle the operation concurrently. If more than 20% of the servers in *group-A* fail to apply the operation, it will be rolled back across that group. If any servers in *group-B* fail to apply the operation it will be rolled back across that group.

2. Once all servers in *group-A* and *group-B* are complete, the operation will be applied to the servers in *group-C*. Those servers will handle the operation concurrently. If more than one server in *group-C* fails to apply the operation it will be rolled back across that group.

3. Once all servers in *group-C* are complete, server groups *group-D* and *group-E* will have the operation applied concurrently. The operation will be applied to the servers in *group-D* in series, while all servers in *group-E* will handle the operation concurrently. If more than 20% of the servers in *group-D* fail to apply the operation, it will be rolled back across that group. If any servers in *group-E* fail to apply the operation it will be rolled back across that group.

Rollout Plan Syntax
You can specify a rollout plan in either of the following ways.

- By defining the rollout plan in the **`deploy`** command operation headers. See Deploy Using a Rollout Plan for details.

- By storing the rollout plan using the **`rollout-plan`** command and then referencing the plan name in the **`deploy`** command operation headers. See Deploy Using a Stored Rollout Plan for details.

Although each method has a different initial command, both methods use the **`rollout`** operation header to define the rollout plan. This uses the following syntax.

```
rollout (id=PLAN_NAME | SERVER_GROUP_LIST) [rollback-across-groups]
```

- **`PLAN_NAME`** is the name for the rollout plan that was stored using the **`rollout-plan`** command.

- **`SERVER_GROUP_LIST`** is the list of server groups. Use a comma (**`,`**) to separate multiple server groups to indicate that that operations should be performed on each server group sequentially. Use a caret (^) separator to indicate that operations should be performed on each server group concurrently.

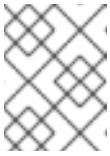
  - For each server group, set any of the following policies in parentheses. Comma separate multiple policies.

    - **`rolling-to-servers`**: A boolean which if set to **`true`**, the operation will be applied to each server in the group in series. If the value is **`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 **`0`**, meaning that a failure on any server will trigger rollback across the group.

    - **`max-failure-percentage`**: An integer between **`0`** and **`100`** that represents 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 **0**, meaning that a failure on any server will trigger rollback across the group.



NOTE

If both **max-failed-servers** and **max-failure-percentage** are set to non-zero values, **max-failure-percentage** takes precedence.

- **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 defaults to **false**.

Deploy Using a Rollout Plan

You can provide the full details of a rollout plan directly to the **deploy** command by passing the **rollout** settings into the **headers** argument. See the Rollout Plan Syntax for more information on the format.

The following management CLI command deploys an application to the **main-server-group** server group using a deployment plan that specifies **rolling-to-servers=true** for serial deployment.

```
deploy /path/to/test-application.war --server-groups=main-server-group --
headers={rollout main-server-group(rolling-to-servers=true)}
```

Deploy Using a Stored Rollout Plan

Since rollout plans can be complex, you have the option to store the details of a rollout plan. This allows you to reference the rollout plan name when you want to use it instead of requiring the full details of the rollout plan each time.

1. Use the **rollout-plan** management CLI command to store a rollout plan. See the Rollout Plan Syntax for more information on the format.

   ```
   rollout-plan add --name=my-rollout-plan --content={rollout main-
   server-group(rolling-to-servers=false,max-failed-servers=1),other-
   server-group(rolling-to-servers=true,max-failure-percentage=20)
   rollback-across-groups=true}
   ```

   This creates the following deployment plan.

   ```
   "rollout-plan" => {
       "in-series" => [
           {"server-group" => {"main-server-group" => {
               "rolling-to-servers" => false,
               "max-failed-servers" => 1
           }}},
           {"server-group" => {"other-server-group" => {
               "rolling-to-servers" => true,
               "max-failure-percentage" => 20
           }}}
       ],
       "rollback-across-groups" => true
   }
   ```

2. Specify the stored rollout plan name when deploying the application.
   The following management CLI command deploys an application to all server groups using the **`my-rollout-plan`** stored rollout plan.

```
deploy /path/to/test-application.war --all-server-groups --headers=
{rollout id=my-rollout-plan}
```

Remove a Stored Rollout Plan
You can remove a stored rollout plan using the **`rollout-plan`** management CLI command by specifying the name of the rollout plan to remove.

```
rollout-plan remove --name=my-rollout-plan
```

Default Rollout Plan
All operations that impact multiple servers will be executed with a rollout plan. If no rollout plan is specified in the operation request, a default rollout 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.

CHAPTER 8. DOMAIN MANAGEMENT

This section discusses concepts and configuration specific to the managed domain operating mode.

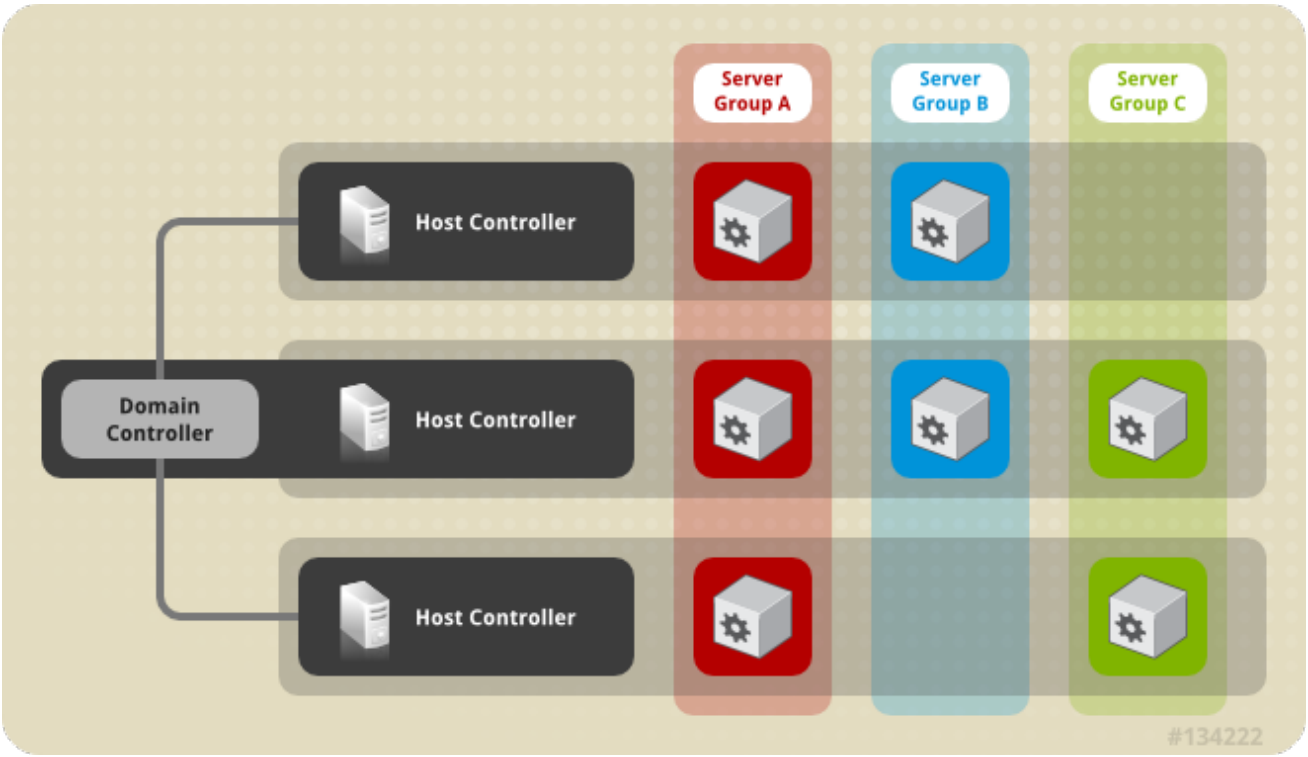
For information on securing a managed domain, see the Securing a Managed Domain section of the JBoss EAP *How to Configure Server Security* guide.

8.1. ABOUT MANAGED DOMAINS

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

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

A domain consists of one domain controller, one or more host controllers, 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 that 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 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.

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. It is possible for a domain controller, a single host controller, and multiple servers to run within

the same JBoss EAP instance, on the same physical system.

8.1.1. About the Domain Controller

A domain controller is the JBoss EAP 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 server instances are configured in accordance with this policy.

By default, the central management policy is stored in the **`EAP_HOME/domain/configuration/domain.xml`** file. This file is required in this directory of the host controller that is set to run as the domain controller.

The **`domain.xml`** file contains profile configurations available for use by the servers in the domain. A profile contains the detailed settings of the various subsystems available in that profile. The domain configuration also includes the definition of socket groups and the server group definitions.

NOTE

It is possible for a JBoss EAP 7 domain controller to administer JBoss EAP 6 hosts and servers, as long as the hosts and servers are running JBoss EAP 6.2 or later. For more information, see Configure a JBoss EAP 7 Domain Controller to Administer JBoss EAP 6 Instances.

For more information, see the Start a Managed Domain and Configure the Domain Controller sections.

8.1.2. About Host Controllers

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 **`EAP_HOME/domain/configuration/host.xml`** file located in the unzipped JBoss EAP installation file on its host's file system. The **`host.xml`** file contains the following configuration information that is specific to the particular host:

- The names of the server instances meant to run from this installation.
- 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 file system paths in **`host.xml`**.

- 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.
  - Whether the host controller is to act as the domain controller

For more information, see the Start a Managed Domain and Configure Host Controllers sections.

8.1.3. About Process Controllers

A process controller is a small, lightweight process that is responsible for spawning the host controller process and monitoring its lifecycle. If the host controller crashes, the process controller will restart it. It also starts server processes as directed by the host controller; however, it will not automatically restart server processes that crash.

The process controller logs to the **`EAP_HOME/domain/log/process-controller.log`** file. You can set JVM options for the process controller in the **`EAP_HOME/bin/domain.conf`** file using the **`PROCESS_CONTROLLER_JAVA_OPTS`** variable.

8.1.4. 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.

For more information, see the Configuring Server Groups section.

8.1.5. About Servers

A server represents an application server instance. In a managed domain, all server instances are members of a server group. The host controller launches each server instance in its own JVM process.

For more information, see the Configuring Servers section.

8.2. NAVIGATING DOMAIN CONFIGURATIONS

JBoss EAP provides scalable management interfaces to support both small and large-scale managed domains.

Management Console

The JBoss EAP management console provides a graphical view of your domain and allows you to easily manage hosts, servers, deployments, and profiles for your domain.

Configuration

From the **Configuration** tab, you can configure the subsystems for each profile used in your domain. Different server groups in your domain may use different profiles depending the capabilities needed.
Once you select the desired profile, all available subsystems for that profile are listed. For more information on configuring profiles, see Managing JBoss EAP Profiles.

Runtime

From the **Runtime** tab, you can manage servers and server groups as well as host configuration. You can browse the domain by host or by server group.
From **Hosts**, you can configure host properties and JVM settings as well as add and configure servers for that host.

From **Server Groups**, you can add new server groups and configure properties and JVM settings as well as add and configure servers for that server group. You can perform operations such as starting, stopping, suspending, and reloading all servers in the selected server group.

From either **Hosts** or **Server Groups**, you can add new servers and configure server properties and JVM settings. You can perform operations such as starting, stopping, suspending, and reloading the selected server. You can also view runtime information, such as JVM usage, server logs, and subsystem-specific information.

Deployments

From the **Deployments** tab, you can add and assign deployments to server groups. You can view all deployments in the content repository, view all unassigned deployments, or view deployments assigned to a particular server group.
For more information on deploying applications using the management console, see Deploy an Application in a Managed Domain

Management CLI

The JBoss EAP management CLI provides a command-line interface to manage hosts, servers, deployments and profiles for your domain.

Subsystem configuration can be accessed once the appropriate profile is selected.

```
/profile=PROFILE_NAME/subsystem=SUBSYSTEM_NAME:read-
resource(recursive=true)
```

NOTE

Instructions and examples throughout this guide may contain management CLI commands for subsystem configuration that apply when running as a standalone server, for example:

```
/subsystem=datasources/data-source=ExampleDS:read-resource
```

To adjust these management CLI commands to be run in a managed domain, you must specify the appropriate profile to configure, for example:

```
/profile=default/subsystem=datasources/data-
source=ExampleDS:read-resource
```

After specifying the appropriate host, you can configure host settings and perform operations on servers on that host.

```
/host=HOST_NAME/server=SERVER_NAME:read-resource
```

After specifying the appropriate host, you can configure servers for that host.

```
/host=HOST_NAME/server-config=SERVER_NAME:write-
attribute(name=ATTRIBUTE_NAME,value=VALUE)
```

After specifying the appropriate server group, you can configure server group settings and perform operations on all servers in the selected server group.

```
/server-group=SERVER_GROUP_NAME:read-resource
```

You can deploy applications in a managed domain by using the **deploy** management CLI command and specifying the appropriate server groups. For instructions, see Deploy an Application in a Managed Domain

8.3. LAUNCHING A MANAGED DOMAIN

8.3.1. Start a Managed Domain

Domain and host controllers can be started using the **domain.sh** or **domain.bat** script provided with JBoss EAP. For a complete listing of all available startup script arguments and their purposes, use the **--help** argument or see the Server Runtime Arguments section.

The domain controller must be started before any slave servers in any server groups in the domain. Start the domain controller first, then start any other associated host controllers in the domain.

Start the Domain Controller
Start the domain controller with the **host-master.xml** configuration file, which is preconfigured for a dedicated domain controller.

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

Depending on your domain setup, you will need to make additional configurations to allow host controllers to connect. Also see the following example domain setups:

- Set Up a Managed Domain on a Single Machine
- Set Up a Managed Domain on Two Machines

Start a Host Controller
Start the host controller with the **`host-slave.xml`** configuration file, which is preconfigured for a slave host controller.

```
$ EAP_HOME/bin/domain.sh  --host-config=host-slave.xml
```

Depending on your domain setup, you will need to make additional configurations connect to, and not conflict with, the domain controller. Also see the following example domain setups:

- Set Up a Managed Domain on a Single Machine
- Set Up a Managed Domain on Two Machines

8.3.2. Configure the Domain Controller

IMPORTANT

It is not supported to configure multiple domain or host controllers on the same machine when using the RPM installation method to install JBoss EAP.

Configure a Host to Act as the Domain Controller
A host is designated to be the domain controller if it includes the **`<local/>`** element in the **`<domain-controller>`** declaration.

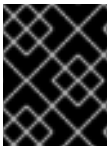
```
<domain-controller>
  <local/>
</domain-controller>
```

The host acting as the domain controller must expose a management interface that is accessible to other hosts in the domain. It is not required to expose an HTTP(S) management interface, but is recommended as it allows access to the management console.

```
<management-interfaces>
  <native-interface security-realm="ManagementRealm">
    <socket interface="management"
port="${jboss.management.native.port:9999}"/>
  </native-interface>
  <http-interface security-realm="ManagementRealm" http-upgrade-
enabled="true">
    <socket interface="management"
port="${jboss.management.http.port:9990}"/>
  </http-interface>
</management-interfaces>
```

The **`EAP_HOME/domain/configuration/host-master.xml`** file is already preconfigured with these settings to act as the domain controller.

8.3.3. Configure Host Controllers



IMPORTANT

It is not supported to configure multiple domain or host controllers on the same machine when using the RPM installation method to install JBoss EAP.

Connect to the Domain Controller
A host controller must be provided with the means to connect to the domain controller so that it can register itself with the domain. This is configured in the **<domain-controller>** element of the configuration.

```
<domain-controller>
  <remote security-realm="ManagementRealm">
    <discovery-options>
      <static-discovery name="primary"
protocol="${jboss.domain.master.protocol:remote}"
host="${jboss.domain.master.address}"
port="${jboss.domain.master.port:9999}"/>
    </discovery-options>
  </remote>
</domain-controller>
```

The **EAP_HOME/domain/configuration/host-slave.xml** file is already preconfigured with these settings to connect to the domain controller. You will need to provide the **jboss.domain.master.address** property when starting the host controller.

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

For more information on domain controller discovery, see the Domain Controller Discovery and Failover section.

Depending on your domain setup, you may also need to provide authentication for the host controller to be authenticated by the domain controller. See Set Up a Managed Domain on Two Machines for details on generating a management user with a secret value and updating the host controller configuration with that value.

8.3.3.1. Configure the Name of a Host

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.

A host controller's name is configured in the **host** element at the top of the relevant **host.xml** configuration file, for example:

```
<host xmlns="urn:jboss:domain:4.0" name="host1">
```

Use the following procedure to update the host name using the management CLI.

1. Start the JBoss EAP host controller.

   ```
   $ EAP_HOME/bin/domain.sh --host-config=host-slave.xml
   ```

2. Launch the management CLI, connecting to the domain controller.

   ```
   $ EAP_HOME/bin/jboss-cli.sh --connect --
   controller=DOMAIN_CONTROLLER_IP_ADDRESS
   ```

3. Use the following command to set a new host name.

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

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

   ```
   <host name="NEW_HOST_NAME" xmlns="urn:jboss:domain:4.0">
   ```

4. Reload the host controller in order for the changes to take effect.

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

If a host controller does not have a name set in the configuration file, you can also pass in the host name at runtime.

```
$ EAP_HOME/bin/domain.sh --host-config=host-slave.xml  -
Djboss.host.name=HOST_NAME
```

8.3.4. 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, 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 preconfigured 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: A Host Controller with Multiple Domain Controller Options

```
<domain-controller>
```

```
  <remote security-realm="ManagementRealm">
    <discovery-options>
      <static-discovery name="primary"
protocol="${jboss.domain.master.protocol:remote}" host="172.16.81.100"
port="${jboss.domain.master.port:9999}"/>
      <static-discovery name="backup"
protocol="${jboss.domain.master.protocol:remote}" host="172.16.81.101"
port="${jboss.domain.master.port:9999}"/>
    </discovery-options>
  </remote>
</domain-controller>
```

A static discovery option includes the following required 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.

Promote a Host Controller to Be the Domain Controller

1. Ensure the original domain controller 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
```

This host controller will now act as the domain controller.

8.4. MANAGING SERVERS

8.4.1. Configure Server Groups

The following is an example of a server group definition:

```
<server-group name="main-server-group" profile="full">
  <jvm name="default">
    <heap size="64m" max-size="512m"/>
  </jvm>
  <socket-binding-group ref="full-sockets"/>
  <deployments>
    <deployment name="test-application.war" runtime-name="test-
application.war"/>
    <deployment name="jboss-helloworld.war" runtime-name="jboss-
helloworld.war" enabled="false"/>
  </deployments>
</server-group>
```

Server groups can be configured using the management CLI or from the management console **Runtime** tab.

Add a Server Group
The following management CLI command can be used to add a server group.

```
/server-group=SERVER_GROUP_NAME:add(profile=PROFILE_NAME,socket-binding-
group=SOCKET_BINDING_GROUP_NAME)
```

Update a Server Group
The following management CLI command can be used to update server group attributes.

```
/server-group=SERVER_GROUP_NAME:write-
attribute(name=ATTRIBUTE_NAME,value=VALUE)
```

Remove a Server Group
The following management CLI command can be used to remove a server group.

```
/server-group=SERVER_GROUP_NAME:remove
```

Server Group Attributes
A server group requires the following 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 can be overridden on a per-server basis.

A server group includes the following optional attributes:

- **`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).
- **`socket-binding-default-interface`**: The socket binding group default interface for this server.

- **socket-binding-port-offset**: The default offset to be added to the port values given by the socket binding group.
- **deployments**: The deployment content to be deployed on the 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.
- **deployment-overlays**: Links between a defined deployment overlay and deployments in this server group.
- **system-properties**: The system properties to be set on servers in the group.

8.4.2. Configure Servers

The default **host.xml** configuration file defines three servers:

```
<servers>
  <server name="server-one" group="main-server-group">
  </server>
  <server name="server-two" group="main-server-group" auto-start="true">
    <socket-bindings port-offset="150"/>
  </server>
  <server name="server-three" group="other-server-group" auto-
start="false">
    <socket-bindings port-offset="250"/>
  </server>
</servers>
```

A server instance named **server-one** is associated with **main-server-group** and inherits the subsystem configuration and socket bindings specified by that server group. A server instance named **server-two** is also associated with **main-server-group**, but also defines a socket binding **port-offset** value, so as not to conflict with the port values used by **server-one**. A server instance named **server-three** is associated with **other-server-group** and uses that group's configurations. It also defines a **port-offset** value and sets **auto-start** to **false** so that this server does not start when the host controller starts.

Servers can be configured using the management CLI or from the management console **Runtime** tab.

Add a Server
The following management CLI command can be used to add a server.

```
/host=HOST_NAME/server-config=SERVER_NAME:add(group=SERVER_GROUP_NAME)
```

Update a Server
The following management CLI command can be used to update server attributes.

```
/host=HOST_NAME/server-config=SERVER_NAME:write-
attribute(name=ATTRIBUTE_NAME,value=VALUE)
```

Remove a Server
The following management CLI command can be used to remove a server.

```
/host=HOST_NAME/server-config=SERVER_NAME:remove
```

Server Attributes
A server requires the following attributes:

- **name**: The name of the server.
- **group**: The name of a server group from the domain model.

A server includes the following optional attributes:

- **auto-start**: Whether or not this server should be started when the host controller starts.
- **socket-binding-group**: The socket binding group to which this server belongs.
- **socket-binding-port-offset**: An offset to be added to the port values given by the socket binding group for this server.
- **update-auto-start-with-server-status**: Update the **auto-start** attribute with the status of the server.
- **interface**: A list of fully-specified named network interfaces available for use on the server.
- **jvm**: The JVM settings for this server. If not declared, the settings are inherited from the parent server group or host.
- **path**: A list of named file system paths.
- **system-property**: A list of system properties to set on this server.

8.4.3. Start and Stop Servers

You can perform operations on servers, such as starting, stopping, and reloading, from the management console by navigating to the **Runtime** tab and selecting the appropriate host or server group.

See the below commands for performing these operations using the management CLI.

Start Servers
You can start a single server on a particular host.

```
/host=HOST_NAME/server-config=SERVER_NAME:start
```

You can start all servers in a specified server group.

```
/server-group=SERVER_GROUP_NAME:start-servers
```

Stop Servers
You can stop a single server on a particular host.

```
/host=HOST_NAME/server-config=SERVER_NAME:stop
```

You can stop all servers in a specified server group.

```
/server-group=SERVER_GROUP_NAME:stop-servers
```

Reload Servers
You can reload a single server on a particular host.

```
/host=HOST_NAME/server-config=SERVER_NAME:reload
```

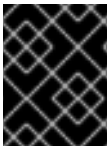
You can reload all servers in a specified server group.

```
/server-group=SERVER_GROUP_NAME:reload-servers
```

8.5. MANAGED DOMAIN SETUPS

8.5.1. Set Up a Managed Domain on a Single Machine

You can run multiple host controllers 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.

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

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

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

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

3. 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
```

4. 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
```

NOTE

When starting a host controller, you must specify the address of the domain controller using the **`jboss.domain.master.address`** property.

Additionally, since this host controller is running on the same machine as the domain controller, you must change the management interface so that it does not conflict with the domain controller's management interface. This command sets the **`jboss.management.native.port`** property.

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

8.5.2. Set Up a 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, where one machine is a domain controller and the other machine is a host. For more information, see About the Domain Controller.

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

Create a Managed Domain on Two Machines

1. On Machine 1
   a. Add a management user so that the host can be authenticated by the domain controller. Use the **`add-user.sh`** script to add the management user for the host controller (**`HOST_NAME`**). Make sure to answer **`yes`** to the last prompt and note the secret value provided (**`<secret value="SECRET_VALUE" />`**). This secret value will be used in the host controller configuration.

   b. Start the domain controller.
   Specify the **`host-master.xml`** configuration file, which is preconfigured for a dedicated domain controller. Also, set the **`jboss.bind.address.management`** property to make the domain controller visible to other machines.

```
$ EAP_HOME/bin/domain.sh --host-config=host-master.xml -
Djboss.bind.address.management=IP1
```

2. On Machine 2
   a. Update the host configuration with the user credentials.
   Edit **`EAP_HOME/domain/configuration/host-slave.xml`** and set the host name (**`HOST_NAME`**) and secret value (**`SECRET_VALUE`**).

```
<host xmlns="urn:jboss:domain:1.6" name="HOST_NAME">
  <management>
    <security-realms>
```

```
        <security-realm name="ManagementRealm">
          <server-identities>
            <secret value="SECRET_VALUE" />
          </server-identities>
          ...
```

b. Start the host controller.
Specify the **`host-slave.xml`** configuration file, which is preconfigured for a slave host controller. Also, set the **`jboss.domain.master.address`** property to connect to the domain controller and the **`jboss.bind.address`** property to set the host controller bind address.

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

You can now manage the domain from the management CLI by specifying the domain controller address with the **`--controller`** parameter when launching.

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

Or you can manage the domain from the management console at **`http://IP1:9990`**.

8.5.3. Configure a JBoss EAP 7 Domain Controller to Administer JBoss EAP 6 Instances

A JBoss EAP 7 domain controller can manage hosts and servers running older versions of JBoss EAP as long as the following conditions are met.

- The JBoss EAP instances must be JBoss EAP 6.2 or later.
- The JBoss EAP version of the domain controller must be higher than or equal to the version on the host controllers that it manages.

Complete the following tasks to successfully manage JBoss EAP 6 instances in a JBoss EAP 7 managed domain.

1. Add the JBoss EAP 6 configuration to the JBoss EAP 7 domain controller .
2. Update the behavior for the JBoss EAP 6 profiles .
3. Set the server group for the JBoss EAP 6 servers .
4. Prevent the JBoss EAP 6 instances from receiving JBoss EAP 7 updates .

Once these tasks are complete, you can manage your JBoss EAP 6 servers and configurations from the JBoss EAP 7 domain controller using the management CLI. Note that JBoss EAP 6 hosts will not be able to take advantage of new JBoss EAP 7 features, such as batch processing.

WARNING

Because the management console is optimized for the latest version of JBoss EAP, you should not use it to update your JBoss EAP 6 hosts, servers, and profiles. Use the management CLI instead when managing your JBoss EAP 6 configurations from a JBoss EAP 7 managed domain.

8.5.3.1. Add the JBoss EAP 6 Configuration to the JBoss EAP 7 Domain Controller

To allow the domain controller to manage your JBoss EAP 6 servers, you must provide the JBoss EAP 6 configuration details in the JBoss EAP 7 domain configuration. You can do this by copying the JBoss EAP 6 profiles, socket binding groups, and server groups to the JBoss EAP 7 **`domain.xml`** configuration file.

You will need to rename resources if any conflict with the existing names in the JBoss EAP 7 configuration. There are also some additional adjustments to make to ensure the proper behavior.

The following procedure uses the JBoss EAP 6 **`default`** profile, **`standard-sockets`** socket binding group, and **`main-server-group`** server group.

1. Edit the JBoss EAP 7 **`domain.xml`** configuration file.

2. Copy the applicable JBoss EAP 6 profiles to the JBoss EAP 7 **`domain.xml`** file.
   This procedure assumes that the JBoss EAP 6 **`default`** profile was copied and renamed to **`eap6-default`**.

   JBoss EAP 7 `domain.xml`

   ```
   <profiles>
     ...
     <profile name="eap6-default">
       ...
     </profile>
   </profiles>
   ```

3. Add the necessary extensions used by this profile.
   If your JBoss EAP 6 profile uses subsystems that are no longer present in JBoss EAP 7, you must add the appropriate extensions to the JBoss EAP domain configuration.

   JBoss EAP 7 `domain.xml`

   ```
   <extensions>
     ...
     <extension module="org.jboss.as.configadmin"/>
     <extension module="org.jboss.as.threads"/>
     <extension module="org.jboss.as.web"/>
   <extensions>
   ```

4. Copy the applicable JBoss EAP 6 socket binding groups to the JBoss EAP 7 **`domain.xml`** file.
   This procedure assumes that the JBoss EAP 6 **`standard-sockets`** socket binding group was

copied and renamed to **eap6-standard-sockets**.

JBoss EAP 7 domain.xml

```
<socket-binding-groups>
  ...
  <socket-binding-group name="eap6-standard-sockets" default-
interface="public">
    ...
  </socket-binding-group>
</socket-binding-groups>
```

5. Copy the applicable JBoss EAP 6 server groups to the JBoss EAP 7 **domain.xml** file.
This procedure assumes that the JBoss EAP 6 **main-server-group** server group was copied and renamed to **eap6-main-server-group**. You must also update this server group to use the JBoss EAP 6 profile, **eap6-default**, and the JBoss EAP 6 socket binding group, **eap6-standard-sockets**.

JBoss EAP 7 domain.xml

```
<server-groups>
  ...
  <server-group name="eap6-main-server-group" profile="eap6-
default">
    ...
    <socket-binding-group ref="eap6-standard-sockets"/>
  </server-group>
</server-groups>
```

8.5.3.2. Update the Behavior for the JBoss EAP 6 Profiles

Additional updates to the profiles used by your JBoss EAP 6 instances are necessary depending on the JBoss EAP version and desired behavior. You may require additional changes depending on the subsystems and configuration that your existing JBoss EAP 6 instances use.

Start the JBoss EAP 7 domain controller and launch its management CLI to perform the following updates. These examples assume that the JBoss EAP 6 profile is **eap6-default**.

- Remove the **bean-validation** subsystem.
JBoss EAP 7 moved bean validation functionality from the **ee** subsystem into its own subsystem, **bean-validation**. If a JBoss EAP 7 domain controller sees a legacy **ee** subsystem, it adds the new **bean-validation** subsystem. However, the JBoss EAP 6 hosts will not recognize this subsystem, so it must be removed.

  JBoss EAP 7 Domain Controller CLI

  ```
  /profile=eap6-default/subsystem=bean-validation:remove
  ```

- Set CDI 1.0 behavior.
This is only necessary if you want CDI 1.0 behavior for your JBoss EAP 6 servers, as opposed to behavior of later CDI versions used in JBoss EAP 7. If you want CDI 1.0 behavior, make the following updates to the **weld** subsystem.

JBoss EAP 7 Domain Controller CLI

```
/profile=eap6-default/subsystem=weld:write-attribute(name=require-
bean-descriptor,value=true)

/profile=eap6-default/subsystem=weld:write-attribute(name=non-
portable-mode,value=true)
```

- Enable datasource statistics for JBoss EAP 6.2.
  This is only necessary if your profile is being used by JBoss EAP 6.2 servers. JBoss EAP 6.3 introduced the **`statistics-enabled`** attribute, which defaults to **`false`** to not collect statistics; however, the JBoss EAP 6.2 behavior was to collect statistics. If this profile is used by a JBoss EAP 6.2 host and a host running a newer JBoss EAP version, the behavior would be inconsistent between hosts, which is not allowed. Therefore, profiles intended for use by a JBoss EAP 6.2 host should make the following change for their datasources.

  JBoss EAP 7 Domain Controller CLI

```
/profile=eap6-default/subsystem=datasources/data-
source=ExampleDS:write-attribute(name=statistics-enabled,value=true)
```

8.5.3.3. Set the Server Group for the JBoss EAP 6 Servers

If you renamed server groups, you will need to update the JBoss EAP 6 host configuration to use the new server groups specified in the JBoss EAP 7 configuration. This example uses the **`eap6-main-server-group`** server group specified in the JBoss EAP 7 **`domain.xml`**.

JBoss EAP 6 host-slave.xml

```
<servers>
    <server name="server-one" group="eap6-main-server-group"/>
    <server name="server-two" group="eap6-main-server-group">
        <socket-bindings port-offset="150"/>
    </server>
</servers>
```

NOTE

A host cannot use features or configuration settings that were introduced in a newer version of JBoss EAP than the one the host is running.

8.5.3.4. Prevent the JBoss EAP 6 Instances from Receiving JBoss EAP 7 Updates

The domain controller in a managed domain forwards configuration updates to its host controllers. You must use the **`host-exclude`** configuration to specify the resources that should be hidden from particular versions. Choose the appropriate preconfigured **`host-exclude`** option for your JBoss EAP 6 version: **`EAP62`**, **`EAP63`**, **`EAP64`**, or **`EAP64z`**.

The **`active-server-groups`** attribute of the **`host-exclude`** configuration specifies the list of server groups that are used by a particular version. These server groups and their associated profiles, socket binding groups, and deployment resources will be available to hosts of this version, but all others will be hidden from these hosts.

This example assumes that the version is JBoss EAP 6.4.z and adds the JBoss EAP 6 server group, **`eap6-main-server-group`** as an active server group.

JBoss EAP 7 Domain Controller CLI

```
/host-exclude=EAP64z:write-attribute(name=active-server-groups,value=
[eap6-main-server-group])
```

If necessary, you can specify additional socket binding groups used by your servers using the **`active-socket-binding-groups`** attribute. This is only required for socket binding groups that are not associated with the server groups specified in **`active-server-groups`**.

8.6. MANAGING JBOSS EAP PROFILES

8.6.1. About Profiles

JBoss EAP uses *profiles* as a way to organize which subsystems are available to a server. A profile consists of a collection of available subsystems along with each subsystem's specific configuration. A profile with a large number of subsystems results in a server with a large set of capabilities. A profile with a small, focused set of subsystems will have fewer capabilities but a smaller footprint.

JBoss EAP comes with four predefined profiles that should satisfy most use cases:

default

Includes commonly used subsystems, such as **`logging`**, **`security`**, **`datasources`**, **`infinispan`**, **`webservices`**, **`ee`**, **`ejb3`**, **`transactions`**, and so on.

ha

Includes the subsystems provided in the *default* profile with the addition of the **`jgroups`** and **`modcluster`** subsystems for high availability

full

Includes the subsystems provided in the *default* profile with the addition of the **`messaging-activemq`** and **`iiop-openjdk`** subsystems

full-ha

Includes the subsystems provided in the *full* profile with the addition of the **`jgroups`** and **`modcluster`** subsystems for high availability

NOTE

JBoss EAP offers the ability to disable extensions or unload drivers and other services manually by removing the subsystems from the configuration of existing profiles. However, for most cases this is unnecessary. Since JBoss EAP dynamically loads subsystems as they are needed, if the server or an application never use a subsystem, it will not be loaded.

In cases where the existing profiles do not provide the necessary capabilities, JBoss EAP also provides the ability to define custom profiles as well.

8.6.2. Cloning a Profile

JBoss EAP allows you to create a new profile in a managed domain by cloning an existing profile. This will create a copy of the original profile's configuration and subsystems.

A profile can be cloned using the management CLI by using the **clone** operation on the desired profile to clone.

```
/profile=full-ha:clone(to-profile=cloned-profile)
```

You can also clone a profile from the management console by selecting the desired profile to clone and clicking **Clone**.

8.6.3. Creating Hierarchical Profiles

In a managed domain, you can create a hierarchy of profiles. This allows you to create a base profile with common extensions that other profiles can inherit.

The managed domain defines several profiles in **domain.xml**. If multiple profiles use the same configuration for a particular subsystem, you can configure it in just one place instead of different profiles. The values in parent profiles cannot be overridden.

A profile can include other profiles in a hierarchy using the management CLI by using the **list-add** operation and providing the profile to include.

```
/profile=new-profile:list-add(name=includes, value=PROFILE_NAME)
```

CHAPTER 9. CONFIGURING JVM SETTINGS

Configuration of Java Virtual Machine (JVM) settings is different for a standalone JBoss EAP server, or a JBoss EAP server in a managed domain.

For a standalone JBoss EAP server instance, the server startup processes pass JVM settings to the JBoss EAP server at startup. These can be declared from the command line before launching JBoss EAP, or using the **System Properties** screen in the management console.

In a managed domain, JVM settings are declared in the **`host.xml`** and **`domain.xml`** configuration files, and can be configured at host, server group, or server levels.

NOTE

System properties must be configured in **`JAVA_OPTS`** to be used by JBoss EAP modules (such as the logging manager) during startup.

9.1. CONFIGURING JVM SETTINGS FOR A STANDALONE SERVER

JVM settings for standalone JBoss EAP 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 on Linux is shown below.

```
$ export JAVA_OPTS="-Xmx1024M"
```

The same setting can be used in a Microsoft Windows environment:

```
set JAVA_OPTS="Xmx1024M"
```

Alternatively, JVM settings can be added to the **`standalone.conf`** file in the **`EAP_HOME/bin`** folder, which contains examples of options to pass to the JVM.

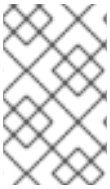
WARNING

Setting the **`JAVA_OPTS`** environment variable will override the default values from **`standalone.conf`**, which may cause JBoss EAP startup issues.

9.2. CONFIGURING JVM SETTINGS FOR A MANAGED DOMAIN

In a JBoss EAP managed domain, you can define JVM settings at multiple levels. You can define custom JVM settings on a particular host, and then apply those settings to server groups, or to individual server instances.

By default, server groups and individual servers will inherit the JVM settings from their parent, but you can choose to override JVM settings at each level.



NOTE

The JVM settings in **domain.conf** are applied to the Java process of the JBoss EAP host controller, and not the individual JBoss EAP server instances controlled by that host controller.

9.2.1. Defining JVM Settings on a Host Controller

You can define JVM settings on a host controller, and apply those settings to server groups or individual servers. JBoss EAP comes with a **default** JVM setting, but the following management CLI command demonstrates creating a new JVM setting named **production_jvm** with some custom JVM settings and options.

```
/host=HOST_NAME/jvm=production_jvm:add(
    heap-size=2048m,
    max-heap-size=2048m,
    max-permgen-size=512m,
    stack-size=1024k,
    jvm-options=["-XX:-UseParallelGC"]
)
```

You can also create and edit JVM settings in the JBoss EAP management console by selecting the **Runtime** tab, selecting **Hosts**, and clicking **JVM** on the host you want to edit.

These settings are stored in the within the **<jvm>** tag in **host.xml**.

9.2.2. Applying JVM Settings to a Server Group

When creating a server group, you can specify a JVM configuration that all servers in the group will use. The following management CLI commands demonstrate creating a server group name **groupA** that uses the **production_jvm** JVM settings that were shown in the previous example.

```
/server-group=groupA:add(profile=default, socket-binding-group=standard-
sockets)
/server-group=groupA/jvm=production_jvm:add()
```

All servers in the server group will inherit JVM settings from **production_jvm**.

You can also override specific JVM settings at the server group level. For example, to set a different heap size, you can use the following command:

```
/server-group=groupA/jvm=production_jvm:write-attribute(name=heap-
size,value="1024m")
```

After applying the above command, the server group **groupA** will inherit the JVM settings from **production_jvm**, except for the heap size which has an overridden value of **1024m**.

You can also edit server group JVM settings in the JBoss EAP management console by selecting the **Runtime** tab, selecting **Server Groups**, and clicking **View** for the server group you want to edit.

These settings for a server group are stored in **domain.xml**.

9.2.3. Applying JVM Settings to an Individual Server

By default, an individual JBoss EAP server instance will inherit the JVM settings of the server group it belongs to. However, you can choose to override the inherited settings with another complete JVM setting definition from the host controller, or choose to override specific JVM settings.

For example, the following command overrides the JVM definition of the server group in the previous example, and sets the JVM settings for **server-one** to the **default** JVM definition:

```
/host=HOST_NAME/server-config=server-one/jvm=default:add()
```

Also, similar to server groups, you can override specific JVM settings at the server level. For example, to set a different heap size, you can use the following command:

```
/host=HOST_NAME/server-config=server-one/jvm=default:write-
attribute(name=heap-size,value="1024m")
```

You can also edit server JVM settings in the JBoss EAP management console by selecting the **Runtime** tab, selecting **Hosts**, selecting the relevant host. Then select the relevant server, and click **View** for the server you want to edit.

These settings for an individual server are stored in **host.xml**.

9.3. DISPLAYING THE JVM STATUS

You can view the status of JVM resources, such as heap and thread usage, for standalone or managed domain servers from the management console. While statistics are not displayed in real time, you can click **Refresh Results** to provide an up-to-date overview of JVM resources.

To display the JVM status for a standalone JBoss EAP server:

- Select the **Runtime** tab, and then select **Standalone Server**. In the **Monitor** column, select **JVM** and click **View**.

To display the JVM status for a JBoss EAP server in a managed domain:

- Select the **Runtime** tab, and then select the server group and server that you want to view. In the **Monitor** column, select **JVM** and click **View**.

This shows the following heap usage information:

Max

The maximum amount of memory that can be used for memory management.

Used

The amount of used memory.

Committed

The amount of memory that is committed for the Java Virtual Machine to use.

Other information, such as JVM uptime and thread usage, is also available.

9.4. SPECIFYING 32 OR 64-BIT JVM 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. The default is 32-bit if neither option is indicated.

Specifying 64-Bit Architecture for a Standalone Server

1. Open **EAP_HOME/bin/standalone.conf**.
2. Add the following line to append the **-d64** option to **JAVA_OPTS**.

```
JAVA_OPTS="$JAVA_OPTS -d64"
```

Specifying 64-Bit Architecture for a Managed Domain

When running 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 **EAP_HOME/bin/domain.conf**.
   b. Add the following line to append the **-d64** option to **JAVA_OPTS**. Ensure that this is inserted before where **PROCESS_CONTROLLER_JAVA_OPTS** and **HOST_CONTROLLER_JAVA_OPTS** are set.

```
JAVA_OPTS="$JAVA_OPTS -d64"
```

Example 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.
   Add **-d64** as a JVM option in your appropriate JVM configuration. The command below shows it being added to the **default** JVM configuration.

```
/host=HOST_NAME/jvm=default:add-jvm-option(jvm-option="-d64")
```

CHAPTER 10. MAIL SUBSYSTEM

10.1. CONFIGURING THE MAIL SUBSYSTEM

The **mail** subsystem allows you to configure mail sessions in JBoss EAP and then inject those sessions into applications using JNDI. It also supports configuration using Java EE 7 **@MailSessionDefinition** and **@MailSessionDefinitions** annotations.

Configuring SMTP server for use in an application

1. Configure the SMTP server and the outbound socket binding using the following CLI commands, for example:

```
/socket-binding-group=standard-sockets/remote-destination-outbound-socket-binding=my-smtp:add(host=localhost, port=25)
```

```
/subsystem=mail/mail-session=mySession:add(jndi-name=java:jboss/mail/MySession)
```

```
/subsystem=mail/mail-session=mySession/server=smtp:add(outbound-socket-binding-ref=my-smtp, username=user, password=pass, tls=true)
```

2. Call the configured mail session within an application

```
@Resource(lookup="java:jboss/mail/MySession")
private Session session;
```

For a full list of the attributes available for configuring mail sessions and servers, see Mail Subsystem Attributes.

10.2. CONFIGURING CUSTOM TRANSPORTS

When using a standard mail server, such as POP3 or IMAP, the mail server has a set of attributes that can be defined. Some of these attributes are mandatory. 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 a host address and port number.

Defining the **outbound-socket-binding-ref** may not be the most effective solution for users who have their host configuration using multiple hosts for load balancing purposes. Standard JavaMail does not support host configuration using multiple hosts for load balancing. Therefore, users who have this configuration using multiple hosts are required to implement custom mail transports. These custom mail transports do not require the **outbound-socket-binding-ref** and allow custom host property formats.

You can configure custom mail transports from the management CLI.

1. Add a new mail session and specify the JNDI name.

```
/subsystem=mail/mail-session=mySession:add(jndi-name=java:jboss/mail/MySession)
```

2. Add an outbound socket binding and specify the host and port.

```
/socket-binding-group=standard-sockets/remote-destination-outbound-
socket-binding=my-smtp-binding:add(host=localhost, port=25)
```

3. Add an SMTP server and specify the outbound socket binding, username, and password.

```
/subsystem=mail/mail-session=mySession/server=smtp:add(outbound-
socket-binding-ref=my-smtp-binding, username=user, password=pass,
tls=true)
```

NOTE

You can configure a POP3 or IMAP server using similar steps.

POP3 Server

```
/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)
```

IMAP Server

```
/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 custom mail server without an outbound socket binding. You can specify the host information in the properties definition of the custom mail server. For example:

```
/subsystem=mail/mail-
session=mySession/custom=myCustomServer:add(username=user,password=pass,
properties={"host" => "myhost", "my-property" =>"value"})
```

If you define a custom protocol, any property name that includes a period (`.`) is considered a fully-qualified name and is passed directly. Any other format, such as **`my-property`**, is translated in the following format: **`mail.server-name.my-property`**.

The following XML is an example mail configuration that includes custom servers.

```
<subsystem xmlns="urn:jboss:domain:mail:2.0">
   <mail-session name="default" jndi-name="java:jboss/mail/Default">
        <smtp-server outbound-socket-binding-ref="mail-smtp"/>
   </mail-session>
    <mail-session name="myMail" from="user.name@domain.org" jndi-
name="java:/Mail">
        <smtp-server password="password" username="user" tls="true"
outbound-socket-binding-ref="mail-smtp"/>
```

```
        <pop3-server outbound-socket-binding-ref="mail-pop3"/>
        <imap-server password="password" username="nobody" outbound-
socket-binding-ref="mail-imap"/>
    </mail-session>
    <mail-session name="custom" jndi-name="java:jboss/mail/Custom"
debug="true">
        <custom-server name="smtp" password="password"
username="username">
            <property name="host" value="mail.example.com"/>
        </custom-server>
    </mail-session>
    <mail-session name="custom2" jndi-name="java:jboss/mail/Custom2"
debug="true">
        <custom-server name="pop3" outbound-socket-binding-ref="mail-
pop3">
            <property name="custom-prop" value="some-custom-prop-value"/>
        </custom-server>
    </mail-session>
</subsystem>
```

CHAPTER 11. CONFIGURING WEB SERVICES

JBoss EAP offers the ability to configure the behavior of deployed web services through the `webservices` subsystem using the management console or the management CLI. You can configure published endpoint addresses and handler chains. You can also enable the collection of runtime statistics for web services.

For more information, see Configuring the Web Services Subsystem in *Developing Web Services Applications* for JBoss EAP.

CHAPTER 12. LOGGING WITH JBOSS EAP

JBoss EAP 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 supports several third-party application logging frameworks in addition to JBoss Logging.

12.1. ABOUT SERVER LOGGING

12.1.1. Server Logging

By default, all JBoss EAP log entries are written to the **server.log** file. The location of this file depends on your operating mode.

- Standalone server: **EAP_HOME/standalone/log/server.log**
- Managed domain: **EAP_HOME/domain/servers/SERVER_NAME/log/server.log**

This file is often referred to as the server log. For more information, see the Root Logger section.

12.1.2. Bootup Logging

During bootup, JBoss EAP logs information about the Java environment and the startup of each service. This log can be useful when troubleshooting. By default, all log entries are written to the server log.

Bootup logging configuration is specified in the **logging.properties** configuration file, which is active until the JBoss EAP **logging** subsystem is started and takes over. The location of this file depends on your operating mode.

- Standalone server: **EAP_HOME/standalone/configuration/logging.properties**
- Managed domain:
  There is a **logging.properties** file for the domain controller and for each server.
  - Domain controller: **EAP_HOME/domain/configuration/logging.properties**
  - Server: **EAP_HOME/domain/servers/SERVER_NAME/data/logging.properties**

WARNING

It is recommended that you do not directly edit the **logging.properties** file unless you know of a specific use case that requires it. Before doing so, it is recommended that you open a support case from the Red Hat Customer Portal .

Changes made manually to the **logging.properties** file are overwritten on startup.

12.1.2.1. View Bootup Errors

When troubleshooting JBoss EAP, checking for errors that occurred during bootup should be one of the first steps taken. You can then use the information provided to diagnose and resolve their causes. Open a support case for assistance in troubleshooting bootup errors.

There are two methods of viewing bootup errors, each with its advantages. You can examine the `server.log` file or read the boot errors using the `read-boot-errors` management CLI command.

Examine the Server Log File
You can open the `server.log` file to view any errors that occurred during bootup.

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.

1. Open the file `server.log` in a file viewer.
2. Navigate to the end of the file.
3. Search backward for the **WFLYSRV0049** message identifier, 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.

The following is an example error description from the `server.log` log file.

```
2016-03-16 14:32:01,627 ERROR [org.jboss.msc.service.fail] (MSC service
thread 1-7) MSC000001: Failed to start service
jboss.undertow.listener.default: org.jboss.msc.service.StartException in
service jboss.undertow.listener.default: Could not start http listener
        at
org.wildfly.extension.undertow.ListenerService.start(ListenerService.java:
142)
        at
org.jboss.msc.service.ServiceControllerImpl$StartTask.startService(Service
ControllerImpl.java:1948)
        at
org.jboss.msc.service.ServiceControllerImpl$StartTask.run(ServiceControlle
rImpl.java:1881)
        at
java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:
1142)
        at
java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java
:617)
        at java.lang.Thread.run(Thread.java:745)
Caused by: java.net.BindException: Address already in use
        ...
```

Read the Boot Errors from the Management CLI
You can use the `read-boot-errors` management CLI command to view errors if a server starts but reported errors during bootup.

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 that starts multiple JBoss EAP

instances, then checks for errors that occurred on bootup.

Run the following management CLI command.

```
/core-service=management:read-boot-errors
```

Any errors that occurred during bootup will be listed.

```
{
    "outcome" => "success",
    "result" => [
        {
            "failed-operation" => {
                "operation" => "add",
                "address" => [
                    ("subsystem" => "undertow"),
                    ("server" => "default-server"),
                    ("http-listener" => "default")
                ]
            },
            "failure-description" => "{\"WFLYCTL0080: Failed services\" =>
{\"jboss.undertow.listener.default\" =>
\"org.jboss.msc.service.StartException in service
jboss.undertow.listener.default: Could not start http listener
    Caused by: java.net.BindException: Address already in use\"}}",
            "failed-services" => {"jboss.undertow.listener.default" =>
"org.jboss.msc.service.StartException in service
jboss.undertow.listener.default: Could not start http listener
    Caused by: java.net.BindException: Address already in use"}
        }
        ...
    ]
}
```

12.1.3. 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. Garbage collection logging is enabled by default for a JBoss EAP standalone server on all supported configurations *except* IBM Java development kit.

The location of the garbage collection log is **`EAP_HOME/standalone/log/gc.log.DIGIT.current`**. Garbage collection logs are limited to 3 MB each, and up to five files are rotated.

12.1.4. Default Log File Locations

The following log files are 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
EAP_HOME/standalone/log/server.log	Contains server log messages, including server startup messages.
EAP_HOME/standalone/log/gc.log.DIGIT.current	Contains garbage collection details.

Table 12.2. Default Log Files for a Managed Domain

Log File	Description
EAP_HOME/domain/log/host-controller.log	Contains log messages related to the startup of the host controller.
EAP_HOME/domain/log/process-controller.log	Contains log messages related to the startup of the process controller.
EAP_HOME/domain/servers/SERVER_NAME/log/server.log	Contains log messages for the named server, including server startup messages.

12.1.5. Set the Default Locale of the Server

You can configure the default locale for JBoss EAP by setting JVM properties in the appropriate startup configuration file. The startup configuration file is **`EAP_HOME/bin/standalone.conf`** for a standalone server or **`EAP_HOME/bin/domain.conf`** for a managed domain.

NOTE

For Windows Server, the JBoss EAP startup configuration files are **`standalone.conf.bat`** and **`domain.conf.bat`**.

Log messages that have been internationalized and localized will use this default locale. See the JBoss EAP *Development Guide* for information on creating internationalized log messages.

Set the Language
Specify the language by setting the **`user.language`** property using the **`JAVA_OPTS`** variable. For example, add the following line to the startup configuration file to set a French locale.

```
JAVA_OPTS="$JAVA_OPTS -Duser.language=fr"
```

Log messages that have been internationalized and localized will now output in French.

Set the Language and Country
In addition to the language, it may also be necessary to specify the country by setting the **`user.country`** property. For example, add the following line to the startup configuration file to set the Portuguese locale for Brazil.

```
JAVA_OPTS="$JAVA_OPTS -Duser.language=pt -Duser.country=BR"
```

Log messages that have been internationalized and localized will now output in Brazilian Portuguese.

Set the Server Locale Using the org.jboss.logging.locale Property
You can configure the `org.jboss.logging.locale` property to override the locale for messages logged using JBoss Logging, including any messages from JBoss EAP and its owned dependencies. Other dependencies, such as JSF, cannot get an overridden locale.

To start the JBoss EAP server with a different locale than the system default, you can edit *`EAP_HOME`*`/bin/standalone.conf` or the *`EAP_HOME`*`/bin/domain.conf` file, depending on your operating mode, and append the following command to set the JVM parameter for the required locale. The value of the property must be specified in the BCP 47 format. For example, to set Brazilian Portuguese, use pt-BR.

```
JAVA_OPTS="$JAVA_OPTS -Dorg.jboss.logging.locale=pt-BR"
```

12.2. VIEWING LOG FILES

Viewing server and application logs is important in order to help diagnose errors, performance problems, and other issues. Some users may prefer to view logs directly on the server file system. For those who do not have direct access to the file system, or who prefer a graphical interface, JBoss EAP allows you to view logs from the management console. You can also view logs using the management CLI.

For a log to be accessible from one of the management interfaces, it must be located in the directory specified by the server's `jboss.server.log.dir` property and be defined as a file, periodic rotating, size rotating, or periodic size rotating log handler. RBAC role assignments are also honored, so a user logged in to the management console or CLI can only view logs that they are authorized to access.

View Logs from the Management Console
You can view logs directly from the management console.

- Select the **Runtime** tab.
- Select **Standalone Server**. If you are running in a managed domain, select the appropriate server.
- Select **Log Files** and click **View**.

Once you choose a log file from the list, you can view and search the log contents directly in the management console. You can also download a log file to your local file system.

WARNING

The management console log viewer is not intended to be a text editor replacement for viewing very large log files, for example, greater than 100MB. You will be prompted for confirmation if you attempt to open a log file that is larger than 15MB. Opening a very large file in the management console could crash your browser, so you should always download large log files locally and open them in a text editor.

View Logs from the Management CLI

You can read the contents of log files from the management CLI using the **read-log-file** command. By default, this displays the last **10** lines of the specified log file.

```
/subsystem=logging/log-file=LOG_FILE_NAME:read-log-file
```

NOTE

In a managed domain, precede this command with **/host=HOST_NAME/server=SERVER_NAME**.

You can use the following parameters to customize the log output.

encoding

The character encoding used to read the file.

lines

The number of lines to read from the file. A value of **-1** will read all log lines. The default is **10**.

skip

The number of lines to skip before reading. The default is **0**.

tail

Whether to read from the end of the file. Defaults to **true**.

For example, the following management CLI command reads the first **5** lines from the top of the **server.log** log file.

```
/subsystem=logging/log-file=server.log:read-log-file(lines=5,tail=false)
```

This produces the following output.

```
{
    "outcome" => "success",
    "result" => [
        "2016-03-24 08:49:26,612 INFO  [org.jboss.modules] (main) JBoss
Modules version 1.5.1.Final-redhat-1",
        "2016-03-24 08:49:26,788 INFO  [org.jboss.msc] (main) JBoss MSC
version 1.2.6.Final-redhat-1",
        "2016-03-24 08:49:26,863 INFO  [org.jboss.as] (MSC service thread
1-7) WFLYSRV0049: JBoss EAP 7.0.0.GA (WildFly Core 2.0.13.Final-redhat-1)
starting",
        "2016-03-24 08:49:27,973 INFO  [org.jboss.as.server] (Controller
Boot Thread) WFLYSRV0039: Creating http management service using socket-
binding (management-http)",
        "2016-03-24 08:49:27,994 INFO  [org.xnio] (MSC service thread 1-1)
XNIO version 3.3.4.Final-redhat-1"
    ]
}
```

12.3. ABOUT THE LOGGING SUBSYSTEM

The JBoss EAP **logging** subsystem is configured using a system of log categories and log handlers. Log categories define what messages to capture. Log handlers define how to deal with those messages, for example, writing to a disk or sending to the console.

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.

12.3.1. Root Logger

The JBoss EAP root logger captures all log messages, of the specified log level or higher, sent to the server that are not captured by a log category.

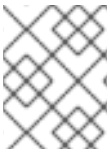
By default, the root logger is configured to use a console and a periodic log handler. The periodic log handler is configured to write to the **server.log** file. This file is often referred to as the server log.

See Configuring the Root Logger for more information.

12.3.2. Log Categories

A log category defines a set of log messages to capture and one or more log handlers that will process the messages.

The log messages to capture are defined by the specified Java package of origin and log level. Messages from classes in that package and of that log level or higher are captured by the log category and sent to the specified log handlers.



NOTE

Although the log category is typically the Java package and class name, it can be any name specified by the **Logger.getLogger(LOGGER_NAME)** method.

Log categories can optionally use the log handlers of the root logger instead of their own handlers.

See Configuring Log Categories for more information.

12.3.3. Log Handlers

Log handlers define how captured log messages are recorded. The available log handler types are *console*, *file*, *periodic*, *size*, *periodic size*, *syslog*, *custom*, and *async*.

NOTE

A log handler must be added to at least one logger in order to be active.

Log Handler Types

Console

A console log handler writes log messages to either the host operating system's standard out (**stdout**) or standard error (**stderr**) stream. These messages are displayed when JBoss EAP 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

A file log handler writes log messages to a specified file.

Periodic

A periodic log handler writes log messages to a named file until a specified period of time has elapsed. Once the time period has passed, 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

A size log handler writes 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

A periodic size log handler writes log messages to a named file until the file reaches the specified size *or* the specified time period has passed. Then, the file is renamed and the handler continues to write to a newly created log file with the original name.
This is a combination of the periodic and size log handlers and supports their combined attributes.

Syslog

A syslog handler 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.

Custom

A custom log handler enables 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. You can also use a Log4J appender as a custom log handler.

Async

An async log handler is a wrapper log handler that provides asynchronous behavior for one or more other log handlers. This is useful for log handlers that may have high latency or other performance problems such as writing a log file to a network file system.

For details on configuring each of these log handlers, see the Configuring Log Handlers section.

12.3.4. Log Levels

A log level is an enumerated value that indicates the nature and severity of a log message. As a developer, you can specify the level of a given log message using the appropriate method of your chosen logging framework to send the message.

JBoss EAP supports all the log levels used by the supported application logging frameworks. The most commonly used log levels from lowest to highest are **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 that 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**.

Supported Log Levels

Log Level	Value	Description

Log Level	Value	Description
ALL	Integer.MIN_VALUE	Provides all log messages.
FINEST	300	-
FINER	400	-
TRACE	400	**TRACE** level log messages provide detailed information about the running state of an application and are usually only captured during debugging.
DEBUG	500	**DEBUG** level log messages indicate the progress of individual requests or application activities and are usually only captured during debugging.
FINE	500	-
CONFIG	700	-
INFO	800	**INFO** level log messages indicate the overall progress of the application. Often used for application startup, shutdown, and other major lifecycle events.
WARN	900	**WARN** level log messages indicate a situation that is not in error, but is not considered ideal. **WARN** log messages can indicate circumstances that could lead to errors in the future.
WARNING	900	-
ERROR	1000	**ERROR** level log messages 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	**FATAL** level log messages indicate events that could cause critical service failure and application shutdown and could cause JBoss EAP to shutdown.
OFF	Integer.MAX_VALUE	Does not display any log message.

NOTE

ALL is the lowest log level and includes messages of all log levels. This provides the most amount of logging.

FATAL is the highest log level and only includes messages of that level. This provides the least amount of logging.

12.3.5. Log Formatters

A log formatter defines the appearance of log messages from that handler. It is a string that uses a syntax based on **`java.util.logging.Formatter`** class.

For example, the default configuration uses the following log formatter string for logging messages to the server log: **`%d{yyyy-MM-dd HH:mm:ss,SSS} %-5p [%c] (%t) %s%e%n`**. This will create log messages like the one shown below.

```
2016-03-18 15:49:32,075 INFO  [org.jboss.as] (Controller Boot Thread)
WFLYSRV0051: Admin console listening on http://127.0.0.1:9990
```

For more information on configuring log formatters, see Configure a Named Pattern Formatter or Configure a Custom Log Formatter .

See the table below for the syntax used in log formatter strings.

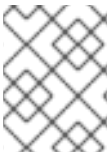
Log Formatter Syntax

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`** format).
%r	The relative time (milliseconds since the log was initialized).
%z	The time zone, which must be specified before the date (**`%d`**). For example, **`%z{GMT}%d{HH:mm:ss,SSS}`**.
%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).
%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).

Symbol	Description
%F	The filename of the class calling the log method (slow).
%l	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).

12.3.6. Filter Expressions

Filter expressions, configured using the `filter-spec` attribute, are used to record log messages based on various criteria. Filter checking is always done on a raw unformatted message. You can include a filter for a logger or handler, but the logger filter takes precedence over the filter put on a handler.



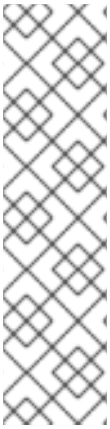
NOTE

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 Expression	Description
accept	Accept all log messages.
deny	Deny all log messages.
not[filter expression]	Returns the inverted value of a single filter expression. For example: `not(match("WFLY"))`
all[filter expression]	Returns concatenated value from a comma-separated list of filter expressions. For example: `all(match("WFLY"),match("WELD"))`

Filter Expression	Description
any[filter expression]	Returns one value from a comma-separated list of filter expressions. For example: **`any(match("WFLY"),match("WELD"))`**
levelChange[level]	Updates the log record with the specified level. For example: **`levelChange(WARN)`**
levels[levels]	Filters log messages with a level listed in the comma-separated list of levels. For example: **`levels(DEBUG,INFO,WARN,ERROR)`**
levelRange[minLevel,maxLevel]	Filters log messages within the specified level range. The **`[`** and **`]`** characters are used to indicate an inclusive level. The **`(`** and **`)`** characters are used to indicate an exclusive level. For example: • **`levelRange[INFO,ERROR]`** ◦ The minimum level must be greater than or equal to **INFO** and the maximum level must be less than or equal to **ERROR**. • **`levelRange[DEBUG,ERROR)`** ◦ The minimum level must be greater than or equal to **DEBUG** and the maximum level must be less than **ERROR**.
match["pattern"]	Filters log messages using the provided regular expression. For example: **`match("WFLY\d+")`**
substitute["pattern","replacement value"]	A filter that replaces the first match to the pattern (first argument) with the replacement text (second argument). For example: **`substitute("WFLY","EAP")`**
substituteAll["pattern","replacement value"]	A filter which replaces all matches of the pattern (first argument) with the replacement text (second argument). For example: **`substituteAll("WFLY","EAP")`**



NOTE

When configuring the filter expression using the management CLI, be sure to escape commas and quotation marks in the filter text so that the value is correctly processed as a string. You must precede commas and quotation marks with a backslash (\) and wrap the entire expression in quotation marks. Below is an example that properly escapes **`substituteAll("WFLY","YLFW")`**.

```
/subsystem=logging/console-handler=CONSOLE:write-
attribute(name=filter-spec,
value="substituteAll(\"WFLY\"\,\"YLFW\")")
```

12.3.7. Implicit Logging Dependencies

By default, the JBoss EAP **`logging`** subsystem adds implicit logging API dependencies to deployments. You can control whether these implicit dependencies are added to deployments by using the **`add-logging-api-dependencies`** attribute, which is **`true`** by default.

Using the management CLI, you can set the **`add-logging-api-dependencies`** attribute to **`false`** so that the implicit logging API dependencies will not be added to deployments.

```
/subsystem=logging:write-attribute(name=add-logging-api-dependencies,
value=false)
```

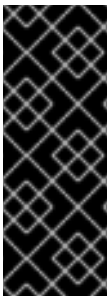
For information on the implicit dependencies for the **`logging`** subsystem, see the Implicit Module Dependencies section in the JBoss EAP *Development Guide*.

12.4. CONFIGURING LOG CATEGORIES

This section shows you how to configure log categories using the management CLI. You can also configure log categories using the management console by navigating to the **Logging** subsystem from the **Configuration** tab and selecting the **Log Categories** tab.

The main tasks you will perform to configure a log category are:

- Add a new log category.
- Configure the log category settings.
- Assign a log handler to the log category.



IMPORTANT

If you are configuring this log category for a logging profile, the start of the command would be **`/subsystem=logging/logging-profile=LOGGING_PROFILE_NAME/`** instead of **`/subsystem=logging/`**.

Additionally, if you are running in a managed domain, precede the commands with **`/profile=PROFILE_NAME`**.

Add a Log Category

The log category name is defined by the Java package of origin. Messages from classes in that package will be captured as long as they adhere to the other settings, for example, the log level.

```
/subsystem=logging/logger=LOG_CATEGORY:add
```

Configure Log Category Settings
Depending on your needs, you may need to set one or more of the following log category attributes. For a full list of available log category attributes and their descriptions, see Log Category Attributes.

- Set the log level.
  Set the appropriate log level for the log category. The default is **ALL**. See Log Levels for all available options.

  ```
  /subsystem=logging/logger=LOG_CATEGORY:write-
  attribute(name=level,value=LEVEL)
  ```

- Set whether this category should use the log handlers of the root logger.
  By default, log categories will use the handlers of the root logger in addition to its own. Set the **use-parent-handlers** attribute to **false** if the log category should use only its assigned handlers.

  ```
  /subsystem=logging/logger=LOG_CATEGORY:write-attribute(name=use-
  parent-handlers,value=USE_PARENT_HANDLERS)
  ```

- Set the filter expression.
  Set the expression for filtering log messages for the log category. Be sure to escape any commas and quotation marks and surround with quotation marks. For example, the **FILTER_EXPRESSION** replaceable variable below would need to be replaced with **"not(match(\"WFLY\"))"** for a filter expression of **not(match("WFLY"))**.

  ```
  /subsystem=logging/logger=LOG_CATEGORY:write-attribute(name=filter-
  spec, value=FILTER_EXPRESSION)
  ```

  See the Filter Expressions section for more information on available filter expressions.

Assign a Handler
Assign a log handler to the log category.

```
/subsystem=logging/logger=LOG_CATEGORY:add-handler(name=LOG_HANDLER_NAME)
```

Remove a Log Category
A log category can be removed with the **remove** operation.

```
/subsystem=logging/logger=LOG_CATEGORY:remove
```

12.5. CONFIGURING LOG HANDLERS

Log handlers define how captured log messages are recorded. See the appropriate section for configuring the type of log handler that you need.

- Console log handler
- File log handler
- Periodic rotating log handler

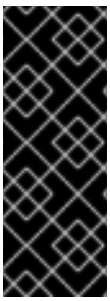
- Size rotating log handler
- Periodic size rotating log handler
- Syslog handler
- Custom log handler
- Async log handler

12.5.1. Configure a Console Log Handler

This section shows you how to configure a console log handler using the management CLI. You can also configure console log handlers using the management console by navigating to the **Logging** subsystem from the **Configuration** tab, selecting the **Handler** tab, and selecting **Console** from the left-hand menu.

The main tasks you will perform to configure a console log handler are:

- Add a new console log handler.
- Configure the console log handler settings.
- Assign the console log handler to a logger.



IMPORTANT

If you are configuring this log handler for a logging profile, the start of the command would be **`/subsystem=logging/logging-profile=LOGGING_PROFILE_NAME/`** instead of **`/subsystem=logging/`**.

Additionally, if you are running in a managed domain, precede the commands with **`/profile=PROFILE_NAME`**.

Add a Console Log Handler

```
/subsystem=logging/console-handler=CONSOLE_HANDLER_NAME:add
```

Configure Console Log Handler Settings
Depending on your needs, you may need to set one or more of the following console log handler attributes. For a full list of available console log handler attributes and their descriptions, see Console Log Handler Attributes.

- Set the log level.
  Set the appropriate log level for the handler. The default is **ALL**. See Log Levels for all available options.

  ```
  /subsystem=logging/console-handler=CONSOLE_HANDLER_NAME:write-
  attribute(name=level,value=LEVEL)
  ```

- Set the target.
  Set the target for the handler, which can be one of **`System.out`**, **`System.err`**, or **`console`**. The default is **`System.out`**.

```
/subsystem=logging/console-handler=CONSOLE_HANDLER_NAME:write-
attribute(name=target,value=TARGET)
```

- Set the encoding.
  Set the encoding for the handler, for example, **`utf-8`**.

```
/subsystem=logging/console-handler=CONSOLE_HANDLER_NAME:write-
attribute(name=encoding,value=ENCODING)
```

- Set the log formatter.
  Set the formatter string for the handler. For example, the default format string is **`%d{HH:mm:ss,SSS} %-5p [%c] (%t) %s%e%n`**. Be sure to enclose the **`FORMAT`** value in quotation marks.

```
/subsystem=logging/console-handler=CONSOLE_HANDLER_NAME:write-
attribute(name=formatter,value=FORMAT)
```

NOTE

Use the **`named-formatter`** attribute if you want to reference a saved formatter.

- Set auto flush.
  Set whether to automatically flush after each write. The default value is **`true`**.

```
/subsystem=logging/console-handler=CONSOLE_HANDLER_NAME:write-
attribute(name=autoflush,value=AUTO_FLUSH)
```

- Set the filter expression.
  Set the expression for filtering log messages for the handler. Be sure to escape any commas and quotation marks and surround with quotation marks. For example, the **`FILTER_EXPRESSION`** replaceable variable below would need to be replaced with **`"not(match(\"WFLY\"))"`** for a filter expression of **`not(match("WFLY"))`**.

```
/subsystem=logging/console-handler=CONSOLE_HANDLER_NAME:write-
attribute(name=filter-spec, value=FILTER_EXPRESSION)
```

  See the Filter Expressions section for more information on available filter expressions.

Assign the Console Log Handler to a Logger
In order for a log handler to be active, you must assign it to a logger.

The following management CLI command assigns the console log handler to the root logger.

```
/subsystem=logging/root-logger=ROOT:add-handler(name=CONSOLE_HANDLER_NAME)
```

The following management CLI command assigns the console log handler to a logger whose name is specified by **`CATEGORY`**.

```
/subsystem=logging/logger=CATEGORY:add-handler(name=CONSOLE_HANDLER_NAME)
```

Remove a Console Log Handler
A log handler can be removed with the **remove** operation. A log handler cannot be removed if it is currently assigned to a logger or async log handler.

```
/subsystem=logging/console-handler=CONSOLE_HANDLER_NAME:remove
```

12.5.2. Configure a File Log Handler

This section shows you how to configure a file log handler using the management CLI. You can also configure file log handlers using the management console by navigating to the **Logging** subsystem from the **Configuration** tab, selecting the **Handler** tab, and selecting **File** from the left-hand menu.

The main tasks you will perform to configure a file log handler are:

- Add a new file log handler.
- Configure the file log handler settings.
- Assign the file log handler to a logger.

IMPORTANT

If you are configuring this log handler for a logging profile, the start of the command would be **/subsystem=logging/logging-profile=LOGGING_PROFILE_NAME/** instead of **/subsystem=logging/**.

Additionally, if you are running in a managed domain, precede the commands with **/profile=PROFILE_NAME**.

Add a File Log Handler
When adding a file log handler, you must specify the file path using the **file** attribute, which is comprised of the **path** and **relative-to** attributes. Use the **path** attribute to set the file path for the log, including the name, for example **my-log.log**. Optionally, use the **relative-to** attribute to set that the **path** is relative to a named path, for example **jboss.server.log.dir**.

```
/subsystem=logging/file-handler=FILE_HANDLER_NAME:add(file=
{path=FILE_PATH,relative-to=RELATIVE_TO_PATH})
```

Configure File Log Handler Settings
Depending on your needs, you may need to set one or more of the following file log handler attributes. For a full list of available file log handler attributes and their descriptions, see File Log Handler Attributes.

- Set the log level.
  Set the appropriate log level for the handler. The default is **ALL**. See Log Levels for all available options.

  ```
  /subsystem=logging/file-handler=FILE_HANDLER_NAME:write-
  attribute(name=level,value=LEVEL)
  ```

- Set the append behavior.
  By default, JBoss EAP will append log messages to the same file when the server is restarted. You can set the **append** attribute to **false** to have the file overwritten upon server restart.

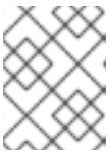
```
/subsystem=logging/file-handler=FILE_HANDLER_NAME:write-
attribute(name=append,value=APPEND)
```

- Set the encoding.
  Set the encoding for the handler, for example, **`utf-8`**.

```
/subsystem=logging/file-handler=FILE_HANDLER_NAME:write-
attribute(name=encoding,value=ENCODING)
```

- Set the log formatter.
  Set the formatter string for the handler. For example, the default format string is **`%d{HH:mm:ss,SSS} %-5p [%c] (%t) %s%e%n`**. Be sure to enclose the **`FORMAT`** value in quotation marks.

```
/subsystem=logging/file-handler=FILE_HANDLER_NAME:write-
attribute(name=formatter,value=FORMAT)
```



NOTE

Use the **`named-formatter`** attribute if you want to reference a saved formatter.

- Set auto flush.
  Set whether to automatically flush after each write. The default value is **`true`**.

```
/subsystem=logging/file-handler=FILE_HANDLER_NAME:write-
attribute(name=autoflush,value=AUTO_FLUSH)
```

- Set the filter expression.
  Set the expression for filtering log messages for the handler. Be sure to escape any commas and quotation marks and surround with quotation marks. For example, the **`FILTER_EXPRESSION`** replaceable variable below would need to be replaced with **`"not(match(\"WFLY\"))"`** for a filter expression of **`not(match("WFLY"))`**.

```
/subsystem=logging/file-handler=FILE_HANDLER_NAME:write-
attribute(name=filter-spec, value=FILTER_EXPRESSION)
```

  See the Filter Expressions section for more information on available filter expressions.

Assign the File Log Handler to a Logger
In order for a log handler to be active, you must assign it to a logger.

The following management CLI command assigns the file log handler to the root logger.

```
/subsystem=logging/root-logger=ROOT:add-handler(name=FILE_HANDLER_NAME)
```

The following management CLI command assigns the file log handler to a logger whose name is specified by **`CATEGORY`**.

```
/subsystem=logging/logger=CATEGORY:add-handler(name=FILE_HANDLER_NAME)
```

Remove a File Log Handler
A log handler can be removed with the **remove** operation. A log handler cannot be removed if it is currently assigned to a logger or async log handler.

```
/subsystem=logging/file-handler=FILE_HANDLER_NAME:remove
```

12.5.3. Configure a Periodic Rotating Log Handler

This section shows you how to configure a periodic rotating log handler using the management CLI. You can also configure periodic log handlers using the management console by navigating to the **Logging** subsystem from the **Configuration** tab, selecting the **Handler** tab, and selecting **Periodic** from the left-hand menu.

The main tasks you will perform to configure a periodic log handler are:

- Add a new periodic log handler .
- Configure the periodic log handler settings .
- Assign the periodic log handler to a logger .

IMPORTANT

If you are configuring this log handler for a logging profile, the start of the command would be **/subsystem=logging/logging-profile=LOGGING_PROFILE_NAME/** instead of **/subsystem=logging/**.

Additionally, if you are running in a managed domain, precede the commands with **/profile=PROFILE_NAME**.

Add a Periodic Log Handler
When adding a periodic log handler, you must specify the file path using the **file** attribute, which is comprised of the **path** and **relative-to** attributes. Use the **path** attribute to set the file path for the log, including the name, for example **my-log.log**. Optionally, use the **relative-to** attribute to set that the **path** is relative to a named path, for example **jboss.server.log.dir**.

You must also set the suffix for rotated logs using the **suffix** attribute. This must be in a format that can be understood by **java.text.SimpleDateFormat**, for example **.yyyy-MM-dd-HH**. The period of the rotation is automatically calculated based on this suffix.

```
/subsystem=logging/periodic-rotating-file-
handler=PERIODIC_HANDLER_NAME:add(file={path=FILE_PATH,relative-
to=RELATIVE_TO_PATH},suffix=SUFFIX)
```

Configure Periodic Log Handler Settings
Depending on your needs, you may need to set one or more of the following periodic log handler attributes. For a full list of available periodic log handler attributes and their descriptions, see Periodic Log Handler Attributes.

- Set the log level.
  Set the appropriate log level for the handler. The default is **ALL**. See Log Levels for all available options.

```
/subsystem=logging/periodic-rotating-file-
handler=PERIODIC_HANDLER_NAME:write-
attribute(name=level,value=LEVEL)
```

- Set the append behavior.
  By default, JBoss EAP will append log messages to the same file when the server is restarted. You can set the **append** attribute to **`false`** to have the file overwritten upon server restart.

```
/subsystem=logging/periodic-rotating-file-
handler=PERIODIC_HANDLER_NAME:write-
attribute(name=append,value=APPEND)
```

- Set the encoding.
  Set the encoding for the handler, for example, **`utf-8`**.

```
/subsystem=logging/periodic-rotating-file-
handler=PERIODIC_HANDLER_NAME:write-
attribute(name=encoding,value=ENCODING)
```

- Set the log formatter.
  Set the formatter string for the handler. For example, the default format string is **`%d{HH:mm:ss,SSS} %-5p [%c] (%t) %s%e%n`**. Be sure to enclose the **FORMAT** value in quotation marks.

```
/subsystem=logging/periodic-rotating-file-
handler=PERIODIC_HANDLER_NAME:write-
attribute(name=formatter,value=FORMAT)
```

NOTE

Use the **`named-formatter`** attribute if you want to reference a saved formatter.

- Set auto flush.
  Set whether to automatically flush after each write. The default value is **`true`**.

```
/subsystem=logging/periodic-rotating-file-
handler=PERIODIC_HANDLER_NAME:write-
attribute(name=autoflush,value=AUTO_FLUSH)
```

- Set the filter expression.
  Set the expression for filtering log messages for the handler. Be sure to escape any commas and quotation marks and surround with quotation marks. For example, the **FILTER_EXPRESSION** replaceable variable below would need to be replaced with **`"not(match(\"WFLY\"))"`** for a filter expression of **`not(match("WFLY"))`**.

```
/subsystem=logging/periodic-rotating-file-
handler=PERIODIC_HANDLER_NAME:write-attribute(name=filter-spec,
value=FILTER_EXPRESSION)
```

  See the Filter Expressions section for more information on available filter expressions.

Assign the Periodic Log Handler to a Logger
In order for a log handler to be active, you must assign it to a logger.

The following management CLI command assigns the periodic log handler to the root logger.

```
/subsystem=logging/root-logger=ROOT:add-
handler(name=PERIODIC_HANDLER_NAME)
```

The following management CLI command assigns the periodic log handler to a logger whose name is specified by **CATEGORY**.

```
/subsystem=logging/logger=CATEGORY:add-handler(name=PERIODIC_HANDLER_NAME)
```

Remove a Periodic Log Handler
A log handler can be removed with the **remove** operation. A log handler cannot be removed if it is currently assigned to a logger or async log handler.

```
/subsystem=logging/periodic-rotating-file-
handler=PERIODIC_HANDLER_NAME:remove
```

12.5.4. Configure a Size Rotating Log Handler

This section shows you how to configure a size rotating log handler using the management CLI. You can also configure size log handlers using the management console by navigating to the **Logging** subsystem from the **Configuration** tab, selecting the **Handler** tab, and selecting **Size** from the left-hand menu.

The main tasks you will perform to configure a size log handler are:

- Add a new size log handler .
- Configure the size log handler settings .
- Assign the size log handler to a logger .

IMPORTANT

If you are configuring this log handler for a logging profile, the start of the command would be **/subsystem=logging/logging-profile=LOGGING_PROFILE_NAME/** instead of **/subsystem=logging/**.

Additionally, if you are running in a managed domain, precede the commands with **/profile=PROFILE_NAME**.

Add a Size Log Handler
When adding a size log handler, you must specify the file path using the **file** attribute, which is comprised of the **path** and **relative-to** attributes. Use the **path** attribute to set the file path for the log, including the name, for example **my-log.log**. Optionally, use the **relative-to** attribute to set that the **path** is relative to a named path, for example **jboss.server.log.dir**.

```
/subsystem=logging/size-rotating-file-handler=SIZE_HANDLER_NAME:add(file=
{path=FILE_PATH,relative-to=RELATIVE_TO_PATH})
```

Configure Size Log Handler Settings
Depending on your needs, you may need to set one or more of the following size log handler attributes. For a full list of available size log handler attributes and their descriptions, see Size Log Handler Attributes.

- Set the log level.
  Set the appropriate log level for the handler. The default is **ALL**. See Log Levels for all available options.

```
/subsystem=logging/size-rotating-file-
handler=SIZE_HANDLER_NAME:write-attribute(name=level,value=LEVEL)
```

- Set the suffix for rotated logs.
  Set the suffix string, which is in a format which can be understood by **`java.text.SimpleDateFormat`**, for example **`.yyyy-MM-dd-HH`**. The period of the rotation is automatically calculated based on this suffix.

```
/subsystem=logging/size-rotating-file-
handler=SIZE_HANDLER_NAME:write-attribute(name=suffix, value=SUFFIX)
```

- Set the rotation size.
  Set the maximum size that the file can reach before being rotated. The default is **`2m`** for 2 megabytes.

```
/subsystem=logging/size-rotating-file-
handler=SIZE_HANDLER_NAME:write-attribute(name=rotate-size,
value=ROTATE_SIZE)
```

- Set the maximum number of backup logs to keep.
  Set the number of backups to keep. The default is **`1`**.

```
/subsystem=logging/size-rotating-file-
handler=SIZE_HANDLER_NAME:write-attribute(name=max-backup-index,
value=MAX_BACKUPS)
```

- Set whether to rotate the log on boot.
  By default, a new log file is not created on server restart. You can set this to **`true`** to rotate the log on server restart.

```
/subsystem=logging/size-rotating-file-
handler=SIZE_HANDLER_NAME:write-attribute(name=rotate-on-boot,
value=ROTATE_ON_BOOT)
```

- Set the append behavior.
  By default, JBoss EAP will append log messages to the same file when the server is restarted. You can set the **append** attribute to **`false`** to have the file overwritten upon server restart.

```
/subsystem=logging/size-rotating-file-
handler=SIZE_HANDLER_NAME:write-attribute(name=append,value=APPEND)
```

- Set the encoding.
  Set the encoding for the handler, for example, **`utf-8`**.

```
/subsystem=logging/size-rotating-file-
handler=SIZE_HANDLER_NAME:write-
attribute(name=encoding,value=ENCODING)
```

- Set the log formatter.
  Set the formatter string for the handler. For example, the default format string is **`%d{HH:mm:ss,SSS} %-5p [%c] (%t) %s%e%n`**. Be sure to enclose the **`FORMAT`** value in quotation marks.

```
/subsystem=logging/size-rotating-file-
handler=SIZE_HANDLER_NAME:write-
attribute(name=formatter,value=FORMAT)
```

NOTE

Use the **`named-formatter`** attribute if you want to reference a saved formatter.

- Set auto flush.
  Set whether to automatically flush after each write. The default value is **`true`**.

```
/subsystem=logging/size-rotating-file-
handler=SIZE_HANDLER_NAME:write-
attribute(name=autoflush,value=AUTO_FLUSH)
```

- Set the filter expression.
  Set the expression for filtering log messages for the handler. Be sure to escape any commas and quotation marks and surround with quotation marks. For example, the **`FILTER_EXPRESSION`** replaceable variable below would need to be replaced with **`"not(match(\"WFLY\"))"`** for a filter expression of **`not(match("WFLY"))`**.

```
/subsystem=logging/size-rotating-file-
handler=SIZE_HANDLER_NAME:write-attribute(name=filter-spec,
value=FILTER_EXPRESSION)
```

  See the Filter Expressions section for more information on available filter expressions.

Assign the Size Log Handler to a Logger
In order for a log handler to be active, you must assign it to a logger.

The following management CLI command assigns the size log handler to the root logger.

```
/subsystem=logging/root-logger=ROOT:add-handler(name=SIZE_HANDLER_NAME)
```

The following management CLI command assigns the size log handler to a logger whose name is specified by **`CATEGORY`**.

```
/subsystem=logging/logger=CATEGORY:add-handler(name=SIZE_HANDLER_NAME)
```

Remove a Size Log Handler
A log handler can be removed with the **`remove`** operation. A log handler cannot be removed if it is currently assigned to a logger or async log handler.

```
/subsystem=logging/size-rotating-file-handler=SIZE_HANDLER_NAME:remove
```

12.5.5. Configure a Periodic Size Rotating Log Handler

This section shows you how to configure a periodic size rotating log handler using the management CLI. You can also configure periodic size log handlers using the management console by navigating to the **Logging** subsystem, selecting the **Handler** tab, and selecting **Periodic Size** from the left-hand menu.

The main tasks you will perform to configure a periodic size log handler are:

- Add a new periodic size log handler .
- Configure the periodic size log handler settings .
- Assign the periodic size log handler to a logger .

IMPORTANT

If you are configuring this log handler for a logging profile, the start of the command would be **`/subsystem=logging/logging-profile=LOGGING_PROFILE_NAME/`** instead of **`/subsystem=logging/`**.

Additionally, if you are running in a managed domain, precede the commands with **`/profile=PROFILE_NAME`**.

Add a Periodic Size Log Handler
When adding a periodic size log handler, you must specify the file path using the **`file`** attribute, which is comprised of the **`path`** and **`relative-to`** attributes. Use the **`path`** attribute to set the file path for the log, including the name, for example **`my-log.log`**. Optionally, use the **`relative-to`** attribute to set that the **`path`** is relative to a named path, for example **`jboss.server.log.dir`**.

You must also set the suffix for rotated logs using the **`suffix`** attribute. This must be in a format that can be understood by **`java.text.SimpleDateFormat`**, for example **`.yyyy-MM-dd-HH`**. The period of the rotation is automatically calculated based on this suffix.

```
/subsystem=logging/periodic-size-rotating-file-
handler=PERIODIC_SIZE_HANDLER_NAME:add(file={path=FILE_PATH,relative-
to=RELATIVE_TO_PATH},suffix=SUFFIX)
```

Configure Periodic Size Log Handler Settings
Depending on your needs, you may need to set one or more of the following periodic size log handler attributes. For a full list of available periodic size log handler attributes and their descriptions, see Periodic Size Log Handler Attributes.

- Set the log level.
  Set the appropriate log level for the handler. The default is **ALL**. See Log Levels for all available options.

  ```
  /subsystem=logging/periodic-size-rotating-file-
  handler=PERIODIC_SIZE_HANDLER_NAME:write-
  attribute(name=level,value=LEVEL)
  ```

- Set the rotation size.
  Set the maximum size that the file can reach before being rotated. The default is **2m** for 2 megabytes.

```
/subsystem=logging/periodic-size-rotating-file-handler=PERIODIC_SIZE_HANDLER_NAME:write-attribute(name=rotate-size, value=ROTATE_SIZE)
```

- Set the maximum number of backup logs to keep.
  Set the number of backups to keep. The default is **1**.

```
/subsystem=logging/periodic-size-rotating-file-handler=PERIODIC_SIZE_HANDLER_NAME:write-attribute(name=max-backup-index, value=MAX_BACKUPS)
```

- Set whether to rotate the log on boot.
  By default, a new log file is not created on server restart. You can set this to **true** to rotate the log on server restart.

```
/subsystem=logging/periodic-size-rotating-file-handler=PERIODIC_SIZE_HANDLER_NAME:write-attribute(name=rotate-on-boot, value=ROTATE_ON_BOOT)
```

- Set the append behavior.
  By default, JBoss EAP will append log messages to the same file when the server is restarted. You can set the **append** attribute to **false** to have the file overwritten upon server restart.

```
/subsystem=logging/periodic-size-rotating-file-handler=PERIODIC_SIZE_HANDLER_NAME:write-attribute(name=append,value=APPEND)
```

- Set the encoding.
  Set the encoding for the handler, for example, **utf-8**.

```
/subsystem=logging/periodic-size-rotating-file-handler=PERIODIC_SIZE_HANDLER_NAME:write-attribute(name=encoding,value=ENCODING)
```

- Set the log formatter.
  Set the formatter string for the handler. For example, the default format string is **%d{HH:mm:ss,SSS} %-5p [%c] (%t) %s%e%n**. Be sure to enclose the **FORMAT** value in quotation marks.

```
/subsystem=logging/periodic-size-rotating-file-handler=PERIODIC_SIZE_HANDLER_NAME:write-attribute(name=formatter,value=FORMAT)
```

NOTE

Use the **named-formatter** attribute if you want to reference a saved formatter.

- Set auto flush.
  Set whether to automatically flush after each write. The default value is **`true`**.

```
/subsystem=logging/periodic-size-rotating-file-
handler=PERIODIC_SIZE_HANDLER_NAME:write-
attribute(name=autoflush,value=AUTO_FLUSH)
```

- Set the filter expression.
  Set the expression for filtering log messages for the handler. Be sure to escape any commas and quotation marks and surround with quotation marks. For example, the **`FILTER_EXPRESSION`** replaceable variable below would need to be replaced with **`"not(match(\"WFLY\"))"`** for a filter expression of **`not(match("WFLY"))`**.

```
/subsystem=logging/periodic-size-rotating-file-
handler=PERIODIC_SIZE_HANDLER_NAME:write-attribute(name=filter-spec,
value=FILTER_EXPRESSION)
```

  See the Filter Expressions section for more information on available filter expressions.

Assign the Periodic Size Log Handler to a Logger
In order for a log handler to be active, you must assign it to a logger.

The following management CLI command assigns the periodic size log handler to the root logger.

```
/subsystem=logging/root-logger=ROOT:add-
handler(name=PERIODIC_SIZE_HANDLER_NAME)
```

The following management CLI command assigns the periodic size log handler to a logger whose name is specified by **`CATEGORY`**.

```
/subsystem=logging/logger=CATEGORY:add-
handler(name=PERIODIC_SIZE_HANDLER_NAME)
```

Remove a Periodic Size Log Handler
A log handler can be removed with the **`remove`** operation. A log handler cannot be removed if it is currently assigned to a logger or async log handler.

```
/subsystem=logging/periodic-size-rotating-file-
handler=PERIODIC_SIZE_HANDLER_NAME:remove
```

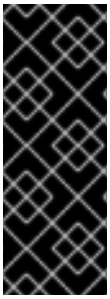
12.5.6. Configure a Syslog Handler

This section shows you how to configure a syslog handler using the management CLI, which can be used to send messages to a remote logging server that supports the Syslog protocol (RFC-3164 or RFC-5424). You can also configure syslog handlers using the management console by navigating to the **Logging** subsystem from the **Configuration** tab, selecting the **Handler** tab, and selecting **Syslog** from the left-hand menu.

The main tasks you will perform to configure a syslog handler are:

- Add a new syslog handler .
- Configure the syslog handler settings .

- Assign the syslog handler to a logger .



IMPORTANT

If you are configuring this log handler for a logging profile, the start of the command would be **`/subsystem=logging/logging-profile=LOGGING_PROFILE_NAME/`** instead of **`/subsystem=logging/`**.

Additionally, if you are running in a managed domain, precede the commands with **`/profile=PROFILE_NAME`**.

Add a Syslog Handler

```
/subsystem=logging/syslog-handler=SYSLOG_HANDLER_NAME:add
```

Configure Syslog Handler Settings
Depending on your needs, you may need to set one or more of the following syslog handler attributes. For a full list of available syslog handler attributes and their descriptions, see Syslog Handler Attributes.

- Set the log level for the handler. The default level is **ALL**. See Log Levels for all available options.

  ```
  /subsystem=logging/syslog-handler=SYSLOG_HANDLER_NAME:write-
  attribute(name=level,value=LEVEL)
  ```

- Set the name of the application that is logging. The default name is **`java`**.

  ```
  /subsystem=logging/syslog-handler=SYSLOG_HANDLER_NAME:write-
  attribute(name=app-name,value=APP_NAME)
  ```

- Set the address of the syslog server. The default address is **`localhost`**.

  ```
  /subsystem=logging/syslog-handler=SYSLOG_HANDLER_NAME:write-
  attribute(name=server-address,value=SERVER_ADDRESS)
  ```

- Set the port of the syslog server. The default port is **514**.

  ```
  /subsystem=logging/syslog-handler=SYSLOG_HANDLER_NAME:write-
  attribute(name=port,value=PORT)
  ```

- Set the syslog format, as defined by an RFC specification. The default format is **RFC5424**.

  ```
  /subsystem=logging/syslog-handler=SYSLOG_HANDLER_NAME:write-
  attribute(name=syslog-format,value=SYSLOG_FORMAT)
  ```

Assign the Syslog Handler to a Logger
In order for a log handler to be active, you must assign it to a logger.

The following management CLI command assigns the syslog handler to the root logger.

```
/subsystem=logging/root-logger=ROOT:add-handler(name=SYSLOG_HANDLER_NAME)
```

The following management CLI command assigns the syslog handler to a logger whose name is specified by **CATEGORY**.

```
/subsystem=logging/logger=CATEGORY:add-handler(name=SYSLOG_HANDLER_NAME)
```

Remove a Syslog Handler
A log handler can be removed with the **remove** operation. A log handler cannot be removed if it is currently assigned to a logger or async log handler.

```
/subsystem=logging/syslog-handler=SYSLOG_HANDLER_NAME:remove
```

12.5.7. Configure a Custom Log Handler

This section shows you how to configure a custom log handler using the management CLI. You can also configure custom log handlers using the management console by navigating to the **Logging** subsystem from the **Configuration** tab, selecting the **Handler** tab, and selecting **Custom** from the left-hand menu.

The main tasks you will perform to configure a custom log handler are:

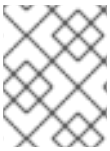
- Add a new custom log handler .
- Configure the custom log handler settings .
- Assign the custom log handler to a logger .

IMPORTANT

If you are configuring this log handler for a logging profile, the start of the command would be **`/subsystem=logging/logging-profile=LOGGING_PROFILE_NAME/`** instead of **`/subsystem=logging/`**.

Additionally, if you are running in a managed domain, precede the commands with **`/profile=PROFILE_NAME`**.

Add a Custom Log Handler
When adding a custom log handler, you must specify the Java class of the handler and the JBoss EAP module in which it is contained. The class must extend **`java.util.logging.Handler`**.



NOTE

You must have already created a module containing the custom logger or this command will fail.

```
/subsystem=logging/custom-
handler=CUSTOM_HANDLER_NAME:add(class=CLASS_NAME,module=MODULE_NAME)
```

Configure Custom Log Handler Settings
Depending on your needs, you may need to set one or more of the following custom log handler attributes. For a full list of available custom log handler attributes and their descriptions, see Custom Log Handler Attributes.

- Set the log level.

Set the appropriate log level for the handler. The default is **ALL**. See Log Levels for all available options.

```
/subsystem=logging/custom-handler=CUSTOM_HANDLER_NAME:write-attribute(name=level,value=LEVEL)
```

- Set the properties.
  Set the necessary properties for the log handler. The properties must be accessible using a setter method.

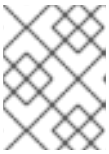
```
/subsystem=logging/custom-handler=CUSTOM_HANDLER_NAME:write-attribute(name=properties.PROPERTY_NAME,value=PROPERTY_VALUE)
```

- Set the encoding.
  Set the encoding for the handler, for example, **utf-8**.

```
/subsystem=logging/custom-handler=CUSTOM_HANDLER_NAME:write-attribute(name=encoding,value=ENCODING)
```

- Set the log formatter.
  Set the formatter string for the handler. For example, the default format string is **%d{HH:mm:ss,SSS} %-5p [%c] (%t) %s%e%n**. Be sure to enclose the **FORMAT** value in quotation marks.

```
/subsystem=logging/custom-handler=CUSTOM_HANDLER_NAME:write-attribute(name=formatter,value=FORMAT)
```



NOTE

Use the **named-formatter** attribute if you want to reference a saved formatter.

- Set the filter expression.
  Set the expression for filtering log messages for the handler. Be sure to escape any commas and quotation marks and surround with quotation marks. For example, the **FILTER_EXPRESSION** replaceable variable below would need to be replaced with **"not(match(\"WFLY\"))"** for a filter expression of **not(match("WFLY"))**.

```
/subsystem=logging/custom-handler=CUSTOM_HANDLER_NAME:write-attribute(name=filter-spec, value=FILTER_EXPRESSION)
```

See the Filter Expressions section for more information on available filter expressions.

Assign the Custom Log Handler to a Logger
In order for a log handler to be active, you must assign it to a logger.

The following management CLI command assigns the custom log handler to the root logger.

```
/subsystem=logging/root-logger=ROOT:add-handler(name=CUSTOM_HANDLER_NAME)
```

The following management CLI command assigns the custom log handler to a logger whose name is specified by **CATEGORY**.

```
/subsystem=logging/logger=CATEGORY:add-handler(name=CUSTOM_HANDLER_NAME)
```

Remove a Custom Log Handler
A log handler can be removed with the **remove** operation. A log handler cannot be removed if it is currently assigned to a logger or async log handler.

```
/subsystem=logging/custom-handler=CUSTOM_HANDLER_NAME:remove
```

12.5.8. Configure an Async Log Handler

This section shows you how to configure an async log handler using the management CLI. You can also configure async log handlers using the management console by navigating to the **Logging** subsystem from the **Configuration** tab, selecting the **Handler** tab, and selecting **Async** from the left-hand menu.

The main tasks you will perform to configure an async log handler are:

- Add a new async log handler .
- Add sub-handlers to the async log handler .
- Configure the async log handler settings .
- Assign the async log handler to a logger .

IMPORTANT

If you are configuring this log handler for a logging profile, the start of the command would be **/subsystem=logging/logging-profile=LOGGING_PROFILE_NAME/** instead of **/subsystem=logging/**.

Additionally, if you are running in a managed domain, precede the commands with **/profile=PROFILE_NAME**.

Add an Async Log Handler
When adding an async log handler, you must specify the queue length. This is the maximum number of log requests that can be held in queue.

```
/subsystem=logging/async-handler=ASYNC_HANDLER_NAME:add(queue-
length=QUEUE_LENGTH)
```

Add a Sub-handler
You can add one or more handlers as sub-handlers for this async log handler. Note that the handlers must already exist in the configuration or this command will fail.

```
/subsystem=logging/async-handler=ASYNC_HANDLER_NAME:add-
handler(name=HANDLER_NAME)
```

Configure Async Log Handler Settings
Depending on your needs, you may need to set one or more of the following async log handler attributes. For a full list of available async log handler attributes and their descriptions, see Async Log Handler Attributes.

- Set the log level.

Set the appropriate log level for the handler. The default is **ALL**. See Log Levels for all available options.

```
/subsystem=logging/async-handler=ASYNC_HANDLER_NAME:write-
attribute(name=level,value=LEVEL)
```

- Set the overflow action.
  Set the action to take when overflowing. The default value is **BLOCK**, which means that threads will block in the event of a full queue. You can change this value to **DISCARD**, which means that log messages will be discarded in the event of a full queue.

```
/subsystem=logging/async-handler=ASYNC_HANDLER_NAME:write-
attribute(name=overflow-action,value=OVERFLOW_ACTION)
```

- Set the filter expression.
  Set the expression for filtering log messages for the handler. Be sure to escape any commas and quotation marks and surround with quotation marks. For example, the **FILTER_EXPRESSION** replaceable variable below would need to be replaced with **`"not(match(\"WFLY\"))"`** for a filter expression of **`not(match("WFLY"))`**.

```
/subsystem=logging/async-handler=ASYNC_HANDLER_NAME:write-
attribute(name=filter-spec, value=FILTER_EXPRESSION)
```

  See the Filter Expressions section for more information on available filter expressions.

Assign the Async Log Handler to a Logger
In order for a log handler to be active, you must assign it to a logger.

The following management CLI command assigns the async log handler to the root logger.

```
/subsystem=logging/root-logger=ROOT:add-handler(name=ASYNC_HANDLER_NAME)
```

The following management CLI command assigns the async log handler to a logger whose name is specified by **CATEGORY**.

```
/subsystem=logging/logger=CATEGORY:add-handler(name=ASYNC_HANDLER_NAME)
```

Remove an Async Log Handler
A log handler can be removed with the **`remove`** operation. A log handler cannot be removed if it is currently assigned to a logger.

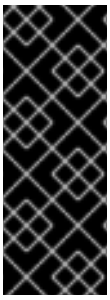
```
/subsystem=logging/async-handler=ASYNC_HANDLER_NAME:remove
```

12.6. CONFIGURING THE ROOT LOGGER

The root logger captures all log messages, of the specified log level or higher, sent to the server that are not captured by a log category.

This section shows you how to configure the root logger using the management CLI. You can also configure the root logger using the management console by navigating to the **Logging** subsystem from the **Configuration** tab and selecting the **Root Logger** tab.

Configure the Root Logger



IMPORTANT

If you are configuring this log handler for a logging profile, the start of the command would be **`/subsystem=logging/logging-profile=LOGGING_PROFILE_NAME/`** instead of **`/subsystem=logging/`**.

Additionally, if you are running in a managed domain, precede the commands with **`/profile=PROFILE_NAME`**.

1. Assign log handlers to the root logger.
   Add a log handler.

   ```
   /subsystem=logging/root-logger=ROOT:add-
   handler(name=LOG_HANDLER_NAME)
   ```

   Remove a log handler.

   ```
   /subsystem=logging/root-logger=ROOT:remove-
   handler(name=LOG_HANDLER_NAME)
   ```

2. Set the log level.

   ```
   /subsystem=logging/root-logger=ROOT:write-
   attribute(name=level,value=LEVEL)
   ```

For a full list of available root logger attributes and their descriptions, see Root Logger Attributes.

12.7. CONFIGURING LOG FORMATTERS

A log formatter defines the appearance of log messages from that handler. You can configure a named pattern formatter or a custom log formatter.

12.7.1. Configure a Named Pattern Formatter

You can create a named pattern formatter that can be used across log handlers to format log messages.

This section shows you how to configure a log formatter using the management CLI. You can also configure log formatters using the management console by navigating to the **Logging** subsystem from the **Configuration** tab, selecting the **Formatter** tab, and selecting **Pattern** from the left-hand menu.

IMPORTANT

If you are configuring this log formatter for a logging profile, the start of the command would be **`/subsystem=logging/logging-profile=LOGGING_PROFILE_NAME/`** instead of **`/subsystem=logging/`**.

Additionally, if you are running in a managed domain, precede the commands with **`/profile=PROFILE_NAME`**.

Create a Named Formatter
When defining a formatter, you provide a pattern string to use to format log messages. See log formatters for more information on the pattern syntax.

```
/subsystem=logging/pattern-
formatter=PATTERN_FORMATTER_NAME:add(pattern=PATTERN_STRING)
```

You can also define a color map to assign a color to different log levels. The format is a comma-separated list of **LEVEL:COLOR**.

- Valid levels: **finest**, **finer**, **fine**, **config**, **trace**, **debug**, **info**, **warning**, **warn**, **error**, **fatal**, **severe**
- Valid colors: **black**, **green**, **red**, **yellow**, **blue**, **magenta**, **cyan**, **white**, **brightblack**, **brightred**, **brightgreen**, **brightblue**, **brightyellow**, **brightmagenta**, **brightcyan**, **brightwhite**

```
/subsystem=logging/pattern-formatter=PATTERN_FORMATTER_NAME:write-
attribute(name=color-map,value="LEVEL:COLOR,LEVEL:COLOR")
```

Assign a Named Formatter to a Log Handler
The following management CLI command assigns a pattern formatter to be used by a periodic rotating file handler.

```
/subsystem=logging/periodic-rotating-file-handler=FILE_HANDLER_NAME:write-
attribute(name=named-formatter,value=PATTERN_FORMATTER_NAME)
```

12.7.2. Configure a Custom Log Formatter

You can create a custom log formatter that can be used across log handlers to format log messages.

This section shows you how to configure a custom log formatter using the management CLI. You can also configure log formatters using the management console by navigating to the **Logging** subsystem from the **Configuration** tab, selecting the **Formatter** tab, and selecting **Custom** from the left-hand menu.

Configure a Custom Log Formatter

IMPORTANT

If you are configuring this log formatter for a logging profile, the start of the command would be **/subsystem=logging/logging-profile=LOGGING_PROFILE_NAME/** instead of **/subsystem=logging/**.

Additionally, if you are running in a managed domain, precede the commands with **/profile=PROFILE_NAME**.

1. Add the custom log formatter.
   When adding a custom log formatter, you must specify the Java class of the formatter and the JBoss EAP module in which it is contained. The class must extend **java.util.logging.Formatter**.

NOTE

You must have already created a module containing the custom formatter or this command will fail.

```
/subsystem=logging/custom-
formatter=CUSTOM_FORMATTER_NAME:add(class=CLASS_NAME,
module=MODULE_NAME)
```

2. Set the necessary properties for the log formatter.
   The properties must be accessible using a setter method.

```
/subsystem=logging/custom-formatter=CUSTOM_FORMATTER_NAME:write-
attribute(name=properties.PROPERTY_NAME,value=PROPERTY_VALUE)
```

3. Assign the custom formatter to a log handler.
   The following management CLI command assigns a custom formatter to be used by a periodic rotating file handler.

```
/subsystem=logging/periodic-rotating-file-
handler=FILE_HANDLER_NAME:write-attribute(name=named-formatter,
value=CUSTOM_FORMATTER_NAME)
```

Example Custom XML Formatter
The following example configures a custom XML formatter. It uses the **`java.util.logging.XMLFormatter`** class provided in the **`org.jboss.logmanager`** module and assigns it to the console log handler.

```
/subsystem=logging/custom-formatter=custom-xml-
formatter:add(class=java.util.logging.XMLFormatter,
module=org.jboss.logmanager)
/subsystem=logging/console-handler=CONSOLE:write-attribute(name=named-
formatter, value=custom-xml-formatter)
```

A log message using this formatter would be formatted as below.

```
<record>
  <date>2016-03-23T12:58:13</date>
  <millis>1458752293091</millis>
  <sequence>93963</sequence>
  <logger>org.jboss.as</logger>
  <level>INFO</level>
  <class>org.jboss.as.server.BootstrapListener</class>
  <method>logAdminConsole</method>
  <thread>22</thread>
  <message>WFLYSRV0051: Admin console listening on http://%s:%d</message>
  <param>127.0.0.1</param>
  <param>9990</param>
</record>
```

12.8. ABOUT APPLICATION LOGGING

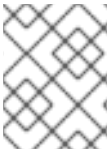
Logging for applications can be configured using the JBoss EAP **logging** subsystem or on a per-deployment basis.

See About the Logging Subsystem for using JBoss EAP log categories and handlers for capturing log messages.

For more information on application logging, such as supported application logging frameworks and configuring per-deployment logging, see the Logging chapter in the JBoss EAP *Development Guide*.

12.8.1. Per-deployment Logging

Per-deployment logging allows a developer to configure the logging configuration for their application in advance. 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.



NOTE

If the per-deployment logging configuration is not done, the configuration from **logging** subsystem is used for all the applications as well as the server.

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 any other logging than the server logging. A disadvantage is that the per-deployment logging configuration is read only on server startup, and so cannot be changed at runtime.

For instructions on using per-deployment logging in your applications, see Add Per-deployment Logging to an Application in the JBoss EAP *Development Guide*.

12.8.1.1. Disable Per-deployment Logging

You can disable per-deployment logging in one of the following ways:

- Set the **use-deployment-logging-config** attribute to **false**.
  The **use-deployment-logging-config** attribute controls whether or not your deployment is scanned for per-deployment logging. This defaults to **true** by default. You can set this attribute to **false** to disable per-deployment logging.

  ```
  /subsystem=logging:write-attribute(name=use-deployment-logging-config,value=false)
  ```

- Exclude the **logging** subsystem using a **jboss-deployment-structure.xml** file.
  For instructions, see Exclude a Subsystem from a Deployment in the JBoss EAP *Development Guide*.

12.8.2. Logging Profiles

Logging profiles are independent sets of logging configurations 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 it cannot refer to configurations in other profiles or the main **logging** subsystem. The design of logging profiles mimics the **logging** subsystem for ease of configuration.

Logging profiles allow administrators to create logging configurations that are specific to one or more

applications without affecting any other logging configurations. Because each profile is defined in the server configuration, the logging configuration can be changed without requiring that the affected applications be redeployed. However, logging profiles cannot be configured using the management console.

Each logging profile can have:

- A unique name (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.

12.8.2.1. Configure a Logging Profile

A logging profile can be configured with log handlers, categories, and a root logger. Configuring a logging profile uses the same syntax as configuring the **`logging`** subsystem, except for the following differences:

- The root configuration path is **`/subsystem=logging/logging-profile=NAME`**.
- A logging profile cannot contain other logging profiles.
- The **`logging`** subsystem has the following attributes that are not available for a logging profile:
  - **`add-logging-api-dependencies`**
  - **`use-deployment-logging-config`**

Creating and Configuring a Logging Profile
The following procedure uses the management CLI to create a logging profile and set a file handler and logger category.

1. Create the logging profile.

```
/subsystem=logging/logging-profile=PROFILE_NAME:add
```

2. Create the file handler.

```
/subsystem=logging/logging-profile=PROFILE_NAME/file-
handler=FILE_HANDLER_NAME:add(file={path=>"LOG_NAME.log", "relative-
to"=>"jboss.server.log.dir"})
```

```
/subsystem=logging/logging-profile=PROFILE_NAME/file-
handler=FILE_HANDLER_NAME:write-attribute(name="level",
value="DEBUG")
```

See File Log Handler Attributes for the list of file handler attributes.

3. Create the logger category.

```
/subsystem=logging/logging-
profile=PROFILE_NAME/logger=CATEGORY_NAME:add(level=TRACE)
```

   See Log Category Attributes for the list of log category attributes.

4. Assign the file handler to the category.

```
/subsystem=logging/logging-
profile=PROFILE_NAME/logger=CATEGORY_NAME:add-
handler(name="FILE_HANDLER_NAME")
```

You can then set the logging profile to use by an application in its **MANIFEST.MF** file. For more information, see Specify a Logging Profile in an Application in the JBoss EAP *Development Guide*.

12.8.2.2. Example Logging Profile Configuration

This example shows the configuration of a logging profile and the application that uses it. It shows the management CLI commands, the resulting XML, and the application's **MANIFEST.MF** file.

The example logging profile 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**.

Management CLI Session

```
/subsystem=logging/logging-profile=accounts-app-profile:add

/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")

/subsystem=logging/logging-profile=accounts-app-
profile/logger=com.company.accounts.ejbs:add(level=TRACE)

/subsystem=logging/logging-profile=accounts-app-
profile/logger=com.company.accounts.ejbs:add-handler(name="ejb-trace-
file")
```

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>
```

Application MANIFEST.MF file

```
Manifest-Version: 1.0
Logging-Profile: accounts-app-profile
```

12.8.3. Viewing Deployment Logging Configuration

You can obtain information about the logging configuration for a particular deployment using the following management CLI command.

```
/deployment=DEPLOYMENT_NAME/subsystem=logging/configuration=CONFIG:read-
resource
```

The logging configuration value (**CONFIG**) for a deployment can be one of three values:

- **default**, if the deployment is using the **logging** subsystem. This will output the **logging** subsystem configuration.
- **profile-PROFILE_NAME**, if the deployment is using a logging profile defined in the **logging** subsystem. This will output the logging profile configuration.
- The path to the configuration file being used, for example, **myear.ear/META-INF/logging.properties**.

For example, the below management CLI command displays the configuration for the **MYPROFILE** logging profile, which is used by the specified deployment.

```
/deployment=mydeployment.war/subsystem=logging/configuration=profile-
MYPROFILE:read-resource(recursive=true,include-runtime=true)
```

This will output the following information.

```
{
    "outcome" => "success",
    "result" => {
        "error-manager" => undefined,
        "filter" => undefined,
        "formatter" => {
            "MYFORMATTER" => {
                "class-name" =>
"org.jboss.logmanager.formatters.PatternFormatter",
                "module" => undefined,
```

```
                "properties" => {"pattern" => "%d{HH:mm:ss,SSS} %-5p [%c]
(%t) %s%e%n"}
            }
        },
        "handler" => {
            "MYPERIODIC" => {
                "class-name" =>
"org.jboss.logmanager.handlers.PeriodicRotatingFileHandler",
                "encoding" => undefined,
                "error-manager" => undefined,
                "filter" => undefined,
                "formatter" => "MYFORMATTER",
                "handlers" => [],
                "level" => "ALL",
                "module" => undefined,
                "properties" => {
                    "append" => "true",
                    "autoFlush" => "true",
                    "enabled" => "true",
                    "suffix" => ".yyyy-MM-dd",
                    "fileName" =>
"EAP_HOME/standalone/log/deployment.log"
                }
            }
        },
        "logger" => {"MYCATEGORY" => {
            "filter" => undefined,
            "handlers" => [],
            "level" => "DEBUG",
            "use-parent-handlers" => true
        }},
        "pojo" => undefined
    }
}
```

You could also use a recursive **`read-resource`** operation to read the logging configuration and other information about a deployment.

```
/deployment=DEPLOYMENT_NAME/subsystem=logging:read-resource(include-
runtime=true, recursive=true)
```

CHAPTER 13. DATASOURCE MANAGEMENT

13.1. ABOUT JBOSS EAP DATASOURCES

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.

For more information, see the JDBC 4.0 specification.

Supported Databases

See JBoss EAP supported configurations for the list of JDBC-compliant databases supported by JBoss EAP 7.

Datasource Types

The two general types of resources are referred to as *datasources* and *XA datasources*.

Non-XA datasources

Used for applications that do not use transactions, or applications that use transactions with a single database.

XA datasources

Used by applications that use multiple databases or other XA resources as part of one XA transaction. XA datasources introduce additional overhead.

You specify which type of datasource to use when creating the datasource using the JBoss EAP management interfaces.

The ExampleDS datasource

JBoss EAP ships with an example datasource configuration (*ExampleDS*), which is provided to demonstrate how datasources are defined. This datasource uses an H2 database, which is a lightweight, relational database management system that provides developers with the ability to quickly build applications.

WARNING

The *ExampleDS* datasource and the H2 database should *not* be used in a production environment. This 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.

13.2. JDBC DRIVERS

Before defining datasources in JBoss EAP for your applications to use, you must first install the appropriate JDBC driver.

13.2.1. Install a JDBC Driver as a Core Module

JDBC drivers can be installed as a core module using the management CLI using the following steps.

1. Download the JDBC driver.
   Download the appropriate JDBC driver from your database vendor. See JDBC Driver Download Locations for standard download locations for JDBC drivers of common databases.

   Make sure to extract the archive if the JDBC driver JAR file is contained within a ZIP or TAR archive.

2. Start the JBoss EAP server.

3. Launch the management CLI, but do not use the **`--connect`** or **`-c`** argument to connect to the running instance.

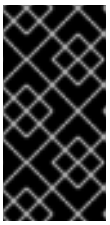
   ```
   $ EAP_HOME/bin/jboss-cli.sh
   ```

4. Use the **`module add`** management CLI command to add the new core module.

   ```
   module add --name=MODULE_NAME --resources=PATH_TO_JDBC_JAR --
   dependencies=DEPENDENCIES
   ```

   For example, the following command adds a MySQL JDBC driver module.

   ```
   module add --name=com.mysql --resources=/path/to/mysql-connector-
   java-5.1.36-bin.jar --dependencies=javax.api,javax.transaction.api
   ```

   IMPORTANT

   Using the **`module`** management CLI command to add and remove modules is provided as technology preview only. This command is not appropriate for use in a managed domain or when connecting to the management CLI remotely. Modules should be added and removed manually in a production environment.

   Execute **`module --help`** for more details on using this command to add and remove modules.

5. Use the **`connect`** management CLI command to connect to the running instance.

   ```
   connect
   ```

6. Register the JDBC driver. When running in a managed domain, be sure to precede this command with **`/profile=PROFILE_NAME`**.

   ```
   /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)
   ```

NOTE

The **driver-class-name** parameter is only required if the JDBC driver jar defines more than one class in its **/META-INF/services/java.sql.Driver** file.

For example, the **/META-INF/services/java.sql.Driver** file in the MySQL 5.1.36 JDBC driver JAR defines two classes:

- com.mysql.jdbc.Driver
- com.mysql.fabric.jdbc.FabricMySQLDriver

For this case, you would pass in **driver-class-name=com.mysql.jdbc.Driver**.

For example, the following command registers a MySQL JDBC driver.

```
/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)
```

The JDBC driver is now available to be referenced by application datasources.

13.2.2. Install a JDBC Driver as a JAR Deployment

JDBC drivers can be installed as a JAR deployment using either the management CLI or the management console. As long as the driver is JDBC 4-compliant, it will automatically be recognized and installed as a JDBC driver upon deployment.

The following steps describe how to install a JDBC driver using the management CLI.

NOTE

The recommended installation method for JDBC drivers is to install them as a core module.

1. Download the JDBC driver.
   Download the appropriate JDBC driver from your database vendor. See JDBC Driver Download Locations for standard download locations for JDBC drivers of common databases.

   Make sure to extract the archive if the JDBC driver JAR file is contained within a ZIP or TAR archive.

2. If the JDBC driver is not JDBC 4-compliant, see the steps to Update a JDBC Driver JAR to be JDBC 4-Compliant.

3. Deploy the JAR to JBoss EAP.

   ```
   deploy PATH_TO_JDBC_JAR
   ```

NOTE

In a managed domain, specify the appropriate server groups.

For example, the following command deploys a MySQL JDBC driver.

```
deploy /path/to/mysql-connector-java-5.1.36-bin.jar
```

A message will be displayed in the JBoss EAP server log that displays the deployed driver name, which will be used when defining datasources.

```
WFLYJCA0018: Started Driver service with driver-name = mysql-
connector-java-5.1.36-bin.jar_com.mysql.jdbc.Driver_5_1
```

The JDBC driver is now available to be referenced by application datasources.

Update a JDBC Driver JAR to be JDBC 4-Compliant
If the JDBC driver JAR is not JDBC 4-compliant, it can be made deployable using the following steps.

1. 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 and add one line to indicate the fully-qualified class name of the JDBC driver.
   For example, the below line would be added for a MySQL JDBC driver.

   ```
   com.mysql.jdbc.Driver
   ```

5. Use the JAR command-line tool to add this new file to the JAR.

   ```
   jar \-uf jdbc-driver.jar META-INF/services/java.sql.Driver
   ```

13.2.3. JDBC Driver Download Locations

The following table gives the standard download locations for JDBC drivers of common databases used with JBoss EAP.

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 13.1. JDBC Driver Download Locations

Vendor	Download Location
MySQL	http://www.mysql.com/products/connector/

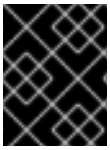
Vendor	Download Location
PostgreSQL	http://jdbc.postgresql.org/
Oracle	http://www.oracle.com/technetwork/database/features/jdbc/index-091264.html
IBM	http://www-306.ibm.com/software/data/db2/java/
Sybase	The jConnect JDBC driver is part of the SDK for SAP ASE installation. Currently there is not a separate download site for this driver by itself.
Microsoft	http://msdn.microsoft.com/data/jdbc/

13.2.4. Access Vendor-Specific Classes

In some cases, it is necessary for an application to use vendor-specific functionality that is not part of the JDBC API. In these cases, you can access vendor-specific APIs by declaring a dependency in that application.

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.

You can define a dependency for an application using either the **`MANIFEST.MF`** file or a **`jboss-deployment-structure.xml`** file.

If you have not yet done so, Install a JDBC Driver as a Core Module .

Using the `MANIFEST.MF` File

1. Edit the application's **`META-INF/MANIFEST.MF`** file.
2. Add the **`Dependencies`** line and specify the module name.
   For example, the below line declares the **`com.mysql`** module as a dependency.

```
Dependencies: com.mysql
```

Using a `jboss-deployment-structure.xml` File

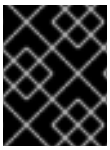
1. Create a file called **jboss-deployment-structure.xml** in the **META-INF/** or **WEB-INF/** folder of the application.

2. Specify the module using the **dependencies** element.
   For example, the following example **jboss-deployment-structure.xml** file declares the **com.mysql** module as a dependency.

```
<jboss-deployment-structure>
  <deployment>
    <dependencies>
      <module name="com.mysql"/>
    </dependencies>
  </deployment>
</jboss-deployment-structure>
```

The example code 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();
```



IMPORTANT

Follow the vendor-specific API guidelines closely, as the connection is being controlled by the IronJacamar container.

13.3. CREATING DATASOURCES

Datasources can be created using the management console or the management CLI.

JBoss EAP 7 allows you to use expressions in datasource attribute values, such as the **enabled** attribute. See the Property Replacement section for details on using expressions in the configuration.

13.3.1. Create a Non-XA Datasource

Non-XA datasources can be defined using the **data-source add** management CLI command. You can also define non-XA datasources using the management console by navigating to **Configuration** → **Subsystems** → **Datasources** → **Non-XA** and clicking **Add** to open the **Create Datasource** wizard.

The below steps describe how to define a non-XA datasource using the management CLI.

1. If you have not yet done so, install and register the appropriate JDBC Driver as a Core Module .

2. Define the datasource using the **data-source add** command, specifying the appropriate argument values.

```
data-source add --name=DATASOURCE_NAME --jndi-name=JNDI_NAME --
driver-name=DRIVER_NAME  --connection-url=CONNECTION_URL
```

NOTE

In a managed domain, you must specify the --profile=PROFILE_NAME argument.

See the Datasource Parameters section below for tips on these parameter values.

For detailed examples, see the Example Datasource Configurations for the supported databases.

Datasource Parameters

jndi-name

The JNDI name for the datasource must start with `java:/` or `java:jboss/`. For example, `java:jboss/datasources/ExampleDS`.

driver-name

The driver name value depends on whether the JDBC driver was installed as a core module or a JAR deployment.

1. For a core module, the driver name value will be the name given to the JDBC driver when it was registered.

2. For a JAR deployment, the driver name is the name of the JAR if there is only one class listed in its `/META-INF/services/java.sql.Driver` file. If there are multiple classes listed, then the value is JAR_NAME + "_" + DRIVER_CLASS_NAME + "_" + MAJOR_VERSION + "_" + MINOR_VERSION (for example `mysql-connector-java-5.1.36-bin.jar_com.mysql.jdbc.Driver_5_1`).
   You can also see the driver name listed in the JBoss EAP server log when the JDBC JAR is deployed.

   ```
   WFLYJCA0018: Started Driver service with driver-name = mysql-
   connector-java-5.1.36-bin.jar_com.mysql.jdbc.Driver_5_1
   ```

connection-url

For details on the connection URL formats for the supported databases, see the list of Datasource Connection URLs.

For a complete listing of all available datasource parameters, see the Datasource Parameters section.

13.3.2. Create an XA Datasource

XA datasources can be defined using the `xa-data-source add` management CLI command. You can also define XA datasources using the management console by navigating to **Configuration** → **Subsystems** → **Datasources** → **XA** and clicking **Add** to open the **Create XA Datasource** wizard.

The below steps describe how to define an XA datasource using the management CLI.

NOTE

In a managed domain, you will need to specify the profile to use. Depending on the format of the management CLI command, you will either precede the command with `/profile=PROFILE_NAME` or pass in the `--profile=PROFILE_NAME` argument.

1. If you have not yet done so, install and register the appropriate JDBC Driver as a Core Module .

2. Define the datasource using the **`xa-data-source add`** command, specifying the appropriate argument values.

```
xa-data-source add --name=XA_DATASOURCE_NAME --jndi-name=JNDI_NAME --driver-name=DRIVER_NAME --xa-datasource-class=XA_DATASOURCE_CLASS --xa-datasource-properties={"ServerName"=>"HOSTNAME","DatabaseName"=>"DATABASE_NAME"}
```

   See the Datasource Parameters section below for tips on these parameter values.

3. Set *XA datasource properties*.
   At least one *XA datasource property* is required when defining an XA datasource or you will receive an error when adding the datasource in the previous step. Any properties that were not set when defining the XA datasource can be set individually afterward.

   a. Set the server name.

```
/subsystem=datasources/xa-data-source=XA_DATASOURCE_NAME/xa-datasource-properties=ServerName:add(value=HOSTNAME)
```

   b. Set the database name.

```
/subsystem=datasources/xa-data-source=XA_DATASOURCE_NAME/xa-datasource-properties=DatabaseName:add(value=DATABASE_NAME)
```

For detailed examples, see the Example Datasource Configurations for the supported databases.

Datasource Parameters

jndi-name

The JNDI name for the datasource must start with **`java:/`** or **`java:jboss/`**. For example, **`java:jboss/datasources/ExampleDS`**.

driver-name

The driver name value depends on whether the JDBC driver was installed as a core module or a JAR deployment.

1. For a core module, the driver name value will be the name given to the JDBC driver when it was registered.

2. For a JAR deployment, the driver name is the name of the JAR if there is only one class listed in its **`/META-INF/services/java.sql.Driver`** file. If there are multiple classes listed, then the value is JAR_NAME + "_" + DRIVER_CLASS_NAME + "_" + MAJOR_VERSION + "_" + MINOR_VERSION (for example **`mysql-connector-java-5.1.36-bin.jar_com.mysql.jdbc.Driver_5_1`**).
   You can also see the driver name listed in the JBoss EAP server log when the JDBC JAR is deployed.

```
WFLYJCA0018: Started Driver service with driver-name = mysql-connector-java-5.1.36-bin.jar_com.mysql.jdbc.Driver_5_1
```

xa-datasource-class

Specify the XA datasource class for the JDBC driver's implementation of the **`javax.sql.XADataSource`** class.

xa-datasource-properties

At least one *XA datasource property* is required when defining an XA datasource or you will receive an error when attempting to add it. Additional properties can also be added to the XA datasource after it has been defined.

For a complete listing of all available datasource parameters, see the Datasource Parameters section.

13.4. MODIFYING DATASOURCES

Datasources settings can be configured using the management console or the management CLI.

JBoss EAP 7 allows you to use expressions in datasource attribute values, such as the **enabled** attribute. See the Property Replacement section for details on using expressions in the configuration.

13.4.1. Modify a Non-XA Datasource

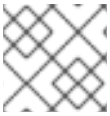
Non-XA datasource settings can be updated using the **data-source** management CLI command. You can also update datasource attributes from the **datasources** subsystem in the management console.

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**.

Settings for a datasource can be updated by using the following management CLI command.

```
data-source --name=DATASOURCE_NAME --ATTRIBUTE_NAME=ATTRIBUTE_VALUE
```



NOTE

In a managed domain, you must specify the **--profile=PROFILE_NAME** argument.

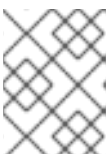
A server reload may be required for the changes to take effect.

13.4.2. Modify an XA Datasource

XA datasource settings can be updated using the **xa-data-source** management CLI command. You can also update datasource attributes from the **datasources** subsystem in the management console.

- Settings for an XA datasource can be updated by using the following management CLI command.

```
xa-data-source --name=XA_DATASOURCE_NAME --
ATTRIBUTE_NAME=ATTRIBUTE_VALUE
```



NOTE

In a managed domain, you must specify the **--profile=PROFILE_NAME** argument.

- An *XA datasource property* can be added using the following management CLI command.

```
/subsystem=datasources/xa-data-source=XA_DATASOURCE_NAME/xa-
datasource-properties=PROPERTY:add(value=VALUE)
```

NOTE

In a managed domain, you must precede this command with **/profile=PROFILE_NAME**.

A server reload may be required for the changes to take effect.

13.5. REMOVING DATASOURCES

Datasources can be removed using the management console or the management CLI.

13.5.1. Remove a Non-XA Datasource

Non-XA datasources can be removed using the **data-source remove** management CLI command. You can also remove datasources from the **datasources** subsystem in the management console.

```
data-source remove --name=DATASOURCE_NAME
```

NOTE

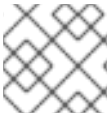
In a managed domain, you must specify the **--profile=PROFILE_NAME** argument.

The server will require a reload after the datasource is removed.

13.5.2. Remove an XA Datasource

XA datasources can be removed using the **xa-data-source remove** management CLI command. You can also remove XA datasources from the **datasources** subsystem in the management console.

```
xa-data-source remove --name=XA_DATASOURCE_NAME
```



NOTE

In a managed domain, you must specify the **--profile=PROFILE_NAME** argument.

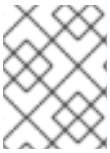
The server will require a reload after the XA datasource is removed.

13.6. TESTING DATASOURCE CONNECTIONS

Once a datasource has been added to JBoss EAP, you can test the connection to verify that the settings are correct. Datasource connections can be tested either using a management CLI command or from the management console with the **Test Connection** button in the **datasources** subsystem.

The following management CLI command can be used to test a datasource's connection.

```
/subsystem=datasources/data-source=DATASOURCE_NAME:test-connection-in-pool
```



NOTE

In a managed domain, you must precede this command with `/host=HOSTNAME/server=SERVER_NAME`.

13.7. XA DATASOURCE RECOVERY

An XA datasource is a datasource that can participate in an XA global transaction, which is coordinated by the transaction manager and can span multiple resources in a single transaction. If one of the participants fails to commit its changes, the other participants abort the transaction and restore their state as it was before the transaction occurred. This is to maintain consistency and avoid potential data loss or corruption.

XA recovery is the process of ensuring that all resources affected by a transaction are updated or rolled back, even if any of the resources or transaction participants crash or become unavailable. XA recovery happens without user intervention.

Each XA resource needs to have a recovery module associated with its configuration. The recovery module is the code that is executed when recovery is being performed. JBoss EAP automatically registers recovery modules for JDBC XA resources. You can register a custom module with your XA datasource if you wish to implement custom recovery code. The recovery module must extend class `com.arjuna.ats.jta.recovery.XAResourceRecovery`.

13.7.1. Configuring XA Recovery

For most JDBC 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.

The following table describes the XA datasource parameters specific to XA recovery. Each of these configuration attributes can be set during datasource creation or afterward. You can set them using either the management console or the management CLI. See Modify an XA Datasource for information on configuring XA datasources.

Table 13.2. Datasource Parameters for XA Recovery

Attribute	Description
recovery-username	The user name to use to connect to the resource for recovery. Note that if this is not specified, the datasource security settings will be used.
recovery-password	The password to use to connect to the resource for recovery. Note that if this is not specified, the datasource security settings will be used.
recovery-security-domain	The security domain to 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`.

Attribute	Description
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 **KEY=VALUE** pairs for the properties.

Disable XA Recovery

If multiple XA datasources connect to the same physical database, then XA recovery typically needs to be configured for only one of them.

Use the following management CLI command to disable recovery for an XA datasource:

```
/subsystem=datasources/xa-data-source=XA_DATASOURCE_NAME:write-
attribute(name=no-recovery,value=true)
```

13.7.2. Vendor-Specific XA Recovery

Vendor-Specific Configuration
Some databases require specific configurations in order to cooperate in XA transactions managed by the JBoss EAP transaction manager. For detailed and up-to-date information, see your database vendor's documentation.

MySQL

No special configuration is required. For more information, see the MySQL documentation.

PostgreSQL and Postgres Plus Advanced Server

For PostgreSQL to be able to handle XA transactions, change the configuration parameter **`max_prepared_transactions`** to a value greater than **`0`** and equal to or greater than **`max_connections`**.

Oracle

Ensure that the Oracle user (**USER**) has access to the tables needed for recovery.

```
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 the Oracle user does not have the proper permissions, you may see an error such as the following:

```
WARN  [com.arjuna.ats.jta.logging.loggerI18N]
[com.arjuna.ats.internal.jta.recovery.xarecovery1] Local
XARecoveryModule.xaRecovery  got XA exception
javax.transaction.xa.XAException, XAException.XAER_RMERR
```

Microsoft SQL Server

For more information, see the Microsoft SQL Server documentation, including http://msdn.microsoft.com/en-us/library/aa342335.aspx.

IBM DB2

No special configuration is required. For more information, see the IBM DB2 documentation.

Sybase

Sybase expects XA transactions to be enabled on the database. Without correct database configuration, XA transactions will not work. The **`enable xact coordination`** parameter 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
```

MariaDB

No special configuration is required. For more information, see the MariaDB documentation.

Known Issues
These known issues with handling XA transactions are for the specific database and JDBC driver versions supported with JBoss EAP 7. For up-to-date information on the supported databases, see JBoss EAP supported configurations.

MySQL

MySQL is not fully capable of handling XA transactions. If a client is disconnected from MySQL, then all the information about such transactions is lost. See this MySQL bug for more information. Note that this issue was fixed in MySQL 5.7.

PostgreSQL and Postgres Plus Advanced Server

The JDBC driver returns the **`XAER_RMERR`** XAException error code when a network failure occurs during the commit phase of two-phase commit (2PC). This error signals an unrecoverable catastrophic event to the transaction manager, but the transaction is left in the **`in-doubt`** state on the database side and could be easily corrected after network connection is established again. The correct return code should be **`XAER_RMFAIL`** or **`XAER_RETRY`**. The incorrect error code causes the transaction to be left in the **`Heuristic`** state on the JBoss EAP side and holding locks in the database which requires manual intervention. See this PostgreSQL bug for more information.
If a connection failure happens when the one-phase commit optimization is used, the JDBC driver returns **`XAER_RMERR`**, but the **`XAER_RMFAIL`** error code should be returned. This could lead to data inconsistency if the database commits data during one-phase commit and the connection goes down at that moment, then the client is informed that transaction was rolled back.

The Postgres Plus JDBC driver returns XIDs for all prepared transactions that exist in the Postgres Plus Server, so there is no way to determine the database to which the XID belongs. If you define more than one data source for the same database in JBoss EAP, an in-doubt transaction recovery attempt could be run under wrong account, which causes the recovery to fail.

Oracle

The JDBC driver returns XIDs belonging to all users on the database instance, when the Recovery Manager calls recovery using datasource configured with some user credentials. The JDBC driver throws the exception **`ORA-24774: cannot switch to specified transaction`** because it tries to recover XIDs belonging to other users.
The workaround for this issue is to grant **`FORCE ANY TRANSACTION`** privilege to the user whose credentials are used in recovery datasource configuration. More information about configuring the privilege can be found here:
http://docs.oracle.com/database/121/ADMIN/ds_txnman.htm#ADMIN12259

Microsoft SQL Server

The JDBC driver returns the **XAER_RMERR** XAException error code when a network failure occurs during the commit phase of two-phase commit (2PC). This error signals an unrecoverable catastrophic event to the transaction manager, but the transaction is left in the **in-doubt** state on the database side and could be easily corrected after network connection is established again. The correct return code should be **XAER_RMFAIL** or **XAER_RETRY**. The incorrect error code causes the transaction to be left in the **Heuristic** state on the JBoss EAP side and holding locks in the database which requires manual intervention. See this Microsoft SQL Server issue report for more information.
If a connection failure happens when the one-phase commit optimization is used, the JDBC driver returns **XAER_RMERR**, but the **XAER_RMFAIL** error code should be returned. This could lead to data inconsistency if the database commits data during one-phase commit and the connection goes down at that moment, then the client is informed that transaction was rolled back.

IBM DB2

If a connection failure happens during a one-phase commit, the JDBC driver returns **XAER_RETRY**, but the **XAER_RMFAIL** error code should be returned. This could lead to data inconsistency if the database commits data during one-phase commit and the connection goes down at that moment, then the client is informed that transaction was rolled back.

Sybase

The JDBC driver returns the **XAER_RMERR** XAException error code when a network failure occurs during the commit phase of two-phase commit (2PC). This error signals an unrecoverable catastrophic event to the transaction manager, but the transaction is left in the **in-doubt** state on the database side and could be easily corrected after network connection is established again. The correct return code should be **XAER_RMFAIL** or **XAER_RETRY**. The incorrect error code causes the transaction to be left in the **Heuristic** state on the JBoss EAP side and holding locks in the database which requires manual intervention.
If a connection failure happens when the one-phase commit optimization is used, the JDBC driver returns **XAER_RMERR**, but the **XAER_RMFAIL** error code should be returned. This could lead to data inconsistency if the database commits data during one-phase commit and the connection goes down at that moment, then the client is informed that transaction was rolled back.

MariaDB

MariaDB is not fully capable of handling XA transactions. If a client is disconnected from MariaDB, then all the information about such transactions is lost.

13.8. DATABASE CONNECTION VALIDATION

Database maintenance, network problems, or other outage events may cause JBoss EAP to lose the connection to the database. In order to recover from these situations, you can enable database connection validation for your datasources.

To configure database connection validation, you specify the validation timing method (when the validation occurs), the validation mechanism (how the validation is performed), and the exception sorter (how exceptions are handled).

1. Choose *one* of the validation timing methods.

   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 the next connection in the pool is retrieved. 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.

background-validation

When the **`background-validation`** option is set to **`true`**, connections are validated periodically in a background thread prior to use. The frequency of the validation is specified by the **`background-validation-millis`** property. The default value of **`background-validation-millis`** is **`0`**, meaning that it is disabled.
When determining the value of the **`background-validation-millis`** property, consider the following:

- This value should not be set to the same value as your **`idle-timeout-minutes`** setting.
- The lower the value, the more frequently the pool is validated and the sooner invalid connections are removed from the pool.
- 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

These options are mutually exclusive. If **`validate-on-match`** is set to **`true`**, then **`background-validation`** must be set to **`false`**. If **`background-validation`** is set to true, then **`validate-on-match`** must be set to **`false`**.

2. Choose *one* of the validation mechanisms.

valid-connection-checker-class-name

Using **`valid-connection-checker-class-name`** is the preferred validation mechanism. This specifies a connection checker class that is used to validate connections for the particular database in use. JBoss EAP 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**

check-valid-connection-sql

Using **check-valid-connection-sql**, you provide the SQL statement that will be used to validate the connection.
The following is an example SQL statement that you might use to validate Oracle connections.

```
select 1 from dual
```

The following is an example SQL statement that you might use to validate MySQL or PostgreSQL connections.

```
select 1
```

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. Choose the appropriate JBoss EAP exception sorter for your datasource type.

- **org.jboss.jca.adapters.jdbc.extensions.db2.DB2ExceptionSorter**
- **org.jboss.jca.adapters.jdbc.extensions.informix.InformixExceptionSorter**
- **org.jboss.jca.adapters.jdbc.extensions.mssql.MSSQLExceptionSorter**
- **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**

13.9. DATASOURCE SECURITY

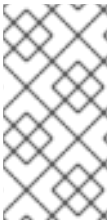
Datasource security refers to encrypting or obscuring passwords for datasource connections. These passwords can be stored in plain text in configuration files, however this represents a security risk.

The preferred solution for datasource security is the use of either security domains or password vaults. Examples of each are included below.

Secure a Datasource Using a Security Domain

A security domain for the datasource is defined.

```
 <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>
```



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.

The **DsRealm** security domain is then referenced by the datasource configuration.

```
<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>
```

For more information on using Security Domains, see the *How to Configure Identity Management* guide.

Secure a Datasource Using a Password Vault

```
<security>
  <user-name>admin</user-name>

<password>${VAULT::ds_ExampleDS::password::N2NhZDYzOTMtNWE0OS00ZGQ0LWE4MmE
tMWNlMDMyNDdmNmI2TElORV9CUkVBS3ZhdWx0}</password>
</security>
```

For more information on using the Password Vault, see the *How To Configure Server Security* guide.

13.10. DATASOURCE STATISTICS

You can view core *pool* and *JDBC* runtime statistics for defined datasources. See the Datasource Statistics for a detailed list of all available statistics.

Enable Datasource Statistics

By default, datasource statistics are *not* enabled. The following management CLI commands enable the collection of statistics for the **ExampleDS** datasource.

NOTE

In a managed domain, precede these commands with **/profile=PROFILE_NAME**.

```
/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)
```

View Datasource Statistics

All datasource statistics can be retrieved from the management CLI. A subset of these statistics can be viewed from the **Runtime** tab of the management console.

The following management CLI command retrieves the core *pool* statistics for the **ExampleDS** datasource.

NOTE

In a managed domain, precede these commands with **/host=HOST_NAME/server=SERVER_NAME**.

```
/subsystem=datasources/data-source=ExampleDS/statistics=pool:read-
resource(include-runtime=true)
{
    "outcome" => "success",
    "result" => {
        "ActiveCount" => 1,
        "AvailableCount" => 20,
        "AverageBlockingTime" => 0L,
        "AverageCreationTime" => 122L,
        "AverageGetTime" => 128L,
        "AveragePoolTime" => 0L,
        "AverageUsageTime" => 0L,
        "BlockingFailureCount" => 0,
        "CreatedCount" => 1,
        "DestroyedCount" => 0,
        "IdleCount" => 1,
        ...
}
```

The following management CLI command retrieves the *JDBC* statistics for the **ExampleDS** datasource.

```
/subsystem=datasources/data-source=ExampleDS/statistics=jdbc:read-
resource(include-runtime=true)
{
```

```
    "outcome" => "success",
    "result" => {
        "PreparedStatementCacheAccessCount" => 0L,
        "PreparedStatementCacheAddCount" => 0L,
        "PreparedStatementCacheCurrentSize" => 0,
        "PreparedStatementCacheDeleteCount" => 0L,
        "PreparedStatementCacheHitCount" => 0L,
        "PreparedStatementCacheMissCount" => 0L,
        "statistics-enabled" => true
    }
}
```

NOTE

Since statistics are runtime information, be sure to specify the **`include-runtime=true`** argument.

13.11. CAPACITY POLICIES

JBoss EAP supports defining capacity polices for JCA deployments, including datasources. Capacity policies define how physical connections for a pool are created (capacity incrementing) and destroyed (capacity decrementing). The default policies are set to create once connection per request for capacity incrementing, and destroy all connections when they time out when the idle timeout is scheduled for capacity decrementing.

To configure capacity polices, you need to specify a capacity incrementer and/or decrementer class:

Example commands

```
/subsystem=datasources/data-source=ExampleDS:write-
attribute(name=capacity-incrementer-class,
value="org.jboss.jca.core.connectionmanager.pool.capacity.SizeIncrementer"
)

/subsystem=datasources/data-source=ExampleDS:write-
attribute(name=capacity-decrementer-class,
value="org.jboss.jca.core.connectionmanager.pool.capacity.SizeDecrementer"
)
```

You can also configure properties on the specified capacity incrementer or decrementer class:

Example Commands

```
/subsystem=datasources/data-source=ExampleDS:write-
attribute(name=capacity-incrementer-properties.size, value=2)

/subsystem=datasources/data-source=ExampleDS:write-
attribute(name=capacity-decrementer-properties.size, value=2)
```

MaxPoolSize Incrementer Policy
Class name:
`org.jboss.jca.core.connectionmanager.pool.capacity.MaxPoolSizeIncrementer`

The *MaxPoolSize* incrementer policy will fill the pool to its max size for each request. This policy is useful when you want to keep the maximum number of connections available all the time.

Size Incrementer Policy

Class name: `org.jboss.jca.core.connectionmanager.pool.capacity.SizeIncrementer`

The *Size* incrementer policy will fill the pool by the specified number of connections for each request. This policy is useful when you want to increment with an additional number of connections per request in anticipation that the next request will also need a connection.

Table 13.3. Size policy properties

Name	Description
Size	The number of connections that should be created

NOTE

This is the default increment policy with a *size* value of 1.

Watermark Incrementer Policy

Class name:
`org.jboss.jca.core.connectionmanager.pool.capacity.WatermarkIncrementer`

The *Watermark* incrementer policy will fill the pool to the specified number of connections for each request. This policy is useful when you want to keep a specified number of connections in the pool at all time.

Table 13.4. Watermark policy properties

Name	Description
Watermark	The watermark level for the number of connections

MinPoolSize Decrementer Policy

Class name:
`org.jboss.jca.core.connectionmanager.pool.capacity.MinPoolSizeDecrementer`

The *MinPoolSize* decrementer policy will decrement the pool to its min size for each request. This policy is useful when you want to limit the number of connections after each idle timeout request. The pool will operate in a First In First Out (FIFO) manner.

Size Decrementer Policy

Class name: `org.jboss.jca.core.connectionmanager.pool.capacity.SizeDecrementer`

The *Size* decrementer policy will decrement the pool by the specified number of connections for each idle timeout request.

Table 13.5. Size policy properties

Name	Description
Size	The number of connections that should be destroyed

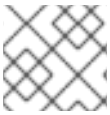
This policy is useful when you want to decrement an additional number of connections per idle timeout request in anticipation that the pool usage will lower over time.

The pool will operate in a First In First Out (FIFO) manner.

TimedOut Decrementer Policy

Class name:
`org.jboss.jca.core.connectionmanager.pool.capacity.TimedOutDecrementer`

The *TimedOut* decrementer policy will removed all connections that have timed out from the pool for each idle timeout request. The pool will operate in a First In Last Out (FILO) manner.



NOTE

This policy is the default decrement policy.

TimedOut/FIFO Decrementer Policy

Class name:
`org.jboss.jca.core.connectionmanager.pool.capacity.TimedOutFIFODecrementer`

The *TimedOutFIFO* decrementer policy will removed all connections that have timed out from the pool for each idle timeout request. The pool will operate in a First In First Out (FIFO) manner.

Watermark Decrementer Policy

Class name:
`org.jboss.jca.core.connectionmanager.pool.capacity.WatermarkDecrementer`

The *Watermark* decrementer policy will decrement the pool to the specified number of connections for each idle timeout request. This policy is useful when you want to keep a specified number of connections in the pool at all time. The pool will operate in a First In First Out (FIFO) manner.

Table 13.6. Watermark policy properties

Name	Description
Watermark	The watermark level for the number of connections

13.12. ENLISTMENT TRACING

Enlistment traces are recorded in order to help locate error situations that happens during enlistment of **`XAResource`** instances. This does have a performance overhead, so in certain situations you may want to disable these traces.

You can disable the recording of enlistment traces for a datasource using the management CLI by setting the **`enlistment-trace`** attribute to **`false`**.

Disable enlistment trace for a non-XA datasource.

```
data-source --name=DATASOURCE_NAME --enlistment-trace=false
```

Disable enlistment trace for an XA datasource.

```
xa-data-source --name=XA_DATASOURCE_NAME --enlistment-trace=false
```

WARNING

Disabling the enlistment trace will make tracking down errors during transaction enlistment more difficult.

13.13. EXAMPLE DATASOURCE CONFIGURATIONS

13.13.1. Example MySQL Datasource

This is an example of a MySQL datasource configuration with connection information, basic security, and validation options.

Example MySQL Datasource Configuration

```
<datasources>
  <datasource jndi-name="java:jboss/MySqlDS" pool-name="MySqlDS">
    <connection-url>jdbc:mysql://localhost:3306/jbossdb</connection-url>
    <driver>mysql</driver>
    <security>
      <user-name>admin</user-name>
      <password>admin</password>
    </security>
    <validation>
      <valid-connection-checker class-
name="org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLValidConnectionChe
cker"/>
      <validate-on-match>true</validate-on-match>
      <background-validation>false</background-validation>
      <exception-sorter class-
name="org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLExceptionSorter"/>
    </validation>
  </datasource>
  <drivers>
    <driver name="mysql" module="com.mysql">
      <driver-class>com.mysql.jdbc.Driver</driver-class>
      <xa-datasource-
class>com.mysql.jdbc.jdbc2.optional.MysqlXADataSource</xa-datasource-
class>
    </driver>
  </drivers>
</datasources>
```

Example MySQL JDBC Driver `module.xml` File

```
<?xml version="1.0" ?>
<module xmlns="urn:jboss:module:1.1" name="com.mysql">
  <resources>
    <resource-root path="mysql-connector-java-5.1.36-bin.jar"/>
  </resources>
  <dependencies>
    <module name="javax.api"/>
    <module name="javax.transaction.api"/>
  </dependencies>
</module>
```

Example Management CLI Commands
This example configuration can be achieved by using the following management CLI commands.

1. Add the MySQL JDBC driver as a core module.

```
module add --name=com.mysql --resources=/path/to/mysql-connector-
java-5.1.36-bin.jar --dependencies=javax.api,javax.transaction.api
```

IMPORTANT

Using the **module** management CLI command to add and remove modules is provided as technology preview only. This command is not appropriate for use in a managed domain or when connecting to the management CLI remotely. Modules should be added and removed manually in a production environment.

2. Register the MySQL JDBC driver.

```
/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)
```

3. Add the MySQL datasource.

```
data-source add --name=MySqlDS --jndi-name=java:jboss/MySqlDS --
driver-name=mysql --connection-
url=jdbc:mysql://localhost:3306/jbossdb --user-name=admin --
password=admin --validate-on-match=true --background-
validation=false --valid-connection-checker-class-
name=org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLValidConnecti
onChecker --exception-sorter-class-
name=org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLExceptionSort
er
```

13.13.2. Example MySQL XA Datasource

This is an example of a MySQL XA datasource configuration with XA datasource properties, basic security, and validation options.

Example MySQL XA Datasource Configuration

```
<datasources>
```

```
  <xa-datasource jndi-name="java:jboss/MySqlXADS" pool-name="MySqlXADS">
    <xa-datasource-property name="ServerName">
      localhost
    </xa-datasource-property>
    <xa-datasource-property name="DatabaseName">
      mysqldb
    </xa-datasource-property>
    <driver>mysql</driver>
    <security>
      <user-name>admin</user-name>
      <password>admin</password>
    </security>
    <validation>
      <valid-connection-checker class-
name="org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLValidConnectionChe
cker"/>
      <validate-on-match>true</validate-on-match>
      <background-validation>false</background-validation>
      <exception-sorter class-
name="org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLExceptionSorter"/>
    </validation>
  </xa-datasource>
  <drivers>
    <driver name="mysql" module="com.mysql">
      <driver-class>com.mysql.jdbc.Driver</driver-class>
      <xa-datasource-
class>com.mysql.jdbc.jdbc2.optional.MysqlXADataSource</xa-datasource-
class>
    </driver>
  </drivers>
</datasources>
```

Example MySQL JDBC Driver `module.xml` File

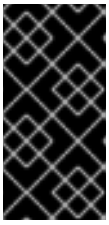
```
<?xml version="1.0" ?>
<module xmlns="urn:jboss:module:1.1" name="com.mysql">
  <resources>
    <resource-root path="mysql-connector-java-5.1.36-bin.jar"/>
  </resources>
  <dependencies>
    <module name="javax.api"/>
    <module name="javax.transaction.api"/>
  </dependencies>
</module>
```

Example Management CLI Commands

This example configuration can be achieved by using the following management CLI commands.

1. Add the MySQL JDBC driver as a core module.

```
module add --name=com.mysql --resources=/path/to/mysql-connector-
java-5.1.36-bin.jar --dependencies=javax.api,javax.transaction.api
```



IMPORTANT

Using the **module** management CLI command to add and remove modules is provided as technology preview only. This command is not appropriate for use in a managed domain or when connecting to the management CLI remotely. Modules should be added and removed manually in a production environment.

2. Register the MySQL JDBC driver.

```
/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)
```

3. Add the MySQL XA datasource.

```
xa-data-source add --name=MySqlXADS --jndi-name=java:jboss/MySqlXADS
--driver-name=mysql --user-name=admin --password=admin --validate-
on-match=true --background-validation=false --valid-connection-
checker-class-
name=org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLValidConnecti
onChecker --exception-sorter-class-
name=org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLExceptionSort
er --xa-datasource-properties=
{"ServerName"=>"localhost","DatabaseName"=>"mysqldb"}
```

13.13.3. Example PostgreSQL Datasource

This is an example of a PostgreSQL datasource configuration with connection information, basic security, and validation options.

Example PostgreSQL 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>
        <valid-connection-checker class-
name="org.jboss.jca.adapters.jdbc.extensions.postgres.PostgreSQLValidConne
ctionChecker"/>
        <validate-on-match>true</validate-on-match>
        <background-validation>false</background-validation>
        <exception-sorter class-
name="org.jboss.jca.adapters.jdbc.extensions.postgres.PostgreSQLExceptionS
orter"/>
    </validation>
  </datasource>
  <drivers>
    <driver name="postgresql" module="com.postgresql">
```

```
        <xa-datasource-class>org.postgresql.xa.PGXADataSource</xa-
datasource-class>
    </driver>
  </drivers>
</datasources>
```

Example PostgreSQL JDBC Driver `module.xml` File

```
<?xml version="1.0" ?>
<module xmlns="urn:jboss:module:1.1" name="com.postgresql">
  <resources>
    <resource-root path="postgresql-9.3-1102.jdbc4.jar"/>
  </resources>
  <dependencies>
    <module name="javax.api"/>
    <module name="javax.transaction.api"/>
  </dependencies>
</module>
```

Example Management CLI Commands
This example configuration can be achieved by using the following management CLI commands.

1. Add the PostgreSQL JDBC driver as a core module.

```
module add --name=com.postgresql --resources=/path/to/postgresql-
9.3-1102.jdbc4.jar --dependencies=javax.api,javax.transaction.api
```

IMPORTANT

Using the **`module`** management CLI command to add and remove modules is provided as technology preview only. This command is not appropriate for use in a managed domain or when connecting to the management CLI remotely. Modules should be added and removed manually in a production environment.

2. Register the PostgreSQL JDBC driver.

```
/subsystem=datasources/jdbc-driver=postgresql:add(driver-
name=postgresql,driver-module-name=com.postgresql,driver-xa-
datasource-class-name=org.postgresql.xa.PGXADataSource)
```

3. Add the PostgreSQL datasource.

```
data-source add --name=PostgresDS --jndi-name=java:jboss/PostgresDS
--driver-name=postgresql --connection-
url=jdbc:postgresql://localhost:5432/postgresdb --user-name=admin --
password=admin --validate-on-match=true --background-
validation=false --valid-connection-checker-class-
name=org.jboss.jca.adapters.jdbc.extensions.postgres.PostgreSQLValid
ConnectionChecker --exception-sorter-class-
name=org.jboss.jca.adapters.jdbc.extensions.postgres.PostgreSQLExcep
tionSorter
```

13.13.4. Example PostgreSQL XA Datasource

This is an example of a PostgreSQL XA datasource configuration with XA datasource properties, basic security, and validation options.

Example PostgreSQL XA Datasource Configuration

```
<datasources>
  <xa-datasource jndi-name="java:jboss/PostgresXADS" pool-
name="PostgresXADS">
    <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>
    <driver>postgresql</driver>
    <security>
      <user-name>admin</user-name>
      <password>admin</password>
    </security>
    <validation>
      <valid-connection-checker class-
name="org.jboss.jca.adapters.jdbc.extensions.postgres.PostgreSQLValidConne
ctionChecker"/>
      <validate-on-match>true</validate-on-match>
      <background-validation>false</background-validation>
      <exception-sorter class-
name="org.jboss.jca.adapters.jdbc.extensions.postgres.PostgreSQLExceptionS
orter"/>
    </validation>
  </xa-datasource>
  <drivers>
    <driver name="postgresql" module="com.postgresql">
      <xa-datasource-class>org.postgresql.xa.PGXADataSource</xa-
datasource-class>
    </driver>
  </drivers>
</datasources>
```

Example PostgreSQL JDBC Driver `module.xml` File

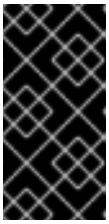
```
<?xml version="1.0" ?>
<module xmlns="urn:jboss:module:1.1" name="com.postgresql">
  <resources>
    <resource-root path="postgresql-9.3-1102.jdbc4.jar"/>
  </resources>
  <dependencies>
    <module name="javax.api"/>
    <module name="javax.transaction.api"/>
  </dependencies>
</module>
```

Example Management CLI Commands

This example configuration can be achieved by using the following management CLI commands.

1. Add the PostgreSQL JDBC driver as a core module.

```
module add --name=com.postgresql --resources=/path/to/postgresql-
9.3-1102.jdbc4.jar --dependencies=javax.api,javax.transaction.api
```



IMPORTANT

Using the **module** management CLI command to add and remove modules is provided as technology preview only. This command is not appropriate for use in a managed domain or when connecting to the management CLI remotely. Modules should be added and removed manually in a production environment.

2. Register the PostgreSQL JDBC driver.

```
/subsystem=datasources/jdbc-driver=postgresql:add(driver-
name=postgresql,driver-module-name=com.postgresql,driver-xa-
datasource-class-name=org.postgresql.xa.PGXADataSource)
```

3. Add the PostgreSQL XA datasource.

```
xa-data-source add --name=PostgresXADS --jndi-
name=java:jboss/PostgresXADS --driver-name=postgresql --user-
name=admin --password=admin --validate-on-match=true --background-
validation=false --valid-connection-checker-class-
name=org.jboss.jca.adapters.jdbc.extensions.postgres.PostgreSQLValid
ConnectionChecker --exception-sorter-class-
name=org.jboss.jca.adapters.jdbc.extensions.postgres.PostgreSQLExcep
tionSorter --xa-datasource-properties=
{"ServerName"=>"localhost","PortNumber"=>"5432","DatabaseName"=>"pos
tgresdb"}
```

13.13.5. Example Oracle Datasource

This is an example of an Oracle datasource configuration with connection information, basic security, and validation options.

Example Oracle Datasource Configuration

```
<datasources>
  <datasource jndi-name="java:jboss/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>
        <valid-connection-checker class-
name="org.jboss.jca.adapters.jdbc.extensions.oracle.OracleValidConnectionC
hecker"/>
        <validate-on-match>true</validate-on-match>
        <background-validation>false</background-validation>
        <stale-connection-checker class-
name="org.jboss.jca.adapters.jdbc.extensions.oracle.OracleStaleConnectionC
```

```
hecker"/>
        <exception-sorter class-
name="org.jboss.jca.adapters.jdbc.extensions.oracle.OracleExceptionSorter"
/>
    </validation>
  </datasource>
  <drivers>
    <driver name="oracle" module="com.oracle">
        <xa-datasource-class>oracle.jdbc.xa.client.OracleXADataSource</xa-
datasource-class>
    </driver>
  </drivers>
</datasources>
```

Example Oracle JDBC Driver `module.xml` File

```
<?xml version="1.0" ?>
<module xmlns="urn:jboss:module:1.1" name="com.oracle">
    <resources>
        <resource-root path="ojdbc7.jar"/>
    </resources>
    <dependencies>
        <module name="javax.api"/>
        <module name="javax.transaction.api"/>
    </dependencies>
</module>
```

Example Management CLI Commands

This example configuration can be achieved by using the following management CLI commands.

1. Add the Oracle JDBC driver as a core module.

```
module add --name=com.oracle --
resources=/path/to/misc/jdbc_drivers/oracle/ojdbc7.jar --
dependencies=javax.api,javax.transaction.api
```

IMPORTANT

Using the **`module`** management CLI command to add and remove modules is provided as technology preview only. This command is not appropriate for use in a managed domain or when connecting to the management CLI remotely. Modules should be added and removed manually in a production environment.

2. Register the Oracle JDBC driver.

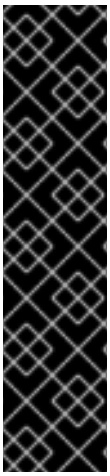
```
/subsystem=datasources/jdbc-driver=oracle:add(driver-
name=oracle,driver-module-name=com.oracle,driver-xa-datasource-
class-name=oracle.jdbc.xa.client.OracleXADataSource)
```

3. Add the Oracle datasource.

```
data-source add --name=OracleDS --jndi-name=java:jboss/OracleDS --
driver-name=oracle --connection-
url=jdbc:oracle:thin:@localhost:1521:XE --user-name=admin --
```

```
password=admin --validate-on-match=true --background-
validation=false --valid-connection-checker-class-
name=org.jboss.jca.adapters.jdbc.extensions.oracle.OracleValidConnec
tionChecker --exception-sorter-class-
name=org.jboss.jca.adapters.jdbc.extensions.oracle.OracleExceptionSo
rter --stale-connection-checker-class-
name=org.jboss.jca.adapters.jdbc.extensions.oracle.OracleStaleConnec
tionChecker
```

13.13.6. Example Oracle XA Datasource



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 EAP 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;**

This is an example of an Oracle XA datasource configuration with XA datasource properties, basic security, and validation options.

Example Oracle XA Datasource Configuration

```
<datasources>
  <xa-datasource jndi-name="java:jboss/OracleXADS" pool-name="OracleXADS">
    <xa-datasource-property name="URL">
      jdbc:oracle:thin:@oracleHostName:1521:orcl
    </xa-datasource-property>
    <driver>oracle</driver>
    <xa-pool>
      <is-same-rm-override>false</is-same-rm-override>
    </xa-pool>
    <security>
      <user-name>admin</user-name>
      <password>admin</password>
    </security>
    <validation>
      <valid-connection-checker class-
name="org.jboss.jca.adapters.jdbc.extensions.oracle.OracleValidConnectionC
hecker"/>
      <validate-on-match>true</validate-on-match>
      <background-validation>false</background-validation>
      <stale-connection-checker class-
name="org.jboss.jca.adapters.jdbc.extensions.oracle.OracleStaleConnectionC
hecker"/>
      <exception-sorter class-
name="org.jboss.jca.adapters.jdbc.extensions.oracle.OracleExceptionSorter"
/>
```

```
    </validation>
  </xa-datasource>
  <drivers>
    <driver name="oracle" module="com.oracle">
      <xa-datasource-class>oracle.jdbc.xa.client.OracleXADataSource</xa-
datasource-class>
    </driver>
  </drivers>
</datasources>
```

Example Oracle JDBC Driver `module.xml` File

```
<?xml version="1.0" ?>
<module xmlns="urn:jboss:module:1.1" name="com.oracle">
    <resources>
        <resource-root path="ojdbc7.jar"/>
    </resources>
    <dependencies>
        <module name="javax.api"/>
        <module name="javax.transaction.api"/>
    </dependencies>
</module>
```

Example Management CLI Commands

This example configuration can be achieved by using the following management CLI commands.

1. Add the Oracle JDBC driver as a core module.

```
module add --name=com.oracle --
resources=/path/to/misc/jdbc_drivers/oracle/ojdbc7.jar --
dependencies=javax.api,javax.transaction.api
```

IMPORTANT

Using the **`module`** management CLI command to add and remove modules is provided as technology preview only. This command is not appropriate for use in a managed domain or when connecting to the management CLI remotely. Modules should be added and removed manually in a production environment.

2. Register the Oracle JDBC driver.

```
/subsystem=datasources/jdbc-driver=oracle:add(driver-
name=oracle,driver-module-name=com.oracle,driver-xa-datasource-
class-name=oracle.jdbc.xa.client.OracleXADataSource)
```

3. Add the Oracle XA datasource.

```
xa-data-source add --name=OracleXADS --jndi-
name=java:jboss/OracleXADS --driver-name=oracle --user-name=admin --
password=admin --validate-on-match=true --background-
validation=false --valid-connection-checker-class-
name=org.jboss.jca.adapters.jdbc.extensions.oracle.OracleValidConnec
tionChecker --exception-sorter-class-
name=org.jboss.jca.adapters.jdbc.extensions.oracle.OracleExceptionSo
```

```
rter --stale-connection-checker-class-
name=org.jboss.jca.adapters.jdbc.extensions.oracle.OracleStaleConnec
tionChecker --same-rm-override=false --xa-datasource-properties=
{"URL"=>"jdbc:oracle:thin:@oracleHostName:1521:orcl"}
```

13.13.7. Example Microsoft SQL Server Datasource

This is an example of a Microsoft SQL Server datasource configuration with connection information, basic security, and validation options.

Example Microsoft SQL Server Datasource Configuration

```
<datasources>
  <datasource jndi-name="java:jboss/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>
      <valid-connection-checker class-
name="org.jboss.jca.adapters.jdbc.extensions.mssql.MSSQLValidConnectionChe
cker"/>
      <validate-on-match>true</validate-on-match>
      <background-validation>false</background-validation>
      <exception-sorter class-
name="org.jboss.jca.adapters.jdbc.extensions.mssql.MSSQLExceptionSorter"/>
    </validation>
  </datasource>
  <drivers>
    <driver name="sqlserver" module="com.microsoft">
      <xa-datasource-
class>com.microsoft.sqlserver.jdbc.SQLServerXADataSource</xa-datasource-
class>
    </driver>
  </drivers>
</datasources>
```

Example Microsoft SQL Server JDBC Driver `module.xml` File

```
<?xml version="1.0" ?>
<module xmlns="urn:jboss:module:1.1" name="com.microsoft">
  <resources>
    <resource-root path="sqljdbc41.jar"/>
  </resources>
  <dependencies>
    <module name="javax.api"/>
    <module name="javax.transaction.api"/>
  </dependencies>
</module>
```

Example Management CLI Commands

This example configuration can be achieved by using the following management CLI commands.

1. Add the Microsoft SQL Server JDBC driver as a core module.

```
module add --name=com.microsoft --resources=/path/to/sqljdbc41.jar --dependencies=javax.api,javax.transaction.api
```

IMPORTANT

Using the `module` management CLI command to add and remove modules is provided as technology preview only. This command is not appropriate for use in a managed domain or when connecting to the management CLI remotely. Modules should be added and removed manually in a production environment.

2. Register the Microsoft SQL Server JDBC driver.

```
/subsystem=datasources/jdbc-driver=sqlserver:add(driver-name=sqlserver,driver-module-name=com.microsoft,driver-xa-datasource-class-name=com.microsoft.sqlserver.jdbc.SQLServerXADataSource)
```

3. Add the Microsoft SQL Server datasource.

```
data-source add --name=MSSQLDS --jndi-name=java:jboss/MSSQLDS --driver-name=sqlserver --connection-url=jdbc:sqlserver://localhost:1433;DatabaseName=MyDatabase --user-name=admin --password=admin --validate-on-match=true --background-validation=false --valid-connection-checker-class-name=org.jboss.jca.adapters.jdbc.extensions.mssql.MSSQLValidConnectionChecker --exception-sorter-class-name=org.jboss.jca.adapters.jdbc.extensions.mssql.MSSQLExceptionSorter
```

13.13.8. Example Microsoft SQL Server XA Datasource

This is an example of a Microsoft SQL Server XA datasource configuration with XA datasource properties, basic security, and validation options.

Example Microsoft SQL Server XA Datasource Configuration

```
<datasources>
  <xa-datasource jndi-name="java:jboss/MSSQLXADS" pool-name="MSSQLXADS">
    <xa-datasource-property name="ServerName">
      localhost
    </xa-datasource-property>
    <xa-datasource-property name="DatabaseName">
      mssqldb
    </xa-datasource-property>
    <xa-datasource-property name="SelectMethod">
      cursor
    </xa-datasource-property>
    <driver>sqlserver</driver>
    <xa-pool>
      <is-same-rm-override>false</is-same-rm-override>
```

```
    </xa-pool>
    <security>
      <user-name>admin</user-name>
      <password>admin</password>
    </security>
    <validation>
      <valid-connection-checker class-
name="org.jboss.jca.adapters.jdbc.extensions.mssql.MSSQLValidConnectionChe
cker"/>
      <validate-on-match>true</validate-on-match>
      <background-validation>false</background-validation>
      <exception-sorter class-
name="org.jboss.jca.adapters.jdbc.extensions.mssql.MSSQLExceptionSorter"/>
    </validation>
  </xa-datasource>
  <drivers>
    <driver name="sqlserver" module="com.microsoft">
      <xa-datasource-
class>com.microsoft.sqlserver.jdbc.SQLServerXADataSource</xa-datasource-
class>
    </driver>
  </drivers>
</datasources>
```

Example Microsoft SQL Server JDBC Driver `module.xml` File

```
<?xml version="1.0" ?>
<module xmlns="urn:jboss:module:1.1" name="com.microsoft">
  <resources>
    <resource-root path="sqljdbc41.jar"/>
  </resources>
  <dependencies>
    <module name="javax.api"/>
    <module name="javax.transaction.api"/>
  </dependencies>
</module>
```

Example Management CLI Commands
This example configuration can be achieved by using the following management CLI commands.

1. Add the Microsoft SQL Server JDBC driver as a core module.

```
module add --name=com.microsoft --resources=/path/to/sqljdbc41.jar --
-dependencies=javax.api,javax.transaction.api
```

IMPORTANT

Using the **`module`** management CLI command to add and remove modules is provided as technology preview only. This command is not appropriate for use in a managed domain or when connecting to the management CLI remotely. Modules should be added and removed manually in a production environment.

2. Register the Microsoft SQL Server JDBC driver.

```
/subsystem=datasources/jdbc-driver=sqlserver:add(driver-
name=sqlserver,driver-module-name=com.microsoft,driver-xa-
datasource-class-
name=com.microsoft.sqlserver.jdbc.SQLServerXADataSource)
```

3. Add the Microsoft SQL Server XA datasource.

```
xa-data-source add --name=MSSQLXADS --jndi-name=java:jboss/MSSQLXADS
--driver-name=sqlserver --user-name=admin --password=admin --
validate-on-match=true --background-validation=false --background-
validation=false --valid-connection-checker-class-
name=org.jboss.jca.adapters.jdbc.extensions.mssql.MSSQLValidConnecti
onChecker --exception-sorter-class-
name=org.jboss.jca.adapters.jdbc.extensions.mssql.MSSQLExceptionSort
er --same-rm-override=false --xa-datasource-properties=
{"ServerName"=>"localhost","DatabaseName"=>"mssqldb","SelectMethod"=
>"cursor"}
```

13.13.9. Example IBM DB2 Datasource

This is an example of an IBM DB2 datasource configuration with connection information, basic security, and validation options.

Example IBM DB2 Datasource Configuration

```
<datasources>
  <datasource jndi-name="java:jboss/DB2DS" pool-name="DB2DS">
    <connection-url>jdbc:db2://localhost:50000/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>
      <valid-connection-checker class-
name="org.jboss.jca.adapters.jdbc.extensions.db2.DB2ValidConnectionChecker
"/>
      <validate-on-match>true</validate-on-match>
      <background-validation>false</background-validation>
      <stale-connection-checker class-
name="org.jboss.jca.adapters.jdbc.extensions.db2.DB2StaleConnectionChecker
"/>
      <exception-sorter class-
name="org.jboss.jca.adapters.jdbc.extensions.db2.DB2ExceptionSorter"/>
    </validation>
  </datasource>
  <drivers>
    <driver name="ibmdb2" module="com.ibm">
        <xa-datasource-class>com.ibm.db2.jcc.DB2XADataSource</xa-
datasource-class>
```

```
    </driver>
  </drivers>
</datasources>
```

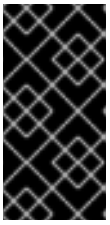
Example IBM DB2 JDBC Driver `module.xml` File

```
<?xml version="1.0" ?>
<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>
```

Example Management CLI Commands
This example configuration can be achieved by using the following management CLI commands.

1. Add the IBM DB2 JDBC driver as a core module.

```
module add --name=com.ibm --resources=/path/to/db2jcc4.jar --
dependencies=javax.api,javax.transaction.api
```



IMPORTANT

Using the **`module`** management CLI command to add and remove modules is provided as technology preview only. This command is not appropriate for use in a managed domain or when connecting to the management CLI remotely. Modules should be added and removed manually in a production environment.

2. Register the IBM DB2 JDBC driver.

```
/subsystem=datasources/jdbc-driver=ibmdb2:add(driver-
name=ibmdb2,driver-module-name=com.ibm,driver-xa-datasource-class-
name=com.ibm.db2.jcc.DB2XADataSource)
```

3. Add the IBM DB2 datasource.

```
data-source add --name=DB2DS --jndi-name=java:jboss/DB2DS --driver-
name=ibmdb2 --connection-url=jdbc:db2://localhost:50000/ibmdb2db --
user-name=admin --password=admin --validate-on-match=true --
background-validation=false --valid-connection-checker-class-
name=org.jboss.jca.adapters.jdbc.extensions.db2.DB2ValidConnectionCh
ecker --exception-sorter-class-
name=org.jboss.jca.adapters.jdbc.extensions.db2.DB2ExceptionSorter -
-stale-connection-checker-class-
name=org.jboss.jca.adapters.jdbc.extensions.db2.DB2StaleConnectionCh
ecker --min-pool-size=0 --max-pool-size=50
```

13.13.10. Example IBM DB2 XA Datasource

This is an example of an IBM DB2 XA datasource configuration with XA datasource properties, basic security, and validation options.

Example IBM DB2 XA Datasource Configuration

```
<datasources>
  <xa-datasource jndi-name="java:jboss/DB2XADS" pool-name="DB2XADS">
    <xa-datasource-property name="ServerName">
      localhost
    </xa-datasource-property>
    <xa-datasource-property name="DatabaseName">
      ibmdb2db
    </xa-datasource-property>
    <xa-datasource-property name="PortNumber">
      50000
    </xa-datasource-property>
    <xa-datasource-property name="DriverType">
      4
    </xa-datasource-property>
    <driver>ibmdb2</driver>
    <xa-pool>
      <is-same-rm-override>false</is-same-rm-override>
    </xa-pool>
    <security>
      <user-name>admin</user-name>
      <password>admin</password>
    </security>
    <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>
    <validation>
      <valid-connection-checker class-
name="org.jboss.jca.adapters.jdbc.extensions.db2.DB2ValidConnectionChecker
"/>
      <validate-on-match>true</validate-on-match>
      <background-validation>false</background-validation>
      <stale-connection-checker class-
name="org.jboss.jca.adapters.jdbc.extensions.db2.DB2StaleConnectionChecker
"/>
      <exception-sorter class-
name="org.jboss.jca.adapters.jdbc.extensions.db2.DB2ExceptionSorter"/>
    </validation>
  </xa-datasource>
  <drivers>
    <driver name="ibmdb2" module="com.ibm">
```

```
        <xa-datasource-class>com.ibm.db2.jcc.DB2XADataSource</xa-
datasource-class>
    </driver>
  </drivers>
</datasources>
```

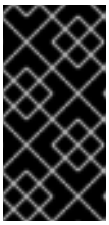
Example IBM DB2 JDBC Driver `module.xml` File

```
<?xml version="1.0" ?>
<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>
```

Example Management CLI Commands
This example configuration can be achieved by using the following management CLI commands.

1. Add the IBM DB2 JDBC driver as a core module.

```
module add --name=com.ibm --resources=/path/to/db2jcc4.jar --
dependencies=javax.api,javax.transaction.api
```



IMPORTANT

Using the **`module`** management CLI command to add and remove modules is provided as technology preview only. This command is not appropriate for use in a managed domain or when connecting to the management CLI remotely. Modules should be added and removed manually in a production environment.

2. Register the IBM DB2 JDBC driver.

```
/subsystem=datasources/jdbc-driver=ibmdb2:add(driver-
name=ibmdb2,driver-module-name=com.ibm,driver-xa-datasource-class-
name=com.ibm.db2.jcc.DB2XADataSource)
```

3. Add the IBM DB2 XA datasource.

```
xa-data-source add --name=DB2XADS --jndi-name=java:jboss/DB2XADS --
driver-name=ibmdb2 --user-name=admin --password=admin --validate-on-
match=true --background-validation=false --background-
validation=false --valid-connection-checker-class-
name=org.jboss.jca.adapters.jdbc.extensions.db2.DB2ValidConnectionCh
ecker --exception-sorter-class-
name=org.jboss.jca.adapters.jdbc.extensions.db2.DB2ExceptionSorter -
-stale-connection-checker-class-
name=org.jboss.jca.adapters.jdbc.extensions.db2.DB2StaleConnectionCh
ecker --same-rm-override=false --recovery-plugin-class-
name=org.jboss.jca.core.recovery.ConfigurableRecoveryPlugin --
recovery-plugin-properties=
```

```
{"EnableIsValid"=>"false","IsValidOverride"=>"false","EnableClose"=>
"false"} --xa-datasource-properties=
{"ServerName"=>"localhost","DatabaseName"=>"ibmdb2db","PortNumber"=>
"50000","DriverType"=>"4"}
```

13.13.11. Example Sybase Datasource

This is an example of a Sybase datasource configuration with connection information, basic security, and validation options.

Example Sybase Datasource Configuration

```
<datasources>
  <datasource jndi-name="java:jboss/SybaseDB" pool-name="SybaseDB">
    <connection-url>jdbc:sybase:Tds:localhost:5000/DATABASE?
JCONNECT_VERSION=6</connection-url>
    <driver>sybase</driver>
    <security>
      <user-name>admin</user-name>
      <password>admin</password>
    </security>
    <validation>
      <valid-connection-checker class-
name="org.jboss.jca.adapters.jdbc.extensions.sybase.SybaseValidConnectionC
hecker"/>
      <validate-on-match>true</validate-on-match>
      <background-validation>false</background-validation>
      <exception-sorter class-
name="org.jboss.jca.adapters.jdbc.extensions.sybase.SybaseExceptionSorter"
/>
    </validation>
  </datasource>
  <drivers>
    <driver name="sybase" module="com.sybase">
      <xa-datasource-class>com.sybase.jdbc4.jdbc.SybXADataSource</xa-
datasource-class>
    </driver>
  </drivers>
</datasources>
```

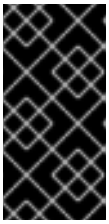
Example Sybase JDBC Driver `module.xml` File

```
<?xml version="1.0" ?>
<module xmlns="urn:jboss:module:1.1" name="com.sybase">
    <resources>
      <resource-root path="jconn4.jar"/>
    </resources>
    <dependencies>
      <module name="javax.api"/>
      <module name="javax.transaction.api"/>
    </dependencies>
</module>
```

Example Management CLI Commands
This example configuration can be achieved by using the following management CLI commands.

1. Add the Sybase JDBC driver as a core module.

```
module add --name=com.sybase --resources=/path/to/jconn4.jar --
dependencies=javax.api,javax.transaction.api
```



IMPORTANT

Using the **module** management CLI command to add and remove modules is provided as technology preview only. This command is not appropriate for use in a managed domain or when connecting to the management CLI remotely. Modules should be added and removed manually in a production environment.

2. Register the Sybase JDBC driver.

```
/subsystem=datasources/jdbc-driver=sybase:add(driver-
name=sybase,driver-module-name=com.sybase,driver-xa-datasource-
class-name=com.sybase.jdbc4.jdbc.SybXADataSource)
```

3. Add the Sybase datasource.

```
data-source add --name=SybaseDB --jndi-name=java:jboss/SybaseDB --
driver-name=sybase --connection-
url=jdbc:sybase:Tds:localhost:5000/DATABASE?JCONNECT_VERSION=6 --
user-name=admin --password=admin --validate-on-match=true --
background-validation=false --valid-connection-checker-class-
name=org.jboss.jca.adapters.jdbc.extensions.sybase.SybaseValidConnec
tionChecker --exception-sorter-class-
name=org.jboss.jca.adapters.jdbc.extensions.sybase.SybaseExceptionSo
rter
```

13.13.12. Example Sybase XA Datasource

This is an example of a Sybase XA datasource configuration with XA datasource properties, basic security, and validation options.

Example Sybase XA Datasource Configuration

```
<datasources>
  <xa-datasource jndi-name="java:jboss/SybaseXADS" pool-name="SybaseXADS">
    <xa-datasource-property name="ServerName">
      localhost
    </xa-datasource-property>
    <xa-datasource-property name="DatabaseName">
      mydatabase
    </xa-datasource-property>
    <xa-datasource-property name="PortNumber">
      4100
    </xa-datasource-property>
    <xa-datasource-property name="NetworkProtocol">
      Tds
    </xa-datasource-property>
    <driver>sybase</driver>
    <xa-pool>
      <is-same-rm-override>false</is-same-rm-override>
```

```
    </xa-pool>
    <security>
      <user-name>admin</user-name>
      <password>admin</password>
    </security>
    <validation>
      <valid-connection-checker class-
name="org.jboss.jca.adapters.jdbc.extensions.sybase.SybaseValidConnectionC
hecker"/>
      <validate-on-match>true</validate-on-match>
      <background-validation>false</background-validation>
      <exception-sorter class-
name="org.jboss.jca.adapters.jdbc.extensions.sybase.SybaseExceptionSorter"
/>
    </validation>
  </xa-datasource>
  <drivers>
    <driver name="sybase" module="com.sybase">
      <xa-datasource-class>com.sybase.jdbc4.jdbc.SybXADataSource</xa-
datasource-class>
    </driver>
  </drivers>
</datasources>
```

Example Sybase JDBC Driver `module.xml` File

```
<?xml version="1.0" ?>
<module xmlns="urn:jboss:module:1.1" name="com.sybase">
    <resources>
      <resource-root path="jconn4.jar"/>
    </resources>
    <dependencies>
      <module name="javax.api"/>
      <module name="javax.transaction.api"/>
    </dependencies>
</module>
```

Example Management CLI Commands
This example configuration can be achieved by using the following management CLI commands.

1. Add the Sybase JDBC driver as a core module.

```
module add --name=com.sybase --resources=/path/to/jconn4.jar --
dependencies=javax.api,javax.transaction.api
```

IMPORTANT

Using the **`module`** management CLI command to add and remove modules is provided as technology preview only. This command is not appropriate for use in a managed domain or when connecting to the management CLI remotely. Modules should be added and removed manually in a production environment.

2. Register the Sybase JDBC driver.

```
/subsystem=datasources/jdbc-driver=sybase:add(driver-
name=sybase,driver-module-name=com.sybase,driver-xa-datasource-
class-name=com.sybase.jdbc4.jdbc.SybXADataSource)
```

3. Add the Sybase XA datasource.

```
xa-data-source add --name=SybaseXADS --jndi-
name=java:jboss/SybaseXADS --driver-name=sybase --user-name=admin --
password=admin --validate-on-match=true --background-
validation=false --background-validation=false --valid-connection-
checker-class-
name=org.jboss.jca.adapters.jdbc.extensions.sybase.SybaseValidConnec
tionChecker --exception-sorter-class-
name=org.jboss.jca.adapters.jdbc.extensions.sybase.SybaseExceptionSo
rter --same-rm-override=false --xa-datasource-properties=
{"ServerName"=>"localhost","DatabaseName"=>"mydatabase","PortNumber"
=>"4100","NetworkProtocol"=>"Tds"}
```

13.13.13. Example MariaDB Datasource

This is an example of a MariaDB datasource configuration with connection information, basic security, and validation options.

Example MariaDB Datasource Configuration

```
<datasources>
  <datasource jndi-name="java:jboss/MariaDBDS" pool-name="MariaDBDS">
    <connection-url>jdbc:mariadb://localhost:3306/jbossdb</connection-url>
    <driver>mariadb</driver>
    <security>
      <user-name>admin</user-name>
      <password>admin</password>
    </security>
    <validation>
      <valid-connection-checker class-
name="org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLValidConnectionChe
cker"/>
      <validate-on-match>true</validate-on-match>
      <background-validation>false</background-validation>
      <exception-sorter class-
name="org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLExceptionSorter"/>
    </validation>
  </datasource>
  <drivers>
    <driver name="mariadb" module="org.mariadb">
      <driver-class>org.mariadb.jdbc.Driver</driver-class>
      <xa-datasource-class>org.mariadb.jdbc.MySQLDataSource</xa-
datasource-class>
    </driver>
  </drivers>
</datasources>
```

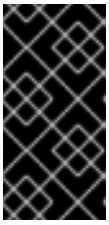
Example MariaDB JDBC Driver `module.xml` File

```
<?xml version="1.0" ?>
<module xmlns="urn:jboss:module:1.1" name="org.mariadb">
  <resources>
    <resource-root path="mariadb-java-client-1.2.3.jar"/>
  </resources>
  <dependencies>
    <module name="javax.api"/>
    <module name="javax.transaction.api"/>
  </dependencies>
</module>
```

Example Management CLI Commands
This example configuration can be achieved by using the following management CLI commands.

1. Add the MariaDB JDBC driver as a core module.

```
module add --name=org.mariadb --resources=/path/to/mariadb-java-
client-1.2.3.jar --dependencies=javax.api,javax.transaction.api
```



IMPORTANT

Using the **module** management CLI command to add and remove modules is provided as technology preview only. This command is not appropriate for use in a managed domain or when connecting to the management CLI remotely. Modules should be added and removed manually in a production environment.

2. Register the MariaDB JDBC driver.

```
/subsystem=datasources/jdbc-driver=mariadb:add(driver-
name=mariadb,driver-module-name=org.mariadb,driver-xa-datasource-
class-name=org.mariadb.jdbc.MySQLDataSource, driver-class-
name=org.mariadb.jdbc.Driver)
```

3. Add the MariaDB datasource.

```
data-source add --name=MariaDBDS --jndi-name=java:jboss/MariaDBDS --
driver-name=mariadb --connection-
url=jdbc:mariadb://localhost:3306/jbossdb --user-name=admin --
password=admin --validate-on-match=true --background-
validation=false --valid-connection-checker-class-
name=org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLValidConnecti
onChecker --exception-sorter-class-
name=org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLExceptionSort
er
```

13.13.14. Example MariaDB XA Datasource

This is an example of a MariaDB XA datasource configuration with XA datasource properties, basic security, and validation options.

Example MariaDB XA Datasource Configuration

```
<datasources>
```

```
  <xa-datasource jndi-name="java:jboss/MariaDBXADS" pool-
name="MariaDBXADS">
    <xa-datasource-property name="ServerName">
      localhost
    </xa-datasource-property>
    <xa-datasource-property name="DatabaseName">
      mariadbdb
    </xa-datasource-property>
    <driver>mariadb</driver>
    <security>
      <user-name>admin</user-name>
      <password>admin</password>
    </security>
    <validation>
      <valid-connection-checker class-
name="org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLValidConnectionChe
cker"/>
      <validate-on-match>true</validate-on-match>
      <background-validation>false</background-validation>
      <exception-sorter class-
name="org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLExceptionSorter"/>
    </validation>
  </xa-datasource>
  <drivers>
    <driver name="mariadb" module="org.mariadb">
      <driver-class>org.mariadb.jdbc.Driver</driver-class>
      <xa-datasource-class>org.mariadb.jdbc.MySQLDataSource</xa-
datasource-class>
    </driver>
  </drivers>
</datasources>
```

Example MariaDB JDBC Driver `module.xml` File

```
<?xml version="1.0" ?>
<module xmlns="urn:jboss:module:1.1" name="org.mariadb">
  <resources>
    <resource-root path="mariadb-java-client-1.2.3.jar"/>
  </resources>
  <dependencies>
    <module name="javax.api"/>
    <module name="javax.transaction.api"/>
  </dependencies>
</module>
```

Example Management CLI Commands
This example configuration can be achieved by using the following management CLI commands.

1. Add the MariaDB JDBC driver as a core module.

```
module add --name=org.mariadb --resources=/path/to/mariadb-java-
client-1.2.3.jar --dependencies=javax.api,javax.transaction.api
```

IMPORTANT

Using the **module** management CLI command to add and remove modules is provided as technology preview only. This command is not appropriate for use in a managed domain or when connecting to the management CLI remotely. Modules should be added and removed manually in a production environment.

2. Register the MariaDB JDBC driver.

```
/subsystem=datasources/jdbc-driver=mariadb:add(driver-
name=mariadb,driver-module-name=org.mariadb,driver-xa-datasource-
class-name=org.mariadb.jdbc.MySQLDataSource, driver-class-
name=org.mariadb.jdbc.Driver)
```

3. Add the MariaDB XA datasource.

```
xa-data-source add --name=MariaDBXADS --jndi-
name=java:jboss/MariaDBXADS --driver-name=mariadb --user-name=admin
--password=admin --validate-on-match=true --background-
validation=false --valid-connection-checker-class-
name=org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLValidConnecti
onChecker --exception-sorter-class-
name=org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLExceptionSort
er --xa-datasource-properties=
{"ServerName"=>"localhost","DatabaseName"=>"mariadbdb"}
```

CHAPTER 14. CONFIGURING TRANSACTIONS

14.1. TRANSACTIONS SUBSYSTEM CONFIGURATION

14.1.1. Configuring the Transaction Manager

You can configure the transaction manager using the web-based management console or the command line management CLI.

Configuring the Transaction Manager Using the Management Console
The following steps explain how to configure the transaction manager using the web-based management console:

1. Select the **Configuration** tab from the top of the screen.
2. If you are running JBoss EAP as a managed domain, choose the desired profile to modify.
3. From the **Subsystem** list, select **Transactions** and click **View**.
4. Click **Edit** on the appropriate tab for the settings that you want to edit, such as **Recovery** for recovery options.
5. Make the necessary changes and click **Save** to save the changes.
6. Click **Need Help?** to display help text.

Configuring the Transaction Manager Using the Management CLI
Using the management CLI, you can configure the transaction manager using a series of commands. The commands all begin with `/subsystem=transactions` for a standalone server or `/profile=default/subsystem=transactions/` for the `default` profile in a managed domain.

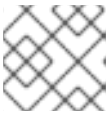
For a detailed listing of all the transaction manager configuration options, see the Transaction Manager Configuration Options for JBoss EAP.

14.1.2. Configuring Your Datasource to Use JTA

This task shows you how to enable Java Transaction API (JTA) on your datasource.

Prerequisites

- Your database must support Java Transaction API. If in doubt, consult the documentation for your database.
- Create a non-XA datasource.



NOTE

XA datasources are already JTA capable by default.

Configuring the Datasource to use Java Transaction API

1. Use the following management CLI command to set the `jta` attribute to `true`.

```
/subsystem=datasources/data-source=DATASOURCE_NAME:write-
attribute(name=jta,value=true)
```

NOTE

In a managed domain, precede this command with **`/profile=PROFILE_NAME`**.

2. Reload the server for the changes to take effect.

```
reload
```

Your datasource is now configured to use JTA.

14.1.3. About Transaction Log Messages

You can track the transaction status while keeping the log files readable by using the **DEBUG** log level for the transaction logger. For detailed debugging, use the **TRACE** log level. Refer to Configuring Logging for the Transactions Subsystem for information on configuring the transaction logger.

Transaction Manager (TM) 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 messages other than these.

Table 14.1. Transaction State Change

Transaction Begin	When a transaction begins, a method **`Begin`** of class **`com.arjuna.ats.arjuna.coordinator.BasicAction`** is executed and presented in the log with the message **`BasicAction::Begin() for action-id <transaction uid>`**.
Transaction Commit	When a transaction commits, a method **`Commit`** of class **`com.arjuna.ats.arjuna.coordinator.BasicAction`** is executed and presented in the log with the message **`BasicAction::Commit() for action-id <transaction uid>`**.
Transaction Rollback	When a transaction rolls back, a method **`Rollback`** of class **`com.arjuna.ats.arjuna.coordinator.BasicAction`** is executed and presented in the log with the message **`BasicAction::Rollback() for action-id <transaction uid>`**.
Transaction Timeout	When a transaction times out, a method **`doCancellations`** of **`com.arjuna.ats.arjuna.coordinator.TransactionReaper`** is executed and presented in log as **`Reaper Worker <thread id> attempting to cancel <transaction uid>`**. You will then see the same thread rolling back the transaction as shown above.

14.1.4. Configuring Logging for the Transactions Subsystem

You can control the amount of information logged about transactions, independent of other logging settings in JBoss EAP. You can configure the logging settings using the management console or the management CLI.

Configuring the Transaction Logger Using the Management Console

1. Navigate to the **Logging** subsystem configuration.

    a. In the management console, click the **Configuration** tab. If you use a managed domain, you must first choose the appropriate server profile.

    b. Select the **Logging** subsystem and click **View**.

2. Edit the `com.arjuna` attributes.
   Select the **Log Categories** tab. The `com.arjuna` entry is already present. Select `com.arjuna` and click **Edit** in the **Attributes** section. You can change the log level and choose whether to use parent handlers or not.

    - Log Level:
      As transactions can produce a lot of logging output, the default logging level is set to **WARN** so that the server log is not overwhelmed by transaction output. If you need to check transaction processing details, use the **TRACE** log level so that transaction IDs are shown.

    - Use Parent Handlers:
      Parent handler indicates whether the logger should send its output to its parent logger. The default behavior is `true`.

3. Click **Save** to save the changes.

Configuring the Transaction Logger Using the Management CLI
Use the following command to set the logging level from the management CLI. For a standalone server, remove the `/profile=default` from the command.

```
/profile=default/subsystem=logging/logger=com.arjuna:write-
attribute(name=level,value=VALUE)
```

14.2. TRANSACTION ADMINISTRATION

14.2.1. 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 TM and the management API of JBoss EAP.

The TM 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`. The `probe` operation reads the transaction logs and creates a node path for each record. You can call the `probe` operation manually, whenever you need to refresh the `log-store`. It is normal for transaction logs to appear and disappear quickly.

Refresh the Log Store
The following 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
```

View All Prepared Transactions
To view all prepared transactions, first refresh the log store (see Refresh the Log Store), then run the following command, which functions similarly to a file system `ls` command.

```
ls /profile=default/subsystem=transactions/log-store=log-
store/transactions
```

Or

```
/host=master/server=server-one/subsystem=transactions/log-store=log-
store:read-children-names(child-type=transactions)
```

Each transaction is shown, along with its unique identifier. Individual operations can be run against an individual transaction (see Manage a Transaction).

14.2.1.1. Manage a Transaction

View the Attributes of a Transaction
To view information about a transaction, such as its JNDI name, EIS product name and version, or its status, use the `read-resource` operation.

```
/profile=default/subsystem=transactions/log-store=log-
store/transactions=0\:ffff7f000001\:-b66efc2\:4f9e6f8f\:9:read-resource
```

View the Details of a Transaction Participant
Each transaction log contains a child element called `participants`. Use the `read-resource` operation on this element to see the details of a participant 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" => "ActiveMQ",
       "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 Participant 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)
```

You can use this alternate command, which shows participant IDs of transaction in an aggregated form.

```
/host=master/server=server-one/subsystem=transactions/log-store=log-
store/transactions=0\:ffff7f000001\:-b66efc2\:4f9e6f8f\:9:read-children-
names(child-type=participants)
```

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\:ffff7f000001\:-b66efc2\:4f9e6f8f\:9:delete
```

This deletes all participants in the transaction as well.

Recover a Transaction Participant
Each transaction participant supports recovery by using the **`recover`** operation.

```
/profile=default/subsystem=transactions/log-store=log-
store/transactions=0\:ffff7f000001\:-
b66efc2\:4f9e6f8f\:9/participants=2:recover
```

If the transaction participant's status is **HEURISTIC**, the **`recover`** operation switches the status to **PREPARE** and asks the periodic recovery process to replay the commit.

If the commit is successful, the participant is removed from the transaction log. You can verify this by 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 Participant
If a transaction needs recovery, you can use the **`refresh`** operation to be sure it still requires recovery, before attempting the recovery.

```
/profile=default/subsystem=transactions/log-store=log-
store/transactions=0\:ffff7f000001\:-
b66efc2\:4f9e6f8f\:9/participants=2:refresh
```

14.2.2. View Transaction Statistics

If transaction manager statistics are enabled, you can view statistics on processed transactions by the transaction manager. See Configuring the Transaction Manager for information about how to enable transaction manager statistics.

You can view statistics using either the management console or the management CLI. In the management console, transaction statistics are available by navigating to the **Transactions** subsystem from the **Runtime** tab. From the management CLI, you can view statistics by using **`include-runtime=true`** to the **`read-resource`** operation. For example:

```
/subsystem=transactions:read-resource(include-runtime=true)
```

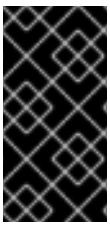
The following table shows each available statistic and its description.

Table 14.2. Transactions Subsystem Statistics

Statistic	Description
number-of-transactions	The total number of transactions processed by the transaction manager on this server.
number-of-committed-transactions	The number of committed transactions processed by the transaction manager on this server.
number-of-aborted-transactions	The number of aborted transactions processed by the transaction manager on this server.
number-of-timed-out-transactions	The number of timed out transactions processed by the transaction manager on this server. The number of timed out transactions is calculated to number of aborted transactions too.
number-of-heuristics	Number of transactions in a heuristic state.
number-of-inflight-transactions	Number of transactions which have begun but not yet terminated.
number-of-application-rollbacks	The number of failed transactions whose failure origin was an application.
number-of-resource-rollbacks	The number of failed transactions whose failure origin was a resource.

14.2.3. Transactions Object Store

Transactions need a place to store objects. One of the options for object storage is a JDBC datasource. If performance is a particular concern, the JDBC object store can be slower than a file system or ActiveMQ journal object store.



IMPORTANT

If the `transactions` subsystem is configured to use Apache ActiveMQ Artemis journal as storage type for transaction logs, then two instances of JBoss EAP are not permitted to use the same directory for storing the journal. Application server instances can not share the same location and each has to configure a unique location for it.

NOTE

Losing a transaction object store can lead to losing data consistency. Thus, the object store needs to be placed on a *safe* drive.

Use a JDBC Datasource as a Transactions Object Store
Follow the below steps to use a JDBC datasource as a transactions object store.

1. Create a datasource, for example, **TransDS**. For instructions, see Create a Non-XA datasource . Note that the datasource's JDBC driver must be installed as a core module , not as a JAR deployment, for the object store to work properly.
2. Set the datasource's **jta** attribute to **false**.

```
/subsystem=datasources/data-source=TransDS:write-attribute(name=jta,
value=false)
```

3. Set the **jdbc-store-datasource** attribute to the JNDI name for the datasource to use, for example, **java:jboss/datasources/TransDS**.

```
/subsystem=transactions:write-attribute(name=jdbc-store-datasource,
value=java:jboss/datasources/TransDS)
```

4. Set the **use-jdbc-store** attribute to **true**.

```
/subsystem=transactions:write-attribute(name=use-jdbc-store,
value=true)
```

5. Restart the JBoss EAP server for the changes to take effect.

Transactions JDBC Store Attributes
The following table identifies all of the available attributes related to JDBC object storage.

Table 14.3. Transactions JDBC Store Attributes

Property	Description
use-jdbc-store	Set this to **true** to enable the JDBC store for transactions.
jdbc-store-datasource	The JNDI name of the JDBC datasource used for storage.
jdbc-action-store-drop-table	Whether to drop and recreate the action store tables at launch. The default is **false**.
jdbc-action-store-table-prefix	The prefix for the action store table names.
jdbc-communication-store-drop-table	Whether to drop and recreate the communication store tables at launch. The default is **false**.
jdbc-communication-store-table-prefix	The prefix for the communication store table names.
jdbc-state-store-drop-table	Whether to drop and recreate the state store tables at launch. The default is **false**.
jdbc-state-store-table-prefix	The prefix for the state store table names.

Use the ActiveMQ Journal Object Store
Follow the below steps to use an ActiveMQ journal object store.

1. Set the **use-journal-store** attribute to **true**.

```
/subsystem=transactions:write-attribute(name=use-journal-
store,value=true)
```

2. Restart the JBoss EAP server for the changes to take effect.

CHAPTER 15. ORB CONFIGURATION

15.1. 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 provides an ORB instance, by means of the Open JDK ORB component.

The ORB is used internally for Java Transaction Service (JTS) transactions, and is also available for use by your own applications.

15.2. CONFIGURE THE ORB FOR JTS TRANSACTIONS

In a default installation of JBoss EAP, the ORB support for transactions is disabled. You can configure ORB settings in the **`iiop-openjdk`** subsystem using the management CLI or the management console.

NOTE

The **`iiop-openjdk`** subsystem is available when using the *full* or *full-ha* profile in a managed domain, or the **`standalone-full.xml`** or **`standalone-full-ha.xml`** configuration file for a standalone server.

For a listing of the available configuration options for the **`iiop-openjdk`** subsystem, see IIOP Subsystem Attributes.

Configure the ORB Using the Management CLI

You can configure each aspect of the ORB using the management CLI. This is the minimum configuration for the ORB to be used with JTS.

The following management CLI 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 are using a standalone server, omit the **`/profile=full`** portion of the commands.

Enable the Security Interceptors

Enable the **`security`** attribute by setting the value to **`identity`**.

```
/profile=full/subsystem=iiop-openjdk:write-
attribute(name=security,value=identity)
```

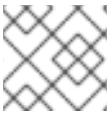
Enable Transactions in the IIOP Subsystem

To enable the ORB for JTS, set the value of **`transactions`** attribute to **`full`**, rather than the default **`spec`**.

```
/profile=full/subsystem=iiop-openjdk:write-attribute(name=transactions,
value=full)
```

Enable JTS in the Transactions 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.

Configure the ORB Using the Management Console

1. Select the **Configuration** tab from the top of the management console.
2. Select **Subsystems**. In a managed domain, you will need to select the appropriate profile first.
3. Select the **IIOP** subsystem and click **View**.
4. Click the **Edit** button and modify the attributes as needed. Click on the **Need Help?** link for detailed explanations of each field.
5. Click **Save** to save the changes.

CHAPTER 16. JAVA CONNECTOR ARCHITECTURE (JCA) MANAGEMENT

16.1. ABOUT THE 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. A resource adapter is a component that implements the Java EE Connector API architecture.

JCA 1.7 provides features for managing:

- connections
- transactions
- security
- life-cycle
- work instances
- transaction inflow
- message inflow

JCA 1.7 was developed under the Java Community Process as JSR-322.

16.2. ABOUT 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. A RAR file may also be included in an Enterprise Archive (EAR) deployment.

16.3. CONFIGURING THE JCA SUBSYSTEM

The `jca` subsystem controls the general settings for the JCA container and resource adapter deployments. You can configure the `jca` subsystem using the management console or the management CLI.

The main JCA elements to configure are:

- Archive validation
- Bean validation

- Work managers
- Bootstrap contexts
- Cached connection manager

Configure JCA settings from the management console

The `jca` subsystem can be configured from the management console by navigating to **Configuration** → **Subsystems** → **JCA**. Then, select the appropriate tab:

- **Common Config** - Contains settings for the cached connection manager, archive validation, and bean validation. Each of these is contained in their own tab as well. Modify these settings by opening the appropriate tab and clicking the **Edit** button.
- **Bootstrap Contexts** - 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.
- **Work Manager** - 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.
  The thread pool attributes can be configured by clicking **View** on the selected work manager.

Configure JCA settings from the management CLI

The `jca` subsystem can be configured from the management CLI from the `/subsystem=jca` address. In a managed domain, you must precede the command with `/profile=PROFILE_NAME`.

Archive Validation

This determines whether archive validation will be performed on the deployment units. The following table describes the attributes you can set for archive validation.

Table 16.1. Archive Validation Attributes

Attribute	Default Value	Description
enabled	true	Specifies whether archive validation is enabled.
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 JCA 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 16.2. 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 16.3. Work Manager Attributes

Attribute	Description
name	Specifies the name of the work manager.
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 1.7 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 16.4. Thread Pool Attributes

Attribute	Default Value	Description
name	default	Specifies the name of the thread pool.

Attribute	Default Value	Description
keepalive-time	10 seconds	Specifies the amount of time that pool threads should be kept after doing work.
allow-core-timeout	false	Boolean setting that determines whether core threads may time out. The default value is false.
thread-factory		Reference to the thread factory.
max-thread	50	The maximum thread pool size.
core-threads	50	The core thread pool size. This must be equal to or smaller than the maximum thread pool size.
queue-length	50	The maximum queue length.

Bootstrap Contexts

This is used to define custom bootstrap contexts. The following table describes the attributes you can set for bootstrap contexts.

Table 16.5. 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

This is 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 16.6. 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.
ignore-unknown-connections	false	Specifies that unknown connections will not be cached.

16.4. CONFIGURING RESOURCE ADAPTERS

16.4.1. Deploy a Resource Adapter

Resource adapters can be deployed just like other deployments using the management CLI or the management console. When running a standalone server, you can also copy the archive to the deployments directory to be picked up by the deployment scanner.

Deploy a Resource Adapter using the Management CLI
To deploy the resource adapter to a standalone server, enter the following management CLI command.

```
deploy /path/to/resource-adapter.rar
```

To deploy the resource adapter to all server groups in a managed domain, enter the following management CLI command.

```
deploy /path/to/resource-adapter.rar --all-server-groups
```

Deploy a Resource Adapter using the Management Console

1. Log in to the management console and click on the **Deployments** tab.
2. Click **Add**. In a managed domain, you will first need to select **Content Repository**.
3. Choose the **Upload a new deployment** option and click **Next**.
4. Browse to the resource adapter archive and click **Next**.
5. Verify the upload, then click **Finish**.
6. In a managed domain, assign the deployment to the appropriate server groups and enable the deployment.

Deploy a Resource Adapter Using the Deployment Scanner
To deploy a resource adapter manually to a standalone server, copy the resource adapter archive to the server deployments directory, for example, **`EAP_HOME/standalone/deployments/`**. This will be picked up and deployed by the deployment scanner.

NOTE

This option is not available for managed domains. You must use either the management console or the management CLI to deploy the resource adapter to server groups.

16.4.2. Configure a Resource Adapter

You can configure resource adapters using the management interfaces. The below example shows how to configure a resource adapter using the management CLI. See your resource adapter vendor's documentation for supported properties and other important information.

Add the Resource Adapter Configuration
Add the resource adapter configuration.

```
/subsystem=resource-adapters/resource-adapter=eis.rar:add(archive=eis.rar,
transaction-support=XATransaction)
```

Configure the Resource Adapter Settings
Configure any of the following settings as necessary.

- Configure **`config-properties`**.
  Add the **`server`** configuration property.

```
/subsystem=resource-adapters/resource-adapter=eis.rar/config-
properties=server:add(value=localhost)
```

  Add the **`port`** configuration property.

```
/subsystem=resource-adapters/resource-adapter=eis.rar/config-
properties=port:add(value=9000)
```

- Configure **`admin-objects`**.
  Add an admin object.

```
/subsystem=resource-adapters/resource-adapter=eis.rar/admin-
objects=aoName:add(class-name=com.acme.eis.ra.EISAdminObjectImpl,
jndi-name=java:/eis/AcmeAdminObject)
```

  Configure an admin object configuration property.

```
/subsystem=resource-adapters/resource-adapter=eis.rar/admin-
objects=aoName/config-properties=threshold:add(value=10)
```

- Configure **`connection-definitions`**.
  Add a connection definition for a managed connection factory.

```
/subsystem=resource-adapters/resource-adapter=eis.rar/connection-
definitions=cfName:add(class-
name=com.acme.eis.ra.EISManagedConnectionFactory, jndi-
name=java:/eis/AcmeConnectionFactory)
```

  Configure a managed connection factory configuration property.

```
/subsystem=resource-adapters/resource-adapter=eis.rar/connection-
definitions=cfName/config-properties=name:add(value=Acme Inc)
```

  Configure whether to record the enlistment trace. You can disable the recording of enlistment traces by setting the **`enlistment-trace`** attribute to **`false`**.

```
/subsystem=resource-adapters/resource-adapter=eis.rar/connection-
definitions=cfName:write-attribute(name=enlistment-
trace,value=false)
```

WARNING

Disabling the enlistment trace will make tracking down errors during transaction enlistment more difficult.

See Resource Adapter Attributes for all available configuration options for resource adapters.

Activate the Resource Adapter
Activate the resource adapter.

```
/subsystem=resource-adapters/resource-adapter=eis.rar:activate
```

NOTE

You can also define capacity policies for resource adapters. For more details, see the Capacity Policies section.

16.4.3. Deploy and Configure the Websphere MQ Resource Adapter

You can find the instructions for deploying the Websphere MQ resource adapter in *Configuring Messaging* for JBoss EAP.

16.4.4. Deploy and Configure the Generic JMS Resource Adapter

You can find the instructions for configuring the generic JMS resource adapter in *Configuring Messaging* for JBoss EAP.

16.5. CONFIGURE MANAGED CONNECTION POOLS

JBoss EAP provides three implementations of the `ManagedConnectionPool` interface.

`org.jboss.jca.core.connectionmanager.pool.mcp.SemaphoreConcurrentLinkedQueueManagedConnectionPool`

This is the default connection pool in JBoss EAP 7 and provides the best out-of-the-box performance.

`org.jboss.jca.core.connectionmanager.pool.mcp.SemaphoreArrayListManagedConnectionPool`

This was the default connection pool in previous JBoss EAP versions.

`org.jboss.jca.core.connectionmanager.pool.mcp.LeakDumperManagedConnectionPool`

This connection pool is used for debugging purposes only and will report any leaks upon shutdown or when the pool is flushed.

You can set the managed connection pool implementation for a datasource using the following management CLI command.

```
/subsystem=datasources/data-source=DATA_SOURCE:write-
attribute(name=mcp,value=MCP_CLASS)
```

You can set the managed connection pool implementation for a resource adapter using the following management CLI command.

```
/subsystem=resource-adapters/resource-adapter=RESOURCE_ADAPTER/connection-
definitions=CONNECTION_DEFINITION:write-
attribute(name=mcp,value=MCP_CLASS)
```

You can set the managed connection pool implementation for a messaging server using the following management CLI command.

```
/subsystem=messaging-activemq/server=SERVER/pooled-connection-
factory=CONNECTION_FACTORY:write-attribute(name=managed-connection-
pool,value=MCP_CLASS)
```

16.6. VIEW CONNECTION STATISTICS

You can read statistics for a defined connection from the **`/deployment=NAME.rar`** subtree. This is so that you can access statistics for any RAR, even if it not defined in a **`standalone.xml`** or **`domain.xml`** configuration.

```
/deployment=NAME.rar/subsystem=resource-
adapters/statistics=statistics/connection-definitions=java\:\/testMe:read-
resource(include-runtime=true)
```

NOTE

Be sure to specify the **`include-runtime=true`** argument, as all statistics are runtime-only information.

See Resource Adapter Statistics for information on the available statistics.

CHAPTER 17. CONFIGURING THE WEB SERVER (UNDERTOW)

17.1. UNDERTOW SUBSYSTEM OVERVIEW

IMPORTANT

In JBoss EAP 7, the **undertow** subsystem takes the place of the **web** subsystem from previous versions of JBoss EAP.

The **undertow** subsystem allows you to configure the web server and servlet container settings. It implements the Java Servlet 3.1 Specification as well as websockets and supports HTTP Upgrade and using high performance non-blocking handlers in servlet deployments. The **undertow** subsystem also has the ability to act as a high performance reverse proxy which supports mod_cluster.

Within the **undertow** subsystem, there are five main components to configure:

- buffer caches
- server
- servlet container
- handlers
- filters

NOTE

While JBoss EAP does offer the ability to update the configuration for each of these components, the default configuration is suitable for most use cases and provides reasonable performance settings.

Default Undertow Subsystem Configuration

```
<subsystem xmlns="urn:jboss:domain:undertow:3.1">
  <buffer-cache name="default"/>
  <server name="default-server">
      <http-listener name="default" socket-binding="http" redirect-
socket="https"/>
      <host name="default-host" alias="localhost">
          <location name="/" handler="welcome-content"/>
          <filter-ref name="server-header"/>
          <filter-ref name="x-powered-by-header"/>
      </host>
  </server>
  <servlet-container name="default">
      <jsp-config/>
      <websockets/>
  </servlet-container>
  <handlers>
      <file name="welcome-content" path="${jboss.home.dir}/welcome-
content"/>
  </handlers>
```

```
  <filters>
      <response-header name="server-header" header-name="Server" header-
value="JBoss-EAP/7"/>
      <response-header name="x-powered-by-header" header-name="X-Powered-
By" header-value="Undertow/1"/>
  </filters>
</subsystem>
```

IMPORTANT

The **undertow** subsystem also relies on the **io** subsystem to provide XNIO workers and buffer pools. The **io** subsystem is configured separately and provides a default configuration which should give optimal performance in most cases.

NOTE

Compared to the **web** subsystem in previous JBoss EAP releases, the **undertow** subsystem in JBoss EAP 7 has different default behaviors of HTTP methods .

17.2. CONFIGURING BUFFER CACHES

The buffer cache is used to cache static resources. JBoss EAP enables multiple caches to be configured and referenced by deployments, allowing different deployments to use different cache sizes. Buffers are allocated in regions and are a fixed size. The total amount of space used can be calculated by multiplying the buffer size by the number of buffers per region by the maximum number of regions. The default size of a buffer cache is 10MB.

JBoss EAP provides a single cache by default:

Default Undertow Subsystem Configuration

```
<subsystem xmlns="urn:jboss:domain:undertow:3.1">
  <buffer-cache name="default"/>
  ....
</subsystem>
```

Updating an Existing Buffer Cache
To update an existing buffer cache:

```
/subsystem=undertow/buffer-cache=default/:write-attribute(name=buffer-
size,value=2048)
```

```
reload
```

Creating a New Buffer Cache
To create a new buffer cache:

```
/subsystem=undertow/buffer-cache=new-buffer:add
```

Deleting a Buffer Cache
To delete a buffer cache:

```
/subsystem=undertow/buffer-cache=new-buffer:remove
```

```
reload
```

For a full list of the attributes available for configuring buffer caches, please see the Undertow Subsystem Attributes section.

17.3. CONFIGURING A SERVER

A server represents an instance of Undertow and consists of several elements:

- host
- http-listener
- https-listener
- ajp-listener

The host element provides a virtual host configuration while the three listeners provide connections of that type to the Undertow instance.

NOTE

Multiple servers may be configured, which allow deployments and servers to be completely isolated. This may be useful in certain scenarios such as multi-tenant environments

JBoss EAP provides a server by default:

Default Undertow Subsystem Configuration

```
<subsystem xmlns="urn:jboss:domain:undertow:3.1">
  <buffer-cache name="default"/>
  <server name="default-server">
      <http-listener name="default" socket-binding="http" redirect-
socket="https"/>
      <host name="default-host" alias="localhost">
          <location name="/" handler="welcome-content"/>
          <filter-ref name="server-header"/>
          <filter-ref name="x-powered-by-header"/>
      </host>
  </server>
  ...
</subsystem>
```

Updating an Existing Server
To update an existing server:

```
/subsystem=undertow/server=default-server:write-attribute(name=default-
host,value=default-host)
```

```
reload
```

Creating a New Server
To create a new server:

```
/subsystem=undertow/server=new-server:add
```

```
reload
```

Deleting a Server
To delete a server:

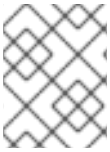
```
/subsystem=undertow/server=new-server:remove
```

```
reload
```

For a full list of the attributes available for configuring servers, see the Undertow Subsystem Attributes section.

17.4. CONFIGURING A SERVLET CONTAINER

A servlet container provides all servlet, JSP and websocket-related configuration, including session-related configuration. While most servers will only need a single servlet container, it is possible to configure multiple servlet containers by adding an additional *servlet-container* element. Having multiple servlet containers enables behavior such as allowing multiple deployments to be deployed to the same context path on different virtual hosts.



NOTE

Much of the configuration provided in by servlet container can be individually overridden by deployed applications using their `web.xml` file.

JBoss EAP provides a servlet container by default:

Default Undertow Subsystem Configuration

```
<subsystem xmlns="urn:jboss:domain:undertow:3.1">
  <buffer-cache name="default"/>
  <server name="default-server">
    ...
  </server>
  <servlet-container name="default">
      <jsp-config/>
      <websockets/>
  </servlet-container>
...
</subsystem>
```

Updating an Existing Servlet Container
To update an existing servlet container:

```
/subsystem=undertow/servlet-container=default:write-attribute(name=ignore-
flush,value=true)
```

```
reload
```

Creating a New Servlet Container
To create a new servlet container:

```
/subsystem=undertow/servlet-container=new-servlet-container:add
```

```
reload
```

Deleting a Servlet Container
To delete a servlet container:

```
/subsystem=undertow/servlet-container=new-servlet-container:remove
```

```
reload
```

For a full list of the attributes available for configuring servlet containers, see the Undertow Subsystem Attributes section.

17.5. CONFIGURING HANDLERS

JBoss EAP allows for two types of handlers to be configured:

- file handlers
- reverse-proxy handlers

File handlers serve static files. Each file handler must be attached to a location in a virtual host. Reverse-proxy handlers allow JBoss EAP to serve as a high performance reverse-proxy.

JBoss EAP provides a file handler by default:

Default Undertow Subsystem Configuration

```
<subsystem xmlns="urn:jboss:domain:undertow:3.1">
  <buffer-cache name="default"/>
  <server name="default-server">
      ...
  </server>
  <servlet-container name="default">
      ...
  </servlet-container>
  <handlers>
      <file name="welcome-content" path="${jboss.home.dir}/welcome-
content"/>
  </handlers>
  ...
</subsystem>
```

Using WebDAV for Static Resources
Previous versions of JBoss EAP allowed for using WebDAV with the **web** subsystem, by way of the **`WebdavServlet`**, to host static resources and enable additional HTTP methods for accessing and manipulating those files. In JBoss EAP 7, the **`undertow`** subsystem does provide a mechanism for

serving static files using a file handler, but the **undertow** subsystem does not support WebDAV. If you want to use WebDAV with JBoss EAP 7, you can write a custom WebDAV servlet.

Updating an Existing File Handler

To update an existing file handler:

```
/subsystem=undertow/configuration=handler/file=welcome-content:write-
attribute(name=case-sensitive,value=true)
```

```
reload
```

Creating a New File Handler

To create a new file handler:

```
/subsystem=undertow/configuration=handler/file=new-file-
handler:add(path="${jboss.home.dir}/welcome-content")
```

Deleting a File Handler

To delete a file handler

```
/subsystem=undertow/configuration=handler/file=new-file-handler:remove
```

```
reload
```

For a full list of the attributes available for configuring handlers, see the Undertow Subsystem Attributes section.

17.6. CONFIGURING FILTERS

A filter enables some aspect of a request to be modified and can use predicates to control when a filter executes. Some common use cases for filters include setting headers or doing GZIP compression.

NOTE

A filter is functionally equivalent to a global valve used in previous versions of JBoss EAP.

The following types of filters can be defined:

- custom-filter
- error-page
- expression-filter
- gzip
- mod-cluster
- request-limit
- response-header

- rewrite

JBoss EAP provides two filters by default:

Default Undertow Subsystem Configuration

```
<subsystem xmlns="urn:jboss:domain:undertow:3.1">
  <buffer-cache name="default"/>
  <server name="default-server">
    ...
  </server>
  <servlet-container name="default">
    ...
  </servlet-container>
  <handlers>
      ...
  </handlers>
  <filters>
      <response-header name="server-header" header-name="Server" header-
value="JBoss-EAP/7"/>
      <response-header name="x-powered-by-header" header-name="X-Powered-
By" header-value="Undertow/1"/>
  </filters>
</subsystem>
```

Updating an Existing Filter
To update an existing filter:

```
/subsystem=undertow/configuration=filter/response-header=server-
header:write-attribute(name=header-value,value="JBoss-EAP")
```

```
reload
```

Creating a New Filter
To create a new filter:

```
/subsystem=undertow/configuration=filter/response-header=new-response-
header:add(header-name=new-response-header,header-value="My Value")
```

Deleting a Filter
To delete a filter:

```
/subsystem=undertow/configuration=filter/response-header=new-response-
header:remove
```

```
reload
```

For a full list of the attributes available for configuring filters, see the Undertow Subsystem Attributes section.

17.7. CONFIGURE THE DEFAULT WELCOME WEB APPLICATION

JBoss EAP includes a default **Welcome** application, which displays at the root context on port 8080 by default.

There is a default server preconfigured in Undertow that serves up the welcome content.

Default Undertow Subsystem Configuration

```
<subsystem xmlns="urn:jboss:domain:undertow:3.1">
  ...
  <server name="default-server">
      <http-listener name="default" socket-binding="http" redirect-
socket="https"/>
      <host name="default-host" alias="localhost">
          <location name="/" handler="welcome-content"/>
          <filter-ref name="server-header"/>
          <filter-ref name="x-powered-by-header"/>
      </host>
  </server>
  ...
  <handlers>
      <file name="welcome-content" path="${jboss.home.dir}/welcome-
content"/>
  </handlers>
  ...
</subsystem>
```

The default server (**default-server**) has a default host (**default-host**) configured. The default host is configured to handle requests to the server's root, using the **<location>** element, with the **welcome-content** file handler. The **welcome-content** handler serves up the content in the location specified in the **path** property.

This default **Welcome** application can be replaced with your own web application. This can be configured in one of two ways:

- By changing the **welcome-content** file handler
- By changing the **default-web-module**

You can also disable the welcome content.

Changing the welcome-content File Handler

Modify the existing **welcome-content** file handler's path to point to the new deployment.

```
/subsystem=undertow/configuration=handler/file=welcome-content:write-
attribute(name=path,value="/path/to/content")
```

NOTE

Alternatively, you could create a different file handler to be used by the server's root.

```
/subsystem=undertow/configuration=handler/file=NEW_FILE_HANDLER
:add(path="/path/to/content")
/subsystem=undertow/server=default-server/host=default-
host/location=\/:write-
attribute(name=handler,value=NEW_FILE_HANDLER)
```

Reload the server for the changes to take effect.

```
reload
```

Changing the default-web-module
Map a deployed web application to the server's root.

```
/subsystem=undertow/server=default-server/host=default-host:write-
attribute(name=default-web-module,value=hello.war)
```

Reload the server for the changes to take effect.

```
reload
```

Disabling the Default Welcome Web Application
Disable the welcome application by removing the **`location`** entry (`/`) for the **`default-host`**.

```
/subsystem=undertow/server=default-server/host=default-
host/location=\/:remove
```

Reload the server for the changes to take effect.

```
reload
```

17.8. CONFIGURING HTTPS

For more details on configuring HTTPS for use with both web applications as well as the management interfaces please see the How to Configure Server Security Guide .

17.9. CONFIGURING HTTP SESSION TIMEOUT

The HTTP session timeout defines the period of inactive time needed to declare an HTTP session invalid. For example, a user accesses an application deployed to JBoss EAP which creates an HTTP session. If that user then attempts to access that application again after the HTTP session timeout, the original HTTP session will be invalidated and the user will be forced to create a new HTTP session. This may result in the loss of unpersisted data or the user having to re-authenticate.

The HTTP session timeout is configured in an application's **`web.xml`** file, but a default HTTP session timeout can be specified within JBoss EAP. The server's timeout value will apply to all deployed applications, but an application's **`web.xml`** will override the server's value.

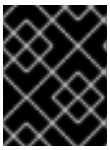
The server value is specified in the *default-session-timeout* property which is found in the *servlet-*

container section of the **undertow** subsystem. The value of *default-session-timeout* is specified in minutes and the default 30.

Configuring the Default Session Timeout
To configure the *default-session-timeout*:

```
/subsystem=undertow/servlet-container=default:write-
attribute(name=default-session-timeout, value=60)
```

```
reload
```



IMPORTANT

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.

17.10. CONFIGURING HTTP-ONLY SESSION MANAGEMENT COOKIES

Session management cookies can be accessed by both HTTP APIs and non-HTTP APIs such as JavaScript. JBoss EAP offers the ability to send the *HttpOnly* header as part of the *Set-Cookie* response header to the client, usually a browser. In supported browsers, enabling this header tells the browser that it should prevent accessing session management cookies through non-HTTP APIs. Restricting session management cookies to only HTTP APIs can help to mitigate the threat of session cookie theft via cross-site scripting attacks. To enable this behavior, the *http-only* attribute should be set to *true*.

IMPORTANT

Using the *HttpOnly* header does not actually prevent cross-site scripting attacks by itself, it merely notifies the browser. The browser must also support *HttpOnly* for this behavior to take affect.

IMPORTANT

Using the *http-only* attribute only applies the restriction to session management cookies and not other browser cookies.

The *http-only* attribute is set in two places in the **undertow** subsystem:

- In the servlet container as a session cookie setting
- In the host section of the server as a single sign-on property

Configuring *host-only* for the Servlet Container Session Cookie
To configure the *host-only* property for the servlet container session cookie:

```
/subsystem=undertow/servlet-container=default/setting=session-cookie:add
```

```
/subsystem=undertow/servlet-container=default/setting=session-
cookie:write-attribute(name=http-only,value=true)
```

```
reload
```

Configuring *host-only* for the Host Single Sign-On
To configure the *host-only* property for the host single sign-on:

```
/subsystem=undertow/server=default-server/host=default-
host/setting=single-sign-on:add
```

```
/subsystem=undertow/server=default-server/host=default-
host/setting=single-sign-on:write-attribute(name=http-only,value=true)
```

```
reload
```

17.11. CONFIGURING HTTP/2

Undertow allows for the use of the HTTP/2 standard, which reduces latency by compressing headers and multiplexing many streams over the same TCP connection. It also provides the ability for a server to push resources to the client before it has requested them, leading to faster page loads. Undertow is also compatible with SPDY, the predecessor to HTTP/2, in order to support clients that have not yet updated to the new specification.

IMPORTANT

HTTP/2 is supported as Technology Preview only for JBoss EAP 7.0 and will only work with browsers that also support the HTTP/2 standard.

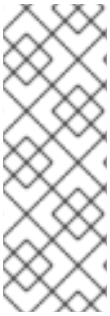
IMPORTANT

Using HTTP/2 requires the use of Java 8 as well as setting up ALPN on the classpath. This is due to the fact that HTTP/2 requires a TLS stack that supports ALPN, which is not provided by the default installation of Java 8.

17.11.1. Configuring Undertow to use HTTP/2

To configure Undertow to use HTTP/2, the following must be done:

Configure Undertow to use HTTPS
Please see the How to Configure Server Security Guide for configuring Undertow to use HTTPS for web applications.



NOTE

It is possible to use HTTP/2 without using HTTPS, in other words, only plain HTTP using HTTP upgrade. In that case, you do not need to install ALPN and can simply enable HTTP/2 in Undertow:

```
/subsystem=undertow/server=default-server/http-
listener=default:write-attribute(name=enable-http2,value=true)
```

Download the ALPN JAR
Start by determining the specific version of Java you have. Run the following command from the terminal to print the version of Java you have installed:

```
java -version
```

Based on your version, consult this page to determine the correct version of the ALPN JAR to download from this page. For example, if you were running Java version **`1.8.0_51`**, you would use ALPN version **`8.1.4.v20150727`** and download **`alpn-boot-8.1.4.v20150727.jar`**.

Add the ALPN JAR to the Boot Classpath
Once you have downloaded the correct version of the ALPN JAR, copy it to **`EAP_HOME/bin`**. You also must add the following to **`bin/standalone.conf`** (or **`bin/domain.conf`** if running in a managed domain) replacing **`$JBOSS_HOME`** and **`$ALPN_VERSION`** with the appropriate values.

```
JAVA_OPTS="$JAVA_OPTS -Xbootclasspath/p:$JBOSS_HOME/bin/alpn-boot-
$ALPN_VERSION.jar"
```

IMPORTANT

You must restart JBoss EAP for the classpath changes to take affect.

Enable HTTP/2 in the HTTPS Listener
To enable the HTTPS listener in Undertow to use HTTP/2, you must set the *enable-http2* attribute to *true*:

```
/subsystem=undertow/server=default-server/https-listener=https:write-
attribute(name=enable-http2,value=true)
```

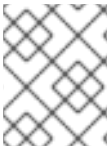
Verify HTTP/2 is Being Used
To verify that Undertow is using HTTP/2, you will need to inspect the headers coming from Undertow. Navigate to your JBoss EAP instance using https, for example *https://localhost:8443*, and use your browser's developer tools to inspect the headers. Some browsers, for example Google Chrome, will show HTTP/2 pseudo headers (*:path*, *:authority*, *:method* and *:scheme*) when using HTTP/2, while other browsers, for example Firefox and Safari, will report the status or version of the header as *HTTP/2.0*.

17.12. CONFIGURING A REQUESTDUMPING HANDLER

The *RequestDumping* handler (*io.undertow.server.handlers.RequestDumpingHandler*) logs the details of a request and corresponding response objects handled by Undertow within JBoss EAP.

IMPORTANT

While this handler can be useful for debugging, it may also log sensitive information. Please keep this in mind when enabling this handler.



NOTE

The *RequestDumping* handler replaces the *RequestDumperValve* from previous versions of JBoss EAP.

You can configure a *RequestDumping* handler at either at the server level directly in JBoss EAP or within an individual application.

17.12.1. Configuring a RequestDumping Handler on the Server

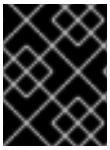
A *RequestDumping* handler should be configured as an expression filter. To configure a *RequestDumping* handler as an expression filter, you need to do the following:

Create a new Expression Filter with the *RequestDumping* Handler

```
/subsystem=undertow/configuration=filter/expression-
filter=requestDumperExpression:add(expression="dump-request")
```

Enable the Expression Filter in the Undertow Web Server

```
/subsystem=undertow/server=default-server/host=default-host/filter-
ref=requestDumperExpression:add
```



IMPORTANT

All requests and corresponding responses handled by the Undertow web server will be logged when enabling the *RequestDumping* handler as a expression filter in this manner.

Configuring a RequestDumping Handler for Specific URLs
In addition to logging all requests, you can also use an expression filter to only log requests and corresponding responses for specific URLs. This can be accomplished using a predicate in your expression such as *path*, *path-prefix*, or *path-suffix*. For example, if you want to log all requests and corresponding responses to */myApplication/test*, you can use the expression **`"path(/myApplication/test) -> dump-request"`** instead of the expression **`"dump-request"`** when creating your expression filter. This will only direct requests with a path exactly matching */myApplication/test* to the *RequestDumping* handler.

17.12.2. Configuring a RequestDumping Handler within an Application

In addition to configuring a *RequestDumping* handler at the server, you can also configure it within individual applications. This will limit the scope of the handler to only that specific application. A *RequestDumping* handler should be configured in **`WEB-INF/undertow-handlers.conf`**.

To configure the *RequestDumping* handler in **`WEB-INF/undertow-handlers.conf`** to log all requests and corresponding responses for this application, add the following expression to **`WEB-INF/undertow-handlers.conf`**:

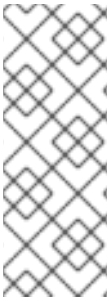
Example `WEB-INF/undertow-handlers.conf`

```
dump-request
```

To configure the *RequestDumping* handler in **`WEB-INF/undertow-handlers.conf`** to only log requests and corresponding responses to specific URLs within this application, you can use a predicate in your expression such as *path*, *path-prefix*, or *path-suffix*. For example, to log all requests and corresponding responses to **`test`** in your application, the following expression with the *path* predicate could be used:

Example `WEB-INF/undertow-handlers.conf`

```
path(/test) -> dump-request
```



NOTE

When using the predicates such as *path*, *path-prefix*, or *path-suffix* in expressions defined in the application's **`WEB-INF/undertow-handlers.conf`**, the value used will be relative to the context root of the application. For example, if the application's context root is *myApplication* with an expression **`path(/test) -> dump-request`** configured in **`WEB-INF/undertow-handlers.conf`**, it will only log requests and corresponding responses to */myApplication/test*.

CHAPTER 18. CONFIGURING REMOTING

18.1. ABOUT THE REMOTING SUBSYSTEM

The **remoting** subsystem allows you to configure inbound and outbound connections for local and remote services as well as the settings for those connections.

JBoss Remoting includes the following configurable elements: the endpoint, connectors, and a series of local and remote connection URIs. Most people will not need to configure the **remoting** subsystem at all, unless they use custom connectors for their own applications. Applications that act as remoting clients, such as EJBs, need separate configuration to connect to a specific connector.

Default Remoting Subsystem Configuration

```
<subsystem xmlns="urn:jboss:domain:remoting:3.0">
    <endpoint/>
    <http-connector name="http-remoting-connector" connector-ref="default"
security-realm="ApplicationRealm"/>
</subsystem>
```

See Remoting Subsystem Attributes for a full list of the attributes available for the **remoting** subsystem.

The Remoting Endpoint

The remoting endpoint uses the XNIO worker declared and configured by the **io** subsystem.

See Configuring the Endpoint for details on how to configure the remoting endpoint.

Connector

The connector is the main remoting configuration element. Multiple connectors are allowed. Each connector consists of a **<connector>** element with several sub-elements, and few other attributes. The default connector is used by several JBoss EAP subsystems. Specific settings for the elements and attributes of your custom connectors depend on your applications. Contact Red Hat Global Support Services for more information.

See Configuring a Connector for details on how to configure connectors.

Outbound Connections

You can specify three different types of outbound connections:

- An outbound connection, specified by a URI
- A local outbound connection, which connects to a local resource such as a socket
- A remote outbound connection, which connects to a remote resource and authenticates using a security realm

Additional Configuration

Remoting depends on several elements that are configured outside of the **remoting** subsystem, such as the network interface and IO worker.

For more information, see Additional Remoting Configuration.

18.2. CONFIGURING THE ENDPOINT

IMPORTANT

In JBoss EAP 6, the worker thread pool was configured directly in the **remoting** subsystem. In JBoss EAP 7, the remoting **endpoint** configuration references a worker from the **io** subsystem.

JBoss EAP provides the following **endpoint** configuration by default.

```
<subsystem xmlns="urn:jboss:domain:remoting:3.0">
  <endpoint/>
  ...
</subsystem>
```

Updating the Existing Endpoint Configuration

```
/subsystem=remoting/configuration=endpoint:write-
attribute(name=authentication-retries,value=2)
```

```
reload
```

Creating a New Endpoint Configuration

```
/subsystem=remoting/configuration=endpoint:add
```

Deleting an Endpoint Configuration

```
/subsystem=remoting/configuration=endpoint:remove
```

```
reload
```

See Endpoint Attributes for a full list of the attributes available for the endpoint configuration.

18.3. CONFIGURING A CONNECTOR

The connector is the main configuration element relating to remoting and contains several sub-elements for additional configuration.

Updating the Existing Connector Configuration

```
/subsystem=remoting/connector=new-connector:write-attribute(name=socket-
binding,value=my-socket-binding)
```

```
reload
```

Creating a New Connector

```
/subsystem=remoting/connector=new-connector:add(socket-binding=my-socket-
binding)
```

Deleting a Connector

```
/subsystem=remoting/connector=new-connector:remove
```

```
reload
```

For a full list of the attributes available for configuring a connector, please see the Remoting Subsystem Attributes section.

18.4. CONFIGURING AN HTTP CONNECTOR

The HTTP connector provides the configuration for the HTTP upgrade-based remoting connector. JBoss EAP provides the following **`http-connector`** configuration by default.

```
<subsystem xmlns="urn:jboss:domain:remoting:3.0">
    ...
    <http-connector name="http-remoting-connector" connector-ref="default"
security-realm="ApplicationRealm"/>
</subsystem>
```

By default, this HTTP connector connects to an HTTP listener named **`default`** that is configured in the **`undertow`** subsystem. For more information, see Configuring the Web Server (Undertow).

Updating the Existing HTTP Connector Configuration

```
/subsystem=remoting/http-connector=new-connector:write-
attribute(name=connector-ref,value=new-connector-ref)
```

```
reload
```

Creating a New HTTP Connector

```
/subsystem=remoting/http-connector=new-connector:add(connector-
ref=default)
```

Deleting an HTTP Connector

```
/subsystem=remoting/http-connector=new-connector:remove
```

See Connector Attributes for a full list of the attributes available for configuring an HTTP connector.

18.5. CONFIGURING AN OUTBOUND CONNECTION

An outbound connection is a generic remoting outbound connection that is fully specified by a URI.

Updating an Existing Outbound Connection

```
/subsystem=remoting/outbound-connection=new-outbound-connection:write-
attribute(name=uri,value=http://example.com)
```

Creating a New Outbound Connection

```
/subsystem=remoting/outbound-connection=new-outbound-
connection:add(uri=http://example.com)
```

-

Deleting an Outbound Connection

```
/subsystem=remoting/outbound-connection=new-outbound-connection:remove
```

See Outbound Connection Attributes for a full list of the attributes available for configuring an outbound connection.

18.6. CONFIGURING A REMOTE OUTBOUND CONNECTION

A remote outbound connection is specified by a protocol, an outbound socket binding, a username and a security realm. The protocol can be either **`remote`**, **`http-remoting`** or **`https-remoting`**.

Updating an Existing Remote Outbound Connection

```
/subsystem=remoting/remote-outbound-connection=new-remote-outbound-
connection:write-attribute(name=outbound-socket-binding-
ref,value=outbound-socket-binding)
```

Creating a New Remote Outbound Connection

```
/subsystem=remoting/remote-outbound-connection=new-remote-outbound-
connection:add(outbound-socket-binding-ref=outbound-socket-binding)
```

Deleting a Remote Outbound Connection

```
/subsystem=remoting/remote-outbound-connection=new-remote-outbound-
connection:remove
```

See Remote Outbound Connection Attributes for a full list of the attributes available for configuring a remote outbound connection.

18.7. CONFIGURING A LOCAL OUTBOUND CONNECTION

A local outbound connection is a remoting outbound connection with a protocol of **`local`**, specified only by an outbound socket binding.

Updating an Existing Local Outbound Connection

```
/subsystem=remoting/local-outbound-connection=new-local-outbound-
connection:write-attribute(name=outbound-socket-binding-
ref,value=outbound-socket-binding)
```

Creating a New Local Outbound Connection

```
/subsystem=remoting/local-outbound-connection=new-local-outbound-
connection:add(outbound-socket-binding-ref=outbound-socket-binding)
```

Deleting a Local Outbound Connection

```
/subsystem=remoting/local-outbound-connection=new-local-outbound-
connection:remove
```

See Local Outbound Connection Attributes for a full list of the attributes available for configuring a local outbound connection.

18.8. ADDITIONAL REMOTING CONFIGURATION

There are several remoting elements that are configured outside of the **remoting** subsystem.

IO worker

Use the following command to set the IO worker for remoting:

```
/subsystem=remoting/configuration=endpoint:write-attribute(name=worker,
value=WORKER_NAME)
```

See Configuring a Worker for details on how to configure an IO worker.

Network interface

The network interface used by the **remoting** subsystem is the **public** interface. This interface is also used by several other subsystems, so 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">
      <inet-address value="${jboss.bind.address.unsecure:127.0.0.1}"/>
   </interface>
</interfaces>
```

In a managed domain, the **public** interface is defined per host in its **host.xml** file.

Socket binding

The default socket binding used by the **remoting** subsystem binds to port **8080**.
For more information about socket binding and socket binding groups, see Socket Bindings.

Remoting connector reference for EJB

The **ejb3** 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 connection, such as HTTPS, Secure Servlet, 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 components that use remoting, such as EJBs, 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 defaults to an unsecured connection. It is inherently susceptible to a man-in-the-middle exploit, where an attacker intercepts the request of the client 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 is an appropriate fall-back.

CHAPTER 19. CONFIGURING THE IO SUBSYSTEM

19.1. IO SUBSYSTEM OVERVIEW

The **io** subsystem defines the XNIO workers and buffer pools used by other subsystems, such as Undertow and Remoting. These workers and buffer pools are defined within the following components in the **io** subsystem:

Default IO Subsystem Configuration

```
<subsystem xmlns="urn:jboss:domain:io:1.1">
    <worker name="default"/>
    <buffer-pool name="default"/>
</subsystem>
```

19.2. CONFIGURING A WORKER

Workers are XNIO worker instances. An XNIO worker instance is an abstraction layer for the Java NIO APIs, which provide functionality such as management of IO and worker threads as well as SSL support. By default, JBoss EAP provides single worker called **default**, but more can be defined.

Updating an Existing Worker
To update an existing worker:

```
/subsystem=io/worker=default:write-attribute(name=io-threads,value=10)
```

```
reload
```

Creating a New Worker
To create a new worker:

```
/subsystem=io/worker=newWorker:add
```

Deleting a Worker
To delete a worker:

```
/subsystem=io/worker=newWorker:remove
```

```
reload
```

For a full list of the attributes available for configuring workers, please see the IO Subsystem Attributes section.

19.3. CONFIGURING A BUFFER POOL

Buffer Pools are pooled NIO buffer instances.

IMPORTANT

Changing the buffer size has a big impact on application performance. For most servers, the ideal size is usually 16k.

Updating an Existing Buffer Pool

To update an existing buffer pool:

```
/subsystem=io/buffer-pool=default:write-attribute(name=direct-
buffers,value=true)
```

```
reload
```

Creating a Buffer Pool

To create a new buffer pool:

```
/subsystem=io/buffer-pool=newBuffer:add
```

Deleting a Buffer Pool

To delete a buffer pool:

```
/subsystem=io/buffer-pool=newBuffer:remove
```

```
reload
```

For a full list of the attributes available for configuring buffer pools, please see the IO Subsystem Attributes section.

CHAPTER 20. CONFIGURING BATCH APPLICATIONS

JBoss EAP 7 introduced support for Java batch applications as defined by JSR-352. You can configure an environment for running batch applications and manage batch jobs using the **batch-jberet** subsystem.

For information on developing batch applications, see Java Batch Application Development in the JBoss EAP *Development Guide*.

20.1. CONFIGURING BATCH JOBS

You can configure settings for batch jobs using the **batch-jberet** subsystem, which is based on the JBeret implementation.

The default **batch-jberet** subsystem configuration defines an in-memory job repository and default thread pool settings.

```
<subsystem xmlns="urn:jboss:domain:batch-jberet:1.0">
    <default-job-repository name="in-memory"/>
    <default-thread-pool name="batch"/>
    <job-repository name="in-memory">
        <in-memory/>
    </job-repository>
    <thread-pool name="batch">
        <max-threads count="10"/>
        <keepalive-time time="30" unit="seconds"/>
    </thread-pool>
</subsystem>
```

By default, any batch jobs stopped during a server suspend will be restarted upon server resume. You can set the **restart-jobs-on-resume** property to **false** to leave jobs in the **STOPPED** state instead.

```
/subsystem=batch-jberet:write-attribute(name=restart-jobs-on-
resume,value=false)
```

You can also configure the settings for batch job repositories and thread pools.

20.1.1. Configure Batch Job Repositories

This section shows you how to configure in-memory and JDBC job repositories for storing batch job information using the management CLI. You can also configure job repositories using the management console by navigating to the **Batch** subsystem from the **Configuration** tab and selecting either **In Memory** or **JDBC** from the left-hand menu.

Add an In-memory Job Repository
You can add a job repository that stores batch job information in memory.

```
/subsystem=batch-jberet/in-memory-job-repository=REPOSITORY_NAME:add
```

Add a JDBC Job Repository
You can add a job repository that stores batch job information in a database. You must specify the name of the datasource for connecting to the database.

-

```
/subsystem=batch-jberet/jdbc-job-repository=REPOSITORY_NAME:add(data-
source=DATASOURCE)
```

Set a Default Job Repository
You can set an in-memory or JDBC job repository as the default job repository for batch applications.

```
/subsystem=batch-jberet:write-attribute(name=default-job-
repository,value=REPOSITORY_NAME)
```

This will require a server reload.

```
reload
```

20.1.2. Configure Batch Thread Pools

This section shows you how to configure thread pools and thread factories to be used for batch jobs using the management CLI. You can also configure thread pools and thread factories using the management console by navigating to the **Batch** subsystem from the **Configuration** tab and selecting **Thread Pools** or **Thread Factories** from the left-hand menu.

Configure a Thread Pool
When adding a thread pool, you must specify the **`max-threads`**, which should always be greater than **`3`** as two threads are reserved to ensure partition jobs can execute as expected.

1. Add a thread pool.

```
/subsystem=batch-jberet/thread-pool=THREAD_POOL_NAME:add(max-
threads=10)
```

2. If desired, set a **`keepalive-time`** value.

```
/subsystem=batch-jberet/thread-pool=THREAD_POOL_NAME:write-
attribute(name=keepalive-time,value={time=60,unit=SECONDS})
```

Use a Thread Factory

1. Add a thread factory.

```
/subsystem=batch-jberet/thread-factory=THREAD_FACTORY_NAME:add
```

2. Configure the desired attributes for the thread factory.
   - **`group-name`** - The name of a thread group to create for this thread factory.
   - **`priority`** - The thread priority of created threads.
   - **`thread-name-pattern`** - The template used to create names for threads. The following patterns may be used:
     - **`%%`** - A percent sign
     - **`%t`** - The per-factory thread sequence number
     - **`%g`** - The global thread sequence number

- **%f** - The factory sequence number
- **%i** - The thread ID

3. Assign the thread factory to a thread pool.

```
/subsystem=batch-jberet/thread-pool=THREAD_POOL_NAME:write-
attribute(name=thread-factory,value=THREAD_FACTORY_NAME)
```

This will require a server reload.

```
reload
```

Set a Default Thread Pool
You can set a different thread pool as the default thread pool.

```
/subsystem=batch-jberet:write-attribute(name=default-thread-
pool,value=THREAD_POOL_NAME)
```

This will require a server reload.

```
reload
```

View Thread Pool Statistics
You can view runtime information about a batch thread pool using the **read-resource** management CLI operation. You must use the **include-runtime=true** parameter in order to see this runtime information.

```
/subsystem=batch-jberet/thread-pool=THREAD_POOL_NAME:read-
resource(include-runtime=true)
{
    "outcome" => "success",
    "result" => {
        "active-count" => 0,
        "completed-task-count" => 0L,
        "current-thread-count" => 0,
        "keepalive-time" => undefined,
        "largest-thread-count" => 0,
        "max-threads" => 15,
        "name" => "THREAD_POOL_NAME",
        "queue-size" => 0,
        "rejected-count" => 0,
        "task-count" => 0L,
        "thread-factory" => "THREAD_FACTORY_NAME"
    }
}
```

You can also view runtime information for batch thread pools using the management console by navigating to the **Batch** subsystem from the **Runtime** tab.

20.2. MANAGING BATCH JOBS

The **batch-jberet** subsystem resource for deployments allows you to start, stop, and restart batch jobs. You can also view the details of job executions.

Restart a Batch Job

You can restart a job that is in a **STOPPED** or **FAILED** state by providing its execution ID and optionally any properties to use when restarting the batch job.

```
/deployment=DEPLOYMENT_NAME/subsystem=batch-jberet:restart-job(execution-
id=EXECUTION_ID,properties={PROPERTY=VALUE})
```

The execution ID must be the most recent execution of the job instance.

Start a Batch Job

You can start a batch job by providing the job XML file and optionally any properties to use when starting the batch job.

```
/deployment=DEPLOYMENT_NAME/subsystem=batch-jberet:start-job(job-xml-
name=JOB_XML_NAME,properties={PROPERTY=VALUE})
```

Stop a Batch Job

You can stop a running batch job by providing its execution ID.

```
/deployment=DEPLOYMENT_NAME/subsystem=batch-jberet:stop-job(execution-
id=EXECUTION_ID)
```

View Batch Job Execution Details

You can view the details of batch job executions. You must use the **include-runtime=true** parameter in order to see this runtime information.

```
/deployment=DEPLOYMENT_NAME/subsystem=batch-jberet:read-
resource(recursive=true,include-runtime=true)
{
    "outcome" => "success",
    "result" => {"job" => {"import-file" => {
        "instance-count" => 2,
        "running-executions" => 0,
        "execution" => {
            "2" => {
                "batch-status" => "COMPLETED",
                "create-time" => "2016-04-11T22:03:12.708-0400",
                "end-time" => "2016-04-11T22:03:12.718-0400",
                "exit-status" => "COMPLETED",
                "instance-id" => 58L,
                "last-updated-time" => "2016-04-11T22:03:12.719-0400",
                "start-time" => "2016-04-11T22:03:12.708-0400"
            },
            "1" => {
                "batch-status" => "FAILED",
                "create-time" => "2016-04-11T21:57:17.567-0400",
                "end-time" => "2016-04-11T21:57:17.596-0400",
                "exit-status" => "Error :
org.hibernate.exception.ConstraintViolationException: could not execute
statement",
                "instance-id" => 15L,
```

```
                "last-updated-time" => "2016-04-11T21:57:17.597-0400",
                "start-time" => "2016-04-11T21:57:17.567-0400"
            }
        }
    }}}
}
```

CHAPTER 21. CONFIGURING THE NAMING SUBSYSTEM

21.1. ABOUT THE NAMING SUBSYSTEM

The **naming** subsystem provides the JNDI implementation for JBoss EAP. You can configure this subsystem to bind entries in global JNDI namespaces . You can also configure it to activate or deactivate the remote JNDI interface.

The following is an example of the **naming** subsystem XML configuration example with all of the elements and attributes specified.

```
<subsystem xmlns="urn:jboss:domain:naming:2.0">
    <bindings>
        <simple name="java:global/simple-integer-binding" value="100"
type="int" />
        <simple name="java:global/jboss.org/docs/url"
value="https://docs.jboss.org" type="java.net.URL" />
        <object-factory name="java:global/foo/bar/factory"
module="org.foo.bar" class="org.foo.bar.ObjectFactory" />
        <external-context name="java:global/federation/ldap/example"
class="javax.naming.directory.InitialDirContext" cache="true">
            <environment>
                <property name="java.naming.factory.initial"
value="com.sun.jndi.ldap.LdapCtxFactory" />
                <property name="java.naming.provider.url"
value="ldap://ldap.example.com:389" />
                <property name="java.naming.security.authentication"
value="simple" />
                <property name="java.naming.security.principal"
value="uid=admin,ou=system" />
                <property name="java.naming.security.credentials"
value="secret" />
            </environment>
        </external-context>
        <lookup name="java:global/new-alias-name"
lookup="java:global/original-name" />
    </bindings>
    <remote-naming/>
</subsystem>
```

21.2. CONFIGURING GLOBAL BINDINGS

The **naming** subsystem allows you to bind entries into the **java:global**, **java:jboss**, or **java** global JNDI namespaces; however, it is recommended that you use the standard portable **java:global** namespace.

Global bindings are configured in the **<bindings>** element of the **naming** subsystem. Four types of bindings are supported:

- simple bindings
- object factory bindings
- external context bindings

- binding lookup aliases

Configuring Simple Bindings

The **simple** XML configuration element binds primitive or **java.net.URL** entries.

- The **name** attribute is mandatory and specifies the target JNDI name for the entry.
- The **value** attribute is mandatory and defines the entry's value.
- The optional **type** attribute, which defaults to **java.lang.String**, specifies the type of the entry's value. In addition to **java.lang.String**, you can specify primitive types and their corresponding object wrapper classes, such as **int** or **java.lang.Integer**, and **java.net.URL**.

The following is an example of a management CLI command that creates a simple binding.

```
/subsystem=naming/binding=java\:global\/simple-integer-
binding:add(binding-type=simple, type=int, value=100)
```

Resulting XML Configuration

```
<subsystem xmlns="urn:jboss:domain:naming:2.0">
    <bindings>
        <simple name="java:global/simple-integer-binding" value="100"
type="int"/>
    </bindings>
    <remote-naming/>
</subsystem>
```

Use the following command to remove the binding.

```
/subsystem=naming/binding=java\:global\/simple-integer-binding:remove
```

Binding Object Factories

The **object-factory** XML configuration element binds **javax.naming.spi.ObjectFactory** entries.

- The **name** attribute is mandatory and specifies the target JNDI name for the entry.
- The **class** attribute is mandatory and defines the object factory's Java type.
- The **module** attribute is mandatory and specifies the JBoss Module ID where the object factory Java class can be loaded from.
- The optional **environment** child element can be used to provide a custom environment to the object factory.

The following is an example of a management CLI command that creates an object factory binding.

```
/subsystem=naming/binding=java\:global\/foo\/bar\/factory:add(binding-
type=object-factory, module=org.foo.bar, class=org.foo.bar.ObjectFactory,
environment=[p1=v1, p2=v2])
```

Resulting XML Configuration

```
<subsystem xmlns="urn:jboss:domain:naming:2.0">
    <bindings>
        <object-factory name="java:global/foo/bar/factory"
module="org.foo.bar" class="org.foo.bar.ObjectFactory">
            <environment>
                <property name="p1" value="v1" />
                <property name="p2" value="v2" />
            </environment>
        </object-factory>
    </bindings>
  </subsystem>
```

Use the following command to remove the binding.

```
/subsystem=naming/binding=java\:global\/foo\/bar\/factory:remove
```

Binding External Contexts

Federation of external JNDI contexts, such as LDAP context, are achieved using the **`external-context`** XML configuration element.

- The **`name`** attribute is mandatory and specifies the target JNDI name for the entry.
- The **`class`** attribute is mandatory and indicates the Java initial naming context type used to create the federated context. Note that such type must have a constructor with a single environment map argument.
- The optional **`module`** attribute specifies the JBoss Module ID where any classes required by the external JNDI context can be loaded from.
- The optional **`cache`** attribute, which defaults to **`false`**, indicates whether the external context instance should be cached.
- The optional **`environment`** child element can be used to provide the custom environment needed to look up the external context.

The following is an example of a management CLI command that creates an external context binding.

```
/subsystem=naming/binding=java\:global\/federation\/ldap\/example:add(bind
ing-type=external-context, cache=true,
class=javax.naming.directory.InitialDirContext,
module=org.jboss.as.naming, environment=
[java.naming.factory.initial=com.sun.jndi.ldap.LdapCtxFactory,
java.naming.provider.url="ldap://ldap.example.com:389",
java.naming.security.authentication=simple,
java.naming.security.principal="uid=admin,ou=system",
java.naming.security.credentials=secret]
```

Resulting XML Configuration

```
<subsystem xmlns="urn:jboss:domain:naming:2.0">
  <bindings>
    <external-context name="java:global/federation/ldap/example"
module="org.jboss.as.naming"
class="javax.naming.directory.InitialDirContext" cache="true">
```

```
      <environment>
        <property name="java.naming.factory.initial"
value="com.sun.jndi.ldap.LdapCtxFactory"/>
        <property name="java.naming.provider.url"
value="ldap://ldap.example.com:389"/>
        <property name="java.naming.security.authentication"
value="simple"/>
        <property name="java.naming.security.principal"
value="uid=admin,ou=system"/>
        <property name="java.naming.security.credentials" value="secret"/>
      </environment>
    </external-context>
  </bindings>
</subsystem>
```

Use the following command to remove the binding.

```
/subsystem=naming/binding=java\:global\/federation\/ldap\/example:remove
```

NOTE

Resource look up for JNDI providers that do not properly implement the **`lookup(Name)`** method can result in a "javax.naming.InvalidNameException: Only support CompoundName names" error.

You might be able to work around this issue by specifying that the external context environment use the **`lookup(String)`** method instead by adding the following property; however, this can result in a performance degradation.

```
<property name="org.jboss.as.naming.lookup.by.string"
value="true"/>
```

Binding Lookup Aliases

The **`lookup`** element allows you to bind existing entries into additional names or aliases.

- The **`name`** attribute is mandatory and specifies the target JNDI name for the entry.
- The **`lookup`** attribute is mandatory and indicates the source JNDI name.

The following is an example of a management CLI command that binds an existing entry to an alias.

```
/subsystem=naming/binding=java\:global\/new-alias-name:add(binding-
type=lookup, lookup=java\:global\/original-name)
```

Resulting XML Configuration

```
<lookup name="java:global/new-alias-name" lookup="java:global/original-
name" />
```

Use the following command to remove the binding.

```
/subsystem=naming/binding=java\:global\/c:remove
```

21.3. CONFIGURING THE REMOTE JNDI INTERFACE

The remote JNDI interface allows clients to look up entries in remote JBoss EAP instances. The **naming** subsystem can be configured to deactivate or activate this interface, which is activated by default. The remote JNDI interface is configured using the **<remote-naming>** element.

Use the following management CLI command to activate or reactivate the remote JNDI interface.

```
/subsystem=naming/service=remote-naming:add
```

Use the following management CLI command to deactivate the remote JNDI interface.

```
/subsystem=naming/service=remote-naming:remove
```

NOTE

Only entries within the **java:jboss/exported** context are accessible over remote JNDI.

CHAPTER 22. CONFIGURING HIGH AVAILABILITY

22.1. INTRODUCTION TO HIGH AVAILABILITY

JBoss EAP provides the following *high availability* services to guarantee the availability of deployed Java EE applications.

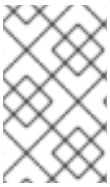
Load balancing

This allows a service to handle a large number of requests by spreading the workload across multiple servers. A client can have timely responses from the service even in the event of a high volume of requests.

Failover

This allows a client to have uninterrupted access to a service even in the event of hardware or network failures. If the service fails, another cluster member takes over the client's requests so that it can continue processing.

Clustering is a term that encompasses all of these capabilities. Members of a cluster can be configured to share workloads (load balancing) and pick up client processing in the event of a failure of another cluster member (failover).



NOTE

It is important to keep in mind that the JBoss EAP operating mode chosen (*standalone server* or *managed domain*) pertains to how you want to manage your servers. High availability services can be configured in JBoss EAP regardless of its operating mode.

JBoss EAP supports high availability at several different levels using various components. Some of those 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 and stateless session Enterprise JavaBeans (EJBs)
- Single sign-on (SSO) mechanisms
- HTTP sessions
- JMS services and message-driven beans (MDBs)
- Singleton MSC services
- Singleton deployments

Clustering is made available to JBoss EAP by the `jgroups`, `infinispan`, and `modcluster` subsystems. The *ha* and *full-ha* profiles have these systems enabled. In JBoss EAP, 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.

See the JBoss EAP *Development Guide* for how to mark an application as distributable .

22.2. CLUSTER COMMUNICATION WITH JGROUPS

22.2.1. About JGroups

JGroups is a toolkit for reliable messaging and can be used to create clusters whose nodes can send messages to each other.

The **jgroups** subsystem provides group communication support for high availability services in JBoss EAP. It allows you to configure named channels and protocol stacks as well as view runtime statistics for channels. The **jgroups** subsystem is available when using a configuration that provides high availability capabilities, such as the *ha* or *full-ha* profile in a managed domain, or the **standalone-ha.xml** or **standalone-full-ha.xml** configuration file for a standalone server.

JBoss EAP is preconfigured with two JGroups stacks:

udp

The nodes in the cluster use User Datagram Protocol (UDP) multicasting to communicate with each other. This is the default stack.

tcp

The nodes in the cluster use Transmission Control Protocol (TCP) to communicate with each other.

You can use the preconfigured stacks or define your own to suit your system's specific requirements.

NOTE

TCP has more overhead and is often considered slower than UDP since it handles error checking, packet ordering, and congestion control itself. JGroups handles these features for UDP, whereas TCP guarantees them itself. TCP is a good choice when using JGroups on unreliable or high congestion networks, or when multicast is not available.

22.2.2. Switch the Default JGroups Channel to Use TCP

By default, cluster nodes communicate using the UDP protocol. The default **ee** JGroups channel uses the predefined **udp** protocol stack.

```
<channels default="ee">
  <channel name="ee" stack="udp"/>
</channels>
<stacks>
  <stack name="udp">
    <transport type="UDP" socket-binding="jgroups-udp"/>
    <protocol type="PING"/>
    ...
  </stack>
  <stack name="tcp">
    <transport type="TCP" socket-binding="jgroups-tcp"/>
    <protocol type="MPING" socket-binding="jgroups-mping"/>
    ...
  </stack>
</stacks>
```

Some networks only allow TCP to be used. Use the following management CLI command to switch the **ee** channel to use the preconfigured **tcp** stack.

```
/subsystem=jgroups/channel=ee:write-attribute(name=stack,value=tcp)
```

This default **tcp** stack uses the **MPING** protocol, which uses IP multicast to discover the initial cluster membership. See the following sections for configuring stacks for alternative membership discovery protocols:

- Using the TCPPING protocol to define a static cluster membership list.
- Using the TCPGOSSIP protocol to use an external gossip router to discover the members of a cluster.

22.2.3. Configure TCPPING

This procedure creates a new JGroups stack that uses the **TCPPING** protocol to define a static cluster membership list. A base script is provided that creates a **tcpping** stack and sets the default **ee** channel to use this new stack. The management CLI commands in this script must be customized for your environment and will be processed as a batch.

1. Copy the following script into a text editor and save it to the local file system.

```
batch
# Add the tcpping stack
/subsystem=jgroups/stack=tcpping:add
/subsystem=jgroups/stack=tcpping/transport=TCP:add(socket-
binding=jgroups-tcp)
/subsystem=jgroups/stack=tcpping/protocol=TCPPING:add
# Set the properties for the TCPPING protocol
/subsystem=jgroups/stack=tcpping/protocol=TCPPING:write-
attribute(name=properties,value=
{initial_hosts="HOST_A[7600],HOST_B[7600]",port_range=0,timeout=3000
})
/subsystem=jgroups/stack=tcpping/protocol=MERGE3:add
/subsystem=jgroups/stack=tcpping/protocol=FD_SOCK:add(socket-
binding=jgroups-tcp-fd)
/subsystem=jgroups/stack=tcpping/protocol=FD:add
/subsystem=jgroups/stack=tcpping/protocol=VERIFY_SUSPECT:add
/subsystem=jgroups/stack=tcpping/protocol=pbcast.NAKACK2:add
/subsystem=jgroups/stack=tcpping/protocol=UNICAST3:add
/subsystem=jgroups/stack=tcpping/protocol=pbcast.STABLE:add
/subsystem=jgroups/stack=tcpping/protocol=pbcast.GMS:add
/subsystem=jgroups/stack=tcpping/protocol=MFC:add
/subsystem=jgroups/stack=tcpping/protocol=FRAG2:add
# Set tcpping as the stack for the ee channel
/subsystem=jgroups/channel=ee:write-
attribute(name=stack,value=tcpping)
run-batch
reload
```

   Note that the order of protocols defined is important.

2. Modify the script for your environment.
   - If you are running in a managed domain, you must specify which profile to update by preceding the **/subsystem=jgroups** commands with **/profile=PROFILE_NAME**.

- Adjust the TCPPING properties, which are optional, for your environment:
  - **`initial_hosts`**: A comma-separated list of the 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`**.
  - **`timeout`**: A timeout value, in milliseconds, for cluster members.

3. Run the script by passing the script file to the management CLI.

```
$ EAP_HOME/bin/jboss-cli.sh --connect --file=/path/to/SCRIPT_NAME
```

The TCPPING stack is now available and TCP is used for network communication.

22.2.4. Configure TCPGOSSIP

This procedure creates a new JGroups stack that uses the **`TCPGOSSIP`** protocol to use an external gossip router to discover the members of a cluster. A base script is provided that creates a **`tcpgossip`** stack and sets the default **`ee`** channel to use this new stack. The management CLI commands in this script must be customized for your environment and will be processed as a batch.

1. Copy the following script into a text editor and save it to the local file system.

```
batch
# Add the tcpgossip stack
/subsystem=jgroups/stack=tcpgossip:add
/subsystem=jgroups/stack=tcpgossip/transport=TCP:add(socket-
binding=jgroups-tcp)
/subsystem=jgroups/stack=tcpgossip/protocol=TCPGOSSIP:add
# Set the properties for the TCPGOSSIP protocol
/subsystem=jgroups/stack=tcpgossip/protocol=TCPGOSSIP:write-
attribute(name=properties,value={initial_hosts="HOST_A[13001]"})
/subsystem=jgroups/stack=tcpgossip/protocol=MERGE3:add
/subsystem=jgroups/stack=tcpgossip/protocol=FD_SOCK:add(socket-
binding=jgroups-tcp-fd)
/subsystem=jgroups/stack=tcpgossip/protocol=FD:add
/subsystem=jgroups/stack=tcpgossip/protocol=VERIFY_SUSPECT:add
/subsystem=jgroups/stack=tcpgossip/protocol=pbcast.NAKACK2:add
/subsystem=jgroups/stack=tcpgossip/protocol=UNICAST3:add
/subsystem=jgroups/stack=tcpgossip/protocol=pbcast.STABLE:add
/subsystem=jgroups/stack=tcpgossip/protocol=pbcast.GMS:add
/subsystem=jgroups/stack=tcpgossip/protocol=MFC:add
/subsystem=jgroups/stack=tcpgossip/protocol=FRAG2:add
# Set tcpgossip as the stack for the ee channel
/subsystem=jgroups/channel=ee:write-
attribute(name=stack,value=tcpgossip)
run-batch
reload
```

Note that the order of protocols defined is important.

2. Modify the script for your environment.
    - If you are running in a managed domain, you must specify which profile to update by preceding the `/subsystem=jgroups` commands with `/profile=PROFILE_NAME`.
    - Adjust the TCPGOSSIP properties, which are optional, for your environment:
        - `initial_hosts`: A comma-separated list of the hosts, using the syntax `HOST[PORT]`, that are considered well-known and will be available to look up the initial membership.
        - `reconnect_interval`: The interval in milliseconds by which a disconnected stub attempts to reconnect to the gossip router.
        - `sock_conn_timeout`: The maximum time for socket creation. The default is `1000` milliseconds.
        - `sock_read_timeout`: The maximum time in milliseconds to block on a read. A value of `0` will block indefinitely.
3. Run the script by passing the script file to the management CLI.

```
$ EAP_HOME/bin/jboss-cli.sh --connect --file=/path/to/SCRIPT_NAME
```

The TCPGOSSIP stack is now available and TCP is used for network communication. This stack is configured for use with a gossip router so that JGroups cluster members can find other cluster members.

22.2.5. Binding JGroups to a Network Interface

By default, JGroups only binds to the `private` network interface, which points to localhost in the default configuration. For security reasons, JGroups will not bind to the network interface defined by the `-b` argument specified during JBoss EAP startup, as clustering traffic should not be exposed on a public network interface.

See the Network and Port Configuration chapter in this guide for information about how to configure network interfaces.

IMPORTANT

For security reasons, JGroups should only be bound to a non-public network interface. For performance reasons, we also recommend that the network interface for JGroups traffic should be part of a dedicated Virtual Local Area Network (VLAN).

22.2.6. Securing a Cluster

There are several concerns to address in order to run a cluster securely:

- Preventing unauthorized nodes from joining the cluster. This is addressed by requiring authentication.
- Preventing non-members from communicating with cluster members. This is addressed by encrypting messages.

Configuring Authentication

JGroups authentication is performed by the **AUTH** protocol. The purpose is to ensure that only authenticated nodes can join a cluster.

In the applicable server configuration file, add the **AUTH** protocol with the appropriate property settings. The **AUTH** protocol should be configured immediately before the **pbcast.GMS** protocol.

```
<subsystem xmlns="urn:jboss:domain:jgroups:4.0">
  <stacks>
    <stack name="udp">
      <transport type="UDP" socket-binding="jgroups-udp"/>
      <protocol type="PING"/>
      <protocol type="MERGE3"/>
      <protocol type="FD_SOCK" socket-binding="jgroups-udp-fd"/>
      <protocol type="FD_ALL"/>
      <protocol type="VERIFY_SUSPECT"/>
      <protocol type="pbcast.NAKACK2"/>
      <protocol type="UNICAST3"/>
      <protocol type="pbcast.STABLE"/>
      <protocol type="AUTH">
        <property name="auth_class">org.jgroups.auth.MD5Token</property>
        <property name="auth_value">mytoken</property> <!-- Change this
password -->
        <property name="token_hash">MD5</property>
      </protocol>
      <protocol type="pbcast.GMS"/>
      <protocol type="UFC"/>
      <protocol type="MFC"/>
      <protocol type="FRAG2"/>
    </stack>
  </stacks>
</subsystem>
```

Configuring Encryption

To encrypt messages, JGroups uses a secret key that is shared by the members of a cluster. The sender encrypts the message using the shared secret key, and the receiver decrypts the message using the same secret key. With symmetric encryption, which is configured using the **SYM_ENCRYPT** protocol, nodes use a shared keystore to retrieve the secret key. With asymmetric encryption, which is configured using the **ASYM_ENCRYPT** protocol, nodes retrieve the secret key from the coordinator of the cluster after being authenticated using **AUTH**.

IMPORTANT

You must apply Red Hat JBoss Enterprise Application Platform 7.0 Update 01 or a newer cumulative patch to your JBoss EAP installation in order to have access to the **SYM_ENCRYPT** and **ASYM_ENCRYPT** protocols.

See the JBoss EAP *Patching and Upgrading Guide* for information on applying cumulative patches.

Using Symmetric Encryption

In order to use **SYM_ENCRYPT**, you must set up a keystore that will be referenced in the JGroups configuration for each node.

1. Create a keystore.
   In the following command, replace **VERSION** with the appropriate JGroups JAR version and

PASSWORD with a keystore password.

```
$ java -cp
EAP_HOME/modules/system/layers/base/org/jgroups/main/jgroups-
VERSION.jar org.jgroups.demos.KeyStoreGenerator --alg AES --size 128
--storeName defaultStore.keystore --storepass PASSWORD --alias mykey
```

This will generate a **defaultStore.keystore** file that will be referenced in the JGroups configuration.

2. Configure the **SYM_ENCRYPT** protocol in the **jgroups** subsystem.
   In the applicable server configuration file, add the **SYM_ENCRYPT** protocol with the appropriate property settings. The **SYM_ENCRYPT** protocol should be configured immediately before the **pbcast.NAKACK2** protocol.

```
<subsystem xmlns="urn:jboss:domain:jgroups:4.0">
  <stacks>
    <stack name="udp">
      <transport type="UDP" socket-binding="jgroups-udp"/>
      <protocol type="PING"/>
      <protocol type="MERGE3"/>
      <protocol type="FD_SOCK" socket-binding="jgroups-udp-fd"/>
      <protocol type="FD_ALL"/>
      <protocol type="VERIFY_SUSPECT"/>
      <protocol type="SYM_ENCRYPT">
        <property name="provider">SunJCE</property>
        <property name="sym_algorithm">AES</property>
        <property name="encrypt_entire_message">true</property>
        <property
name="keystore_name">/path/to/defaultStore.keystore</property>
        <property name="store_password">PASSWORD</property>
        <property name="alias">mykey</property>
      </protocol>
      <protocol type="pbcast.NAKACK2"/>
      <protocol type="UNICAST3"/>
      <protocol type="pbcast.STABLE"/>
      <protocol type="pbcast.GMS"/>
      <protocol type="UFC"/>
      <protocol type="MFC"/>
      <protocol type="FRAG2"/>
    </stack>
  </stacks>
</subsystem>
```

NOTE

Configuring **AUTH** is optional when using **SYM_ENCRYPT**.

Using Asymmetric Encryption

1. Configure the **ASYM_ENCRYPT** protocol in the **jgroups** subsystem.
   In the applicable server configuration file, add the **ASYM_ENCRYPT** protocol with the appropriate property settings. The **ASYM_ENCRYPT** protocol should be configured immediately before the **pbcast.NAKACK2** protocol.

```
<subsystem xmlns="urn:jboss:domain:jgroups:4.0">
  <stacks>
    <stack name="udp">
      <transport type="UDP" socket-binding="jgroups-udp"/>
      <protocol type="PING"/>
      <protocol type="MERGE3"/>
      <protocol type="FD_SOCK" socket-binding="jgroups-udp-fd"/>
      <protocol type="FD_ALL"/>
      <protocol type="VERIFY_SUSPECT"/>
      <protocol type="ASYM_ENCRYPT">
        <property name="encrypt_entire_message">true</property>
        <property name="sym_keylength">128</property>
        <property
name="sym_algorithm">AES/ECB/PKCS5Padding</property>
        <property name="asym_keylength">512</property>
        <property name="asym_algorithm">RSA</property>
      </protocol>
      <protocol type="pbcast.NAKACK2"/>
      <protocol type="UNICAST3"/>
      <protocol type="pbcast.STABLE"/>
      <!-- Configure AUTH protocol here -->
      <protocol type="pbcast.GMS"/>
      <protocol type="UFC"/>
      <protocol type="MFC"/>
      <protocol type="FRAG2"/>
    </stack>
  </stacks>
</subsystem>
```

2. Configure the **AUTH** protocol in the **jgroups** subsystem.
   AUTH is required by **ASYM_ENCRYPT**. See the Configuring Authentication section for instructions.

22.2.7. Configure JGroups Thread Pools

The **jgroups** subsystem contains the **default**, **internal**, **oob**, and **timer** thread pools. These pools can be configured for any JGroups stack.

The following table lists the attributes you can configure for each thread pool and the default value for each.

Thread Pool Name	keepalive-time	max-threads	min-threads	queue-length
default	60000L	300	20	100
internal	60000L	4	2	100
oob	60000L	300	20	0
timer	5000L	4	2	500

Use the following syntax to configure a JGroups thread pool using the management CLI.

```
/subsystem=jgroups/stack=STACK_TYPE/transport=TRANSPORT_TYPE/thread-
pool=THREAD_POOL_NAME:write-attribute(name=ATTRIBUTE_NAME,
value=ATTRIBUTE_VALUE)
```

The following is an example of the management CLI command to set the **max-threads** value to **500** in the **default** thread pool for the **udp** stack.

```
/subsystem=jgroups/stack=udp/transport=UDP/thread-pool=default:write-
attribute(name="max-threads", value="500")
```

22.2.8. Configure JGroups Send and Receive Buffers

Resolving Buffer Size Warnings
By default, JGroups is configured with certain send and receive buffer values. However, your operating system may limit the available buffer sizes and JBoss EAP may not be able to use its configured buffer values. In this situation you will see warnings in the JBoss EAP logs similar to the following:

```
WARNING [org.jgroups.protocols.UDP] (ServerService Thread Pool -- 68)
JGRP000015: the send buffer of socket DatagramSocket was set to 640KB, but
the OS only allocated 212.99KB.
This might lead to performance problems. Please set your max send buffer
in the OS correctly (e.g. net.core.wmem_max on Linux)
WARNING [org.jgroups.protocols.UDP] (ServerService Thread Pool -- 68)
JGRP000015: the receive buffer of socket DatagramSocket was set to 20MB,
but the OS only allocated 212.99KB.
This might lead to performance problems. Please set your max receive
buffer in the OS correctly (e.g. net.core.rmem_max on Linux)
```

To resolve this, consult your operating system documentation for instructions on how to increase the buffer size. For Red Hat Enterprise Linux systems, edit **/etc/sysctl.conf** as the root user to configure maximum values for buffer sizes that will survive system restarts. For example:

```
# Allow a 25MB UDP receive buffer for JGroups
net.core.rmem_max = 26214400
# Allow a 1MB UDP send buffer for JGroups
net.core.wmem_max = 1048576
```

After modifying **/etc/sysctl.conf**, run **sysctl -p** for the changes to take effect.

Configuring JGroups Buffer Sizes
You can configure the JGroups buffer sizes that JBoss EAP uses by setting the following transport properties on the UDP and TCP JGroups stacks.

UDP Stack

- **ucast_recv_buf_size**
- **ucast_send_buf_size**
- **mcast_recv_buf_size**
- **mcast_send_buf_size**

TCP Stack

- **recv_buf_size**
- **send_buf_size**

JGroups buffer sizes can be configured using the management console or the management CLI.

Use the following syntax to set a JGroups buffer size property using the management CLI.

```
/subsystem=jgroups/stack=STACK_NAME/transport=TRANSPORT/property=PROPERTY_
NAME:add(value=BUFFER_SIZE)
```

The following is an example management CLI command to set the **recv_buf_size** property to **20000000** on the **tcp** stack.

```
/subsystem=jgroups/stack=tcp/transport=TRANSPORT/property=recv_buf_size:ad
d(value=20000000)
```

JGroups buffer sizes can also be configured using the management console by navigating to the **JGroups** subsystem from the **Configuration** tab, viewing the relevant stack, selecting **Transport**, and selecting the transport **Properties** tab.

22.2.9. JGroups Troubleshooting

22.2.9.1. Nodes Do Not Form a Cluster

Make sure your machine is set up correctly for IP multicast. There are two test programs that ship with JBoss EAP that can be used to test IP multicast: **McastReceiverTest** and **McastSenderTest**.

In a terminal, start **McastReceiverTest**.

```
$ java -cp EAP_HOME/bin/client/jboss-client.jar
org.jgroups.tests.McastReceiverTest -mcast_addr 230.11.11.11 -port 5555
```

Then in another terminal window, start **McastSenderTest**.

```
$ java -cp EAP_HOME/bin/client/jboss-client.jar
org.jgroups.tests.McastSenderTest -mcast_addr 230.11.11.11 -port 5555
```

If you want to bind to a specific network interface card (NIC), use **-bind_addr YOUR_BIND_ADDRESS**, where **YOUR_BIND_ADDRESS** is the IP address of the NIC to which you want to bind. Use this parameter in both the sender and the receiver.

When you type in the **McastSenderTest** terminal window, you should see the output in the **McastReceiverTest** window. If you do not, try the following steps.

- Increase the time-to-live for multicast packets by adding **-ttl VALUE** to the sender command. The default used by this test program is **32** and the **VALUE** must be no greater than **255**.
- If the machines have multiple interfaces, verify that you are using the correct interface.

- Contact a system administrator to make sure that multicast will work on the interface you have chosen.

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.

22.2.9.2. Causes of Missing Heartbeats in Failure Detection

Sometimes a cluster member is suspected by failure detection (FD) because a heartbeat acknowledgement has not been received for some time **T**, which is defined by `timeout` and `max_tries`.

For an example cluster of nodes A, B, C, and D, where A pings B, B pings C, C pings D, and D pings A, C can be suspected for any of the following reasons:

- B or C are running at 100% CPU for more than **T** seconds. So even if C sends a heartbeat acknowledgement to B, B may not be able to process it because it is at 100% CPU usage.
- B or C are garbage collecting, which results in the same situation as above.
- A combination of the two 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, and TCP packets last.
- B or C are processing a callback. For example, if C received a remote method call over its channel that takes **T** + 1 seconds to process, during this time, C will not process any other messages, including heartbeats. Therefore, B will not receive the heartbeat acknowledgement and will suspect C.

22.3. INFINISPAN

22.3.1. About Infinispan

Infinispan is a Java data grid platform that provides a JSR-107-compatible cache interface for managing cached data. For more information about Infinispan functionality and configuration options see the Infinispan Documentation.

The `infinispan` subsystem provides caching support for JBoss EAP. It allows you to configure and view runtime metrics for named cache containers and caches.

When using a configuration that provides high availability capabilities, such as the *ha* or *full-ha* profile in a managed domain, or the `standalone-ha.xml` or `standalone-full-ha.xml` configuration file for a standalone server, the `infinispan` subsystem provides caching, state replication, and state distribution support. In non-high-availability configurations, the `infinispan` subsystem provides local caching support.

IMPORTANT

Infinispan is delivered as a private module in JBoss EAP to provide the caching capabilities of JBoss EAP. Infinispan is not supported for direct use by applications.

22.3.2. Cache Containers

A cache container is a repository for the caches used by a subsystem. Each cache container defines a default cache to be used.

JBoss EAP 7 defines the following default Infinispan cache containers:

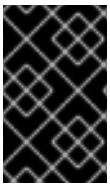
- **server** for singleton caching
- **web** for web session clustering
- **ejb** for stateful session bean clustering
- **hibernate** for entity caching

Example: Default Infinispan Configuration

```
<subsystem xmlns="urn:jboss:domain:infinispan:4.0">
  <cache-container name="server" aliases="singleton cluster" default-
cache="default" module="org.wildfly.clustering.server">
    <transport lock-timeout="60000"/>
    <replicated-cache name="default" mode="SYNC">
      <transaction mode="BATCH"/>
    </replicated-cache>
  </cache-container>
  <cache-container name="web" default-cache="dist"
module="org.wildfly.clustering.web.infinispan">
    <transport lock-timeout="60000"/>
    <distributed-cache name="dist" mode="ASYNC" l1-lifespan="0"
owners="2">
      <locking isolation="REPEATABLE_READ"/>
      <transaction mode="BATCH"/>
      <file-store/>
    </distributed-cache>
  </cache-container>
  <cache-container name="ejb" aliases="sfsb" default-cache="dist"
module="org.wildfly.clustering.ejb.infinispan">
    <transport lock-timeout="60000"/>
    <distributed-cache name="dist" mode="ASYNC" l1-lifespan="0"
owners="2">
      <locking isolation="REPEATABLE_READ"/>
      <transaction mode="BATCH"/>
      <file-store/>
    </distributed-cache>
  </cache-container>
  <cache-container name="hibernate" default-cache="local-query"
module="org.hibernate.infinispan">
    <transport lock-timeout="60000"/>
    <local-cache name="local-query">
      <eviction strategy="LRU" max-entries="10000"/>
      <expiration max-idle="100000"/>
    </local-cache>
    <invalidation-cache name="entity" mode="SYNC">
      <transaction mode="NON_XA"/>
      <eviction strategy="LRU" max-entries="10000"/>
      <expiration max-idle="100000"/>
    </invalidation-cache>
```

```
    <replicated-cache name="timestamps" mode="ASYNC"/>
  </cache-container>
</subsystem>
```

Note the default cache defined in each cache container. For example, the **web** cache container defines the **dist** distributed cache as the default. The **dist** cache will therefore be used when clustering web sessions.



IMPORTANT

You can add additional caches and cache containers, for example, for HTTP sessions, stateful session beans, or singleton services or deployments. It is not supported to use these caches directly by user applications.

22.3.2.1. Configure Cache Containers

Cache containers and cache attributes can be configured using the management console or management CLI.

WARNING

You should avoid changing cache or cache container names, as other components in the configuration may reference them.

Configure Caches Using the Management Console
Once you navigate to the **Infinispan** subsystem from the **Configuration** tab in the management console, you can configure caches and cache containers. In a managed domain, make sure to select the appropriate profile to configure.

- Add a cache container.
  Click the **Add** button next to the **Cache Container** heading and enter the settings for the new cache container.

- Update cache container settings.
  Choose the appropriate cache container and select **Container Settings** from the drop down. Configure the cache container settings as necessary.

- Update cache container transport settings.
  Choose the appropriate cache container and select **Transport Settings** from the drop down. Configure the cache container transport settings as necessary.

- Configure caches.
  Choose the appropriate cache container and select **View**. From the appropriate cache tab (for example, **Replicated Caches**), you can add, update, and remove caches.

Configure Caches Using the Management CLI
You can configure caches and cache containers using the management CLI. In a managed domain, you must specify the profile to update by preceding these commands with **/profile=PROFILE_NAME**.

- Add a cache container.

```
/subsystem=infinispan/cache-container=CACHE_CONTAINER:add
```

- Add a replicated cache.

```
/subsystem=infinispan/cache-container=CACHE_CONTAINER/replicated-
cache=CACHE:add(mode=MODE)
```

- Set the default cache for a cache container.

```
/subsystem=infinispan/cache-container=CACHE_CONTAINER:write-
attribute(name=default-cache,value=CACHE)
```

- Configure batching for a replicated cache.

```
/subsystem=infinispan/cache-container=CACHE_CONTAINER/replicated-
cache=CACHE/component=transaction:write-
attribute(name=mode,value=BATCH)
```

Change the Default EJB Cache Container
You can use cache containers in the **ejb3** subsystem as described below:

- To support passivation of EJB session beans, you can use the **ejb** cache container defined in the **infinispan** subsystem to store the sessions.

- For remote EJB clients connecting to a clustered deployment on a server, you must provide the cluster topology information to these clients, so that they can fail over to other nodes in the cluster if the node they are interacting with fails.

If you are changing or renaming the default cache container, named **ejb**, which supports passivation and the provision of topology information, you must add the **cache-container** attribute to the **passivation-stores** element and the **cluster** attribute to the **remote** element, as shown in the example below. If you are just adding a new cache for your own use, you need not make those changes.

```
<subsystem xmlns="urn:jboss:domain:ejb3:4.0">
    <passivation-stores>
        <passivation-store name="infinispan" cache-container="ejb-cltest"
max-size="10000"/>
    </passivation-stores>

    <remote cluster="ejb-cltest" connector-ref="http-remoting-connector"
thread-pool-name="default"/>
</subsystem>

<subsystem xmlns="urn:jboss:domain:infinispan:4.0">
    ...
    <cache-container name="ejb-cltest" aliases="sfsb" default-cache="dist"
module="org.wildfly.clustering.ejb.infinispan">
</subsystem>
```

22.3.3. Clustering Modes

Clustering can be configured in two different ways in JBoss EAP using Infinispan. The best 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.

Cache Modes

Replicated

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.

Distributed

Distributed mode allows Infinispan to scale the cluster linearly. Distributed mode uses a consistent hash algorithm to determine where in a cluster a new node should be placed. The number of copies (owners) 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.

You should consider using distributed mode as a caching strategy when the cluster size exceeds 6-8 nodes. With distributed mode, data is distributed to only a subset of nodes within the cluster, as opposed to all nodes.

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.

Synchronous replication

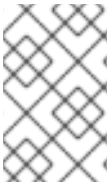
With synchronous replication, the thread that handles the user request is blocked until replication has been successful. When the replication is successful, a response is sent back to the client, and only then is the thread is released. Synchronous replication has an impact on network traffic because 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

With asynchronous replication, Infinispan uses a thread pool to carry out replication in the background. The sender does not wait for replies from other nodes in the cluster. However, cache reads for the same session will block until the previous replication completes so that stale data is not read. Replication is triggered either on a time basis or by queue size. Failed replication attempts are written to a log, not notified in real time.

22.3.3.1. Configure the Cache Mode

You can change the default cache using the management CLI.

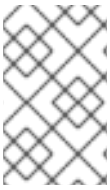


NOTE

This section shows instructions specific to configuring the web session cache, which defaults to distributed mode. The steps and management CLI commands can easily be adjusted to apply to other cache containers.

Change to Replicated Cache Mode

The default JBoss EAP 7 configuration for the web session cache does not include a **repl** replicated cache. This cache must first be added.



NOTE

The below management CLI commands are for a standalone server. When running in a managed domain, you must specify which profile to update by preceding the **/subsystem=infinispan** commands with **/profile=PROFILE_NAME**.

1. Add the **repl** replicated cache.

```
/subsystem=infinispan/cache-container=web/replicated-
cache=repl:add(mode=ASYNC)
/subsystem=infinispan/cache-container=web/replicated-
cache=repl/component=transaction:write-
attribute(name=mode,value=BATCH)
/subsystem=infinispan/cache-container=web/replicated-
cache=repl/component=locking:write-attribute(name=isolation,
value=REPEATABLE_READ)
/subsystem=infinispan/cache-container=web/replicated-
cache=repl/store=file:add
```

2. Change the default cache to the **repl** replicated cache.

```
/subsystem=infinispan/cache-container=web:write-
attribute(name=default-cache,value=repl)
```

3. Reload the server.

```
reload
```

Change to Distributed Cache Mode

The default JBoss EAP 7 configuration for the web session cache already includes a **dist** distributed cache.

NOTE

The below management CLI commands are for a standalone server. When running in a managed domain, you must specify which profile to update by preceding the **/subsystem=infinispan** commands with **/profile=PROFILE_NAME**.

1. Change the default cache to the **dist** distributed cache.

```
/subsystem=infinispan/cache-container=web:write-
attribute(name=default-cache,value=dist)
```

2. Set the number of owners for the distributed cache. The following command sets **5** owners. The default is **2**.

```
/subsystem=infinispan/cache-container=web/distributed-
cache=dist/:write-attribute(name=owners,value=5)
```

3. Reload the server.

```
reload
```

22.3.3.2. Cache Strategy Performance

When using a **SYNC** caching strategy, the cost of replication is easy to measure and directly seen in response times since the request does not complete until the replication completes.

Although it seems that the **ASYNC** caching strategy should result in lower response times than the **SYNC** caching strategy, this is only true under the right conditions. The **ASYNC** caching strategy is more difficult to measure, but it can provide better performance than the **SYNC** strategy when the duration between requests is long enough for the cache operation to complete. This is because the cost of replication is not immediately seen in response times.

If requests for the same session are made too quickly, the cost of replication for the previous request is shifted to the front of the subsequent request since it must wait for the replication from the previous request to complete. For rapidly fired requests where a subsequent request is sent immediately after a response is received, the **ASYNC** caching strategy will perform worse than the **SYNC** caching strategy. Consequently, there is a threshold for the period of time between requests for the same session where the **SYNC** caching strategy will actually perform better than the **ASYNC** caching strategy. In real world usage, requests for the same session are not normally received in rapid succession. Instead, there is typically a period of time in the order of a few seconds or more between the requests. In this case, the **ASYNC** caching strategy is a sensible default and provides the fastest response times.

22.3.4. Configure Infinispan Thread Pools

The `infinispan` subsystem contains the `async-operations`, `expiration`, `listener`, `persistence`, `remote-command`, `state-transfer`, and `transport` thread pools. These pools can be configured for any Infinispan cache container.

The following table lists the attributes you can configure for each thread pool in the `infinispan` subsystem and the default value for each.

Thread Pool Name	keepalive-time	max-threads	min-threads	queue-length
async-operations	60000L	25	25	1000
expiration	60000L	1	N/A	N/A
listener	60000L	1	1	100000
persistence	60000L	4	1	0
remote-command	60000L	200	1	0
state-transfer	60000L	60	1	0
transport	60000L	25	25	100000

Use the following syntax to configure an Infinispan thread pool using the management CLI.

```
/subsystem=infinispan/cache-container=CACHE_CONTAINER_NAME/thread-
pool=THREAD_POOL_NAME:write-attribute(name=ATTRIBUTE_NAME,
value=ATTRIBUTE_VALUE)
```

The following is an example of the management CLI command to set the **max-threads** value to **10** in the **persistence** thread pool for the **server** cache container.

```
/subsystem=infinispan/cache-container=server/thread-
pool=persistence:write-attribute(name="max-threads", value="10")
```

22.3.5. Infinispan Statistics

Runtime statistics about Infinispan caches and cache containers can be enabled for monitoring purposes. Statistics collection is not enabled by default for performance reasons.

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.

22.3.5.1. Enable Infinispan Statistics

WARNING

Enabling Infinispan statistics may have a negative impact on the performance of the **infinispan** subsystem. Statistics should be enabled only when required.

You can enable or disable the collection of Infinispan statistics using the management console or the management CLI. From the management console, navigate to the **Infinispan** subsystem from the **Configuration** tab, select the appropriate cache or cache container, and edit the **Statistics enabled** attribute. Use the below commands to enable statistics using the management CLI.

Enable statistics collection for a cache container. A server reload will be required.

```
/subsystem=infinispan/cache-container=CACHE_CONTAINER:write-
attribute(name=statistics-enabled,value=true)
```

Enable statistics collection for a cache. A server reload will be required.

```
/subsystem=infinispan/cache-
container=CACHE_CONTAINER/CACHE_TYPE=CACHE:write-
attribute(name=statistics-enabled,value=true)
```

NOTE

You can use the following command to undefine the **`statistics-enabled`** attribute of a cache so that it will inherit the settings of its cache container's **`statistics-enabled`** attribute.

```
/subsystem=infinispan/cache-
container=CACHE_CONTAINER/CACHE_TYPE=CACHE:undefine-
attribute(name=statistics-enabled)
```

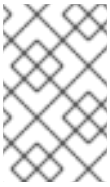
22.3.6. Infinispan Partition Handling

An *Infinispan cluster* is built out of several nodes where data is stored. To prevent data loss if multiple nodes fail, Infinispan copies the same data over multiple nodes. This level of data redundancy is configured using the **`owners`** attribute. As long as fewer than the configured number of nodes crash simultaneously, Infinispan will have a copy of the data available.

However, there are potential catastrophic situations that could occur when too many nodes disappear from the cluster:

Split brain

This splits the cluster in two or more partitions, or sub-clusters, that operate independently. In these circumstances, multiple clients reading and writing from different partitions see different versions of the same cache entry, which for many applications is problematic.



NOTE

There are ways to alleviate the possibility for the split brain to happen, such as redundant networks or IP bonding. However, these only reduce the window of time for the problem to occur.

Multiple nodes crash in sequence

If multiple nodes, specifically the number of owners, crash in rapid sequence and Infinispan does not have the time to properly rebalance its state between crashes, the result is partial data loss.

The goal is to avoid situations in which incorrect data is returned to the user as a result of either split brain or multiple nodes crashing in rapid sequence.

22.3.6.1. Split Brain

In a split brain situation, each network partition will install its own JGroups view, removing the nodes from the other partitions. We do not have a direct way to determine whether the cluster has been split into two or more partitions, since the partitions are unaware of each other. Instead, we assume the cluster has split when one or more nodes disappear from the JGroups cluster without sending an explicit leave message.

With partition handling disabled, each such partition would continue to function as an independent cluster. Each partition may only see a part of the data, and each partition could write conflicting updates in the cache.

With partition handling enabled, if we detect a split, each partition does not start a rebalance immediately, but first checks whether it should enter degraded mode instead:

- If at least one segment has lost all its owners, meaning that at least the number of owners specified has left since the last rebalance ended, the partition enters degraded mode.
- If the partition does not contain a simple majority of the nodes (floor(numNodes/2) + 1) in the *latest stable topology*, the partition also enters degraded mode.
- Otherwise, the partition keeps functioning normally and starts a rebalance.

The *stable topology* is updated every time a rebalance operation ends and the coordinator determines that another rebalance is not necessary. These rules ensure that at most, one partition stays in available mode, and the other partitions enter degraded mode.

When a partition is in degraded mode, it only allows access to the keys that are wholly owned:

- Requests (reads and writes) for entries that have all the copies on nodes within this partition are honored.
- Requests for entries that are partially or totally owned by nodes that have disappeared are rejected with an **`AvailabilityException`**.

This guarantees that partitions cannot write different values for the same key (cache is consistent), and also that one partition can not read keys that have been updated in the other partitions (no stale data).

NOTE

Two partitions could start up isolated, and as long as they do not merge, they can read and write inconsistent data. In the future, we may allow custom availability strategies (e.g. check that a certain node is part of the cluster, or check that an external machine is accessible) that could handle that situation as well.

22.3.6.2. Configuring Partition Handling

Currently the partition handling is disabled by default. Use the following management CLI command to enable partition handling:

```
/subsystem=infinispan/cache-container=web/distributed-
cache=dist/component=partition-handling:write-attribute(name=enabled,
value=true)
```

22.3.7. Externalize HTTP Sessions to JBoss Data Grid

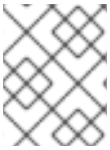
NOTE

You will need a Red Hat JBoss Data Grid subscription to use this functionality.

Red Hat JBoss Data Grid can be used as an external cache container for application-specific data in JBoss EAP, such as HTTP sessions. This allows scaling of the data layer independent of the application, and enables different JBoss EAP clusters, which 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 following example shows how to externalize HTTP sessions. It applies to both standalone instances of JBoss EAP as well as managed domains. However, in a managed domain, 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 JBoss 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.

To externalize HTTP sessions:

1. Define the location of the remote Red Hat JBoss Data Grid server by adding the networking information to the **`socket-binding-group`**.

   Example Adding Remote Socket Bindings

   ```
   /socket-binding-group=standard-sockets/remote-destination-outbound-
   socket-binding=remote-jdg-server1:add(host=JDGHostName1, port=11222)

   /socket-binding-group=standard-sockets/remote-destination-outbound-
   socket-binding=remote-jdg-server2:add(host=JDGHostName2, port=11222)
   ```

   Resulting XML

   ```
   <socket-binding-group name="standard-sockets" ... >
     ...
     <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>
   ```

   NOTE

   You will need a remote socket binding configured for each Red Hat JBoss Data Grid server.

2. Ensure the remote cache containers are defined in JBoss 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.
   If you are running in a managed domain, precede these commands with **`/profile=PROFILE_NAME`**.

   Example Adding a Remote Cache Container

   ```
   /subsystem=infinispan/cache-container=web/invalidation-
   cache=jdg:add(mode=SYNC)

   /subsystem=infinispan/cache-container=web/invalidation-
   cache=jdg/component=locking:write-
   attribute(name=isolation,value=REPEATABLE_READ)
   ```

```
/subsystem=infinispan/cache-container=web/invalidation-
cache=jdg/component=transaction:write-
attribute(name=mode,value=BATCH)

/subsystem=infinispan/cache-container=web/invalidation-
cache=jdg/store=remote:add(remote-servers=["remote-jdg-
server1","remote-jdg-server2"], cache=default, socket-timeout=60000,
passivation=false, purge=false, shared=true)
```

Resulting XML

```
<subsystem xmlns="urn:jboss:domain:infinispan:4.0">
  ...
  <cache-container name="web" default-cache="dist"
module="org.wildfly.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 cache="default" socket-timeout="60000" remote-
servers="remote-jdg-server1 remote-jdg-server2" passivation="false"
purge="false" shared="true"/>
    </invalidation-cache>
    ...
  </cache-container>
</subsystem>
```

3. Add 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 appropriate cache located in this container.

   Example `jboss-web.xml` File

```
<?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>
```

22.4. CONFIGURING JBOSS EAP AS A FRONT-END LOAD BALANCER

You can configure JBoss EAP and the **`undertow`** subsystem to act as a front-end load balancer to proxy requests to back-end JBoss EAP servers. Since Undertow makes use of asynchronous IO, the IO thread that is responsible for the connection is the only thread that is involved in the request. That same thread is also used for the connection made to the back-end server.

You can use the following protocols:

- HTTP over plain text (**http**), supporting HTTP/1 and HTTP/2 (**h2c**)

NOTE

HTTP/2 is provided as technology preview only.

- HTTP over secured connection (**https**), supporting HTTP/1 and HTTP/2 (**h2**)

NOTE

HTTP/2 is provided as technology preview only.

- AJP (**ajp**)

You can either define a static load balancer and specify the back-end hosts in your configuration, or use the mod_cluster front end to dynamically update the hosts.

22.4.1. Configure Undertow as a Load Balancer Using mod_cluster

You can use the built-in mod_cluster front-end load balancer to load balance other JBoss EAP instances.

This procedure assumes that you are running in a managed domain and already have the following configured:

- A JBoss EAP server that will act as the load balancer.
  - This server uses the **default** profile, which is bound to the **standard-sockets** socket binding group.
- Two JBoss EAP servers, which will act as the back-end servers.
  - These servers are running in a cluster and use the **ha** profile, which is bound to the **ha-sockets** socket binding group.
- The distributable application to be load balanced deployed to the back-end servers.

Configure the mod_cluster Front-end Load Balancer
The below steps load balance servers in a managed domain, but they can be adjusted to apply to a set of standalone servers. Be sure to update the management CLI command values to suit your environment.

1. Set the mod_cluster advertise security key.
   Adding the advertise security key allows the load balancer and servers to authenticate during discovery.

   Use the following management CLI command to set the mod_cluster advertise security key.

   ```
   /profile=ha/subsystem=modcluster/mod-cluster-
   config=configuration:write-attribute(name=advertise-security-key,
   value=mypassword)
   ```

2. Create a socket binding with the multicast address and port for mod_cluster.

You need to create a socket configuration for mod_cluster to use for discovery and communication with the servers that it is going to load balance.

Use the following management CLI command to add a **modcluster** socket binding with the appropriate multicast address and port configured.

```
/socket-binding-group=standard-sockets/socket-
binding=modcluster:add(multicast-port=23364, multicast-
address=224.0.1.105)
```

3. Include the mod_cluster load balancer.
Once you have the advertise security key and socket binding set up, you need to add the mod_cluster filter to Undertow for the load balancer instance of JBoss EAP.

Use the following management CLI command to add the mod_cluster filter.

```
/profile=default/subsystem=undertow/configuration=filter/mod-
cluster=modcluster:add(management-socket-binding=http, advertise-
socket-binding=modcluster, security-key=mypassword)
```

Use the following management CLI command to bind the mod_cluster filter to the default host.

```
/profile=default/subsystem=undertow/server=default-
server/host=default-host/filter-ref=modcluster:add
```

IMPORTANT

It is recommended that the management and advertise socket bindings used by mod_cluster only be exposed to the internal network, not a public IP address.

The load balancer JBoss EAP server can now load balance the two back-end JBoss EAP servers.

22.4.2. Configure Undertow as a Static Load Balancer

To configure a static load balancer with Undertow, you need to configure a proxy handler in the **undertow** subsystem. To configure a proxy handler in Undertow, you need to do the following on your JBoss EAP instance that will serve as your static load balancer:

1. Add a reverse proxy handler
2. Define the outbound socket bindings for each remote host
3. Add each remote host to the reverse proxy handler
4. Add the reverse proxy location

The following example shows how to configure a JBoss EAP instance to be a static load balancer. The JBoss EAP instance is located at **lb.example.com** and will load balance between two additional servers: **server1.example.com** and **server2.example.com**. The load balancer will reverse-proxy to the location **/app** and will the AJP protocol.

1. To add a reverse proxy handler:

```
/subsystem=undertow/configuration=handler/reverse-proxy=my-
```

```
handler:add
```

2. To define the outbound socket bindings for each remote host:

```
/socket-binding-group=standard-sockets/remote-destination-outbound-
socket-binding=remote-host1/:add(host=server1.example.com,
port=8009)

/socket-binding-group=standard-sockets/remote-destination-outbound-
socket-binding=remote-host2/:add(host=server2.example.com,
port=8009)
```

3. To add each remote host to the reverse proxy handler:

```
/subsystem=undertow/configuration=handler/reverse-proxy=my-
handler/host=host1:add(outbound-socket-binding=remote-host1,
scheme=ajp, instance-id=myroute, path=/test)

/subsystem=undertow/configuration=handler/reverse-proxy=my-
handler/host=host2:add(outbound-socket-binding=remote-host2,
scheme=ajp, instance-id=myroute, path=/test)
```

4. To add the reverse proxy location:

```
/subsystem=undertow/server=default-server/host=default-
host/location=\/app:add(handler=my-handler)
```

When accessing **`lb.example.com:8080/app`**, you will now see the content proxied from **`server1.example.com`** and **`server2.example.com`**.

22.5. USING AN EXTERNAL WEB SERVER AS A PROXY SERVER

JBoss EAP can accept requests from an external web server using the supported HTTP, HTTPS, or AJP protocol, depending on the external web server configuration.

See Overview of HTTP Connectors for details on the supported HTTP connectors for each web server. Once you have decided which web server and HTTP connector to use, see the appropriate section for information on configuring your connector:

- See the mod_cluster, mod_jk, or mod_proxy section for Apache HTTP Server.
- See the ISAPI connector section for Microsoft IIS.
- See the NSAPI connector section for Oracle iPlanet Web Server.

For the most current information about supported configurations for HTTP connectors, see JBoss EAP supported configurations.

You also will need to make sure that JBoss EAP is configured to accept requests from external web servers.

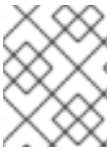
22.5.1. Overview of HTTP Connectors

JBoss EAP has the ability to use load-balancing and clustering mechanisms built into external web

servers, such as Apache HTTP Server, Microsoft IIS, and Oracle iPlanet as well as through Undertow. JBoss EAP communicates with the web servers using a connector. These connectors are configured within the **undertow** subsystem of JBoss EAP.

The web servers include software modules which control the way that HTTP requests are routed to JBoss EAP 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 nodes, to move work loads to alternate servers in case of a failure event, or both.

JBoss EAP supports several different connectors. The one you choose depends on the web server in use and the functionality you need. See the tables below for comparisons of the supported configurations and features of the various HTTP connectors that are compatible with JBoss EAP.



NOTE

See Configure Undertow as a Load Balancer Using mod_cluster for using JBoss EAP 7 as a multi-platform load balancer.

For the most current information about supported configurations for HTTP connectors, see JBoss EAP supported configurations.

Table 22.1. HTTP Connector Supported Configurations

Connector	Web Server	Supported Operating Systems	Supported Protocols
mod_cluster	Red Hat JBoss Core Services Apache HTTP Server, Red Hat JBoss Web Server Apache HTTP Server, JBoss EAP (Undertow)	Red Hat Enterprise Linux, Microsoft Windows Server, Oracle Solaris	HTTP, HTTPS, AJP, WebSocket
mod_jk	Red Hat JBoss Core Services Apache HTTP Server, Red Hat JBoss Web Server Apache HTTP Server	Red Hat Enterprise Linux, Microsoft Windows Server, Oracle Solaris	AJP
mod_proxy	Red Hat JBoss Core Services Apache HTTP Server, Red Hat JBoss Web Server Apache HTTP Server	Red Hat Enterprise Linux, Microsoft Windows Server, Oracle Solaris	HTTP, HTTPS, AJP
ISAPI connector	Microsoft IIS	Microsoft Windows Server	AJP
NSAPI connector	Oracle iPlanet Web Server	Oracle Solaris	AJP

Table 22.2. HTTP Connector Features

Connector	Supports Sticky Sessions	Adapts to Deployment Status
mod_cluster	Yes	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.
mod_jk	Yes	No. Directs client requests to the container as long as the container is available, regardless of application status.
mod_proxy	Yes	No. Directs client requests to the container as long as the container is available, regardless of application status.
ISAPI connector	Yes	No. Directs client requests to the container as long as the container is available, regardless of application status.
NSAPI connector	Yes	No. Directs client requests to the container as long as the container is available, regardless of application status.

22.5.2. Apache HTTP Server

A standalone Apache HTTP Server bundle is now available as a separate download with Red Hat JBoss Core Services. This simplifies installation and configuration, and allows for a more consistent update experience.

22.5.2.1. Installing Apache HTTP Server

For information on installing Apache HTTP Server, see the JBoss Core Services *Apache HTTP Server Installation Guide*.

22.5.3. Accepting Requests from External Web Servers

JBoss EAP does not require any special configuration to begin accepting requests from a proxy server as long as the correct protocol handler, for example AJP, HTTP, or HTTPS, is configured.

If the proxy server uses mod_jk, mod_proxy, ISAPI, or NSAPI, it sends requests to JBoss EAP and JBoss EAP simply provides a response. With mod_cluster, you must also configure your network to allow JBoss EAP to send information, such as its current load, application lifecycle events, and health status, to the proxy server to help it determine where to route requests. For more information about configuring a mod_cluster proxy server, see The mod_cluster HTTP Connector .

Update JBoss EAP Configuration
In the following procedure, substitute the protocols and ports in the examples with the ones you need to configure.

1. Configure the `instance-id` attribute of Undertow.
   The external web server identifies the JBoss EAP instance in its connector configuration using the `instance-id`. Use the following management CLI command to set the `instance-id` attribute in Undertow.

```
/subsystem=undertow:write-attribute(name=instance-id,value=node1)
```

In the above example, the external web server identifies the current JBoss EAP instance as **node1**.

2. Add the necessary listeners to Undertow.
In order for an external web server to be able to connect to JBoss EAP, Undertow needs a listener. Each protocol needs its own listener, which is tied to a socket binding.

NOTE

Depending on your desired protocol and port configuration, this step may not be necessary. An HTTP listener is configured in all default JBoss EAP configurations, and an AJP listener is configured if you are using the *ha* or *full-ha* profile.

You can check whether the required listeners are already configured by reading the default server configuration:

```
/subsystem=undertow/server=default-server:read-resource
```

To add a listener to Undertow, 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 management CLI command adds an **ajp** socket binding, bound to port **8009**, to the **standard-sockets** socket binding group

```
/socket-binding-group=standard-sockets/socket-
binding=ajp:add(port=8009)
```

The following management CLI command adds an **ajp** listener to Undertow, using the **ajp** socket binding.

```
/subsystem=undertow/server=default-server/ajp-
listener=ajp:add(socket-binding=ajp)
```

22.6. THE MOD_CLUSTER HTTP CONNECTOR

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 mod_cluster connector has several advantages over other connectors.

- The mod_cluster Management Protocol (MCMP) is an additional connection between the JBoss EAP servers and the Apache HTTP Server with the mod_cluster module enabled. It is used by the JBoss EAP 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 servers to join the load-balancing arrangement without manual configuration.
- JBoss EAP 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 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.

For more details on the specific configuration options of the **`modcluster`** subsystem, see the ModCluster Subsystem Attributes.

22.6.1. Configure mod_cluster in Apache HTTP Server

The mod_cluster modules are already included when installing JBoss Core Services Apache HTTP Server or using JBoss Web Server and are loaded by default.

NOTE

Apache HTTP Server is no longer distributed with JBoss Web Server as of version 3.1.0.

See the steps below to configure the mod_cluster module to suit your environment.

NOTE

Red Hat customers can also use the Load Balancer Configuration Tool on the Red Hat Customer Portal to quickly generate optimal configuration templates for mod_cluster and other connectors. Note that you must be logged in to access this tool.

Configure mod_cluster
Apache HTTP Server already contains a mod_cluster configuration file, **`mod_cluster.conf`**, that loads the mod_cluster modules and provides basic configuration. The IP address, port, and other settings in this file, shown below, can be configured to suit your needs.

```
# mod_proxy_balancer should be disabled when mod_cluster is used
LoadModule proxy_cluster_module modules/mod_proxy_cluster.so
LoadModule cluster_slotmem_module modules/mod_cluster_slotmem.so
LoadModule manager_module modules/mod_manager.so
LoadModule advertise_module modules/mod_advertise.so

MemManagerFile cache/mod_cluster

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

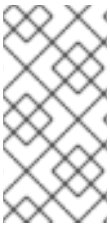
The Apache HTTP Server server is configured as a load balancer and can work with the **modcluster** subsystem running on JBoss EAP. You must configure a mod_cluster worker node to make JBoss EAP aware of mod_cluster.

If you want to disable advertising for mod_cluster and configure a static proxy list instead, see Disable Advertisement for mod_cluster. For more information on the available mod_cluster configuration options in Apache HTTP Server, see the Apache HTTP Server mod_cluster Directives

For more details on configuring mod_cluster, see the Configure Load Balancing Using Apache HTTP Server and mod_cluster section of the JBoss Web Server *HTTP Connectors and Load Balancing Guide*

22.6.2. Disable Advertising for mod_cluster

By default, the **modcluster** subsystem's balancer uses multicast UDP to advertise its availability to the background workers. You can disable advertising and to use a proxy list instead using the following procedure.



NOTE

The management CLI commands in the following procedure assume that you are using the **full-ha** profile in a managed domain. If you are using a profile other than **full-ha**, use the appropriate profile name in the command. If you are running a standalone server, remove the **/profile=full-ha** completely.

1. Modify the Apache HTTP Server configuration.
   Edit the **httpd.conf** Apache HTTP Server configuration file. Make the following updates to the virtual host that listens for MCPM requests, using the **EnableMCPMReceive** directive.

   a. Add the directive to disable server advertisement.
      Set the **ServerAdvertise** directive to **Off** to disable server advertisement.

      ```
      ServerAdvertise Off
      ```

   b. Disable the advertise frequency.
      If your configuration specifies the **AdvertiseFrequency** parameter, comment it out using a # character.

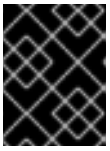
      ```
      # AdvertiseFrequency 5
      ```

   c. Enable the ability to receive MCPM messages.
      Ensure that the **EnableMCPMReceive** directive exists, to allow the web server to receive MCPM messages from the worker nodes.

      ```
      EnableMCPMReceive
      ```

2. Disable advertising in the JBoss EAP **modcluster** subsystem.
   Use the following management CLI command to disable advertising.

   ```
   /profile=full-ha/subsystem=modcluster/mod-cluster-
   config=configuration/:write-attribute(name=advertise,value=false)
   ```



IMPORTANT

Be sure to continue to the next step to provide the list of proxies. Advertising will not be disabled if the list of proxies is empty.

3. Provide a list of proxies in the JBoss EAP **`modcluster`** subsystem.
   It is necessary to provide a list of proxies because the **`modcluster`** subsystem will not be able to automatically discover proxies if advertising is disabled.

   First, define the outbound socket bindings in the appropriate socket binding group.

   ```
   /socket-binding-group=full-ha-sockets/remote-destination-outbound-
   socket-binding=proxy1:add(host=10.33.144.3,port=6666)
   /socket-binding-group=full-ha-sockets/remote-destination-outbound-
   socket-binding=proxy2:add(host=10.33.144.1,port=6666)
   ```

   Next, add the proxies to the mod_cluster configuration.

   ```
   /profile=full-ha/subsystem=modcluster/mod-cluster-
   config=configuration:list-add(name=proxies,value=proxy1)
   /profile=full-ha/subsystem=modcluster/mod-cluster-
   config=configuration:list-add(name=proxies,value=proxy2)
   ```

The Apache HTTP Server balancer no longer advertises its presence to worker nodes and UDP multicast is no longer used.

22.6.3. Configure a mod_cluster Worker Node

A mod_cluster worker node consists of a JBoss EAP server. This server can be a standalone server or part of a server group in a managed domain. A separate process runs within JBoss EAP, which manages all of the worker nodes of the cluster. This is called the master.

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.

- 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 ships with a cluster-enabled server group called other-server-group which meets these requirements.

Configure a Worker Node
The management CLI commands in this procedure assume that you are using a managed domain with the **`full-ha`** profile. If you are running a standalone server, remove the **`/profile=full-ha`** portion of the commands.

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.

   Use the following management CLI commands to modify the external IP addresses for the **`management`**, **`public`**, and **`unsecure`** interfaces as appropriate for your environment. 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 the server.

```
reload
```

2. Configure host names.
   Set a unique host name for each host that participates in a managed domain. 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.

   a. Start the JBoss EAP slave host, using the appropriate **host.xml** configuration file.

   ```
   $ EAP_HOME/bin/domain.sh --host-config=host-slave.xml
   ```

   b. Use the following management CLI command to set a unique host name. This example uses **slave1** as the new host name.

   ```
   /host=EXISTING_HOST_NAME:write-attribute(name=name,value=slave1)
   ```

   For more information on configuring a host name, see Configure the Name of a Host.

3. Configure each host to connect to the domain controller.

NOTE

This step does not apply for a standalone server.

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.

   a. Start the JBoss EAP slave host, using the appropriate **host.xml** configuration file.

   ```
   $ EAP_HOME/bin/domain.sh --host-config=host-slave.xml
   ```

   b. Use the following management CLI command to configure the domain controller settings.

   ```
   /host=SLAVE_HOST_NAME:write-remote-domain-
   controller(host=DOMAIN_CONTROLLER_IP_ADDRESS,port=${jboss.domain.
   master.port:9999},security-realm="ManagementRealm")
   ```

   This modifies the XML in the host-slave.xml file as follows:

   ```
   <domain-controller>
       <remote host="DOMAIN_CONTROLLER_IP_ADDRESS"
   ```

```
port="${jboss.domain.master.port:9999}" security-
realm="ManagementRealm"/>
</domain-controller>
```

For more information, see Connect to the Domain Controller.

4. Configure authentication for each slave host.
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 for each host. Add a management user for each host with the username that matches the host name of the slave.

Be sure to answer **yes** to the last question, that asks "Is this new user going to be used for one AS process to connect to another AS process?", so that you are provided with a secret value.

Example add-user script output *(trimmed)*

```
$ EAP_HOME/bin/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

Username : slave1
Password : changeme
Re-enter Password : changeme
What groups do you want this user to belong to? (Please enter a
comma separated list, or leave blank for none)[  ]:
About to add user 'slave1' for realm 'ManagementRealm'
Is this correct yes/no? yes
Is this new user going to be used for one AS process to connect to
another AS process?
e.g. for a slave host controller connecting to the master or for a
Remoting connection for server to server EJB calls.
yes/no? yes
To represent the user add the following to the server-identities
definition <secret value="SECRET_VALUE" />
```

Copy the Base64-encoded secret value provided from this output (**SECRET_VALUE**), which may be used in the next step.

For more information, see the Adding a User to the Master Domain Controller section of the JBoss EAP *How To Configure Server Security* guide.

5. Modify the slave host's security realm to use the new authentication.
You can specify the password by setting the secret value in the server configuration, getting the password from the vault, or passing the password as a system property.

- Specify the Base64-encoded password value in the server configuration file using the Management CLI.
Use the following management CLI command to specify the secret value. Be sure to replace the **SECRET_VALUE** with the secret value returned from the add-user output from the previous step.

```
/host=SLAVE_HOST_NAME/core-service=management/security-
realm=ManagementRealm/server-
identity=secret:add(value="SECRET_VALUE")
```

You will need to reload the server. The **--host** argument is not applicable for a standalone server.

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

For more information, see the Configuring the Slave Controllers to Use the Credential section of the JBoss EAP *How To Configure Server Security* guide.

- Configure the host to get the password from the vault.

  a. Use the **EAP_HOME/bin/vault.sh** script to generate a masked password. It will generate a string in the format **VAULT::secret::password::VAULT_SECRET_VALUE**, for example:

  ```
  VAULT::secret::password::ODVmYmJjNGMtZDU2ZC00YmNlLWE4ODMtZjQ1N
  WNmNDU4ZDc1TElORV9CUkVBS3ZhdWx0.
  ```

  NOTE

  When creating a password in the vault, it must be specified in plain text, not Base64-encoded.

  b. Use the following management CLI command to specify the secret value. Be sure to replace the **VAULT_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::VAULT_SE
  CRET_VALUE}")
  ```

  You will need to reload the server. The **--host** argument is not applicable for a standalone server.

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

  For more information, see the Password Vault section of the JBoss EAP *How To Configure Server Security* guide.

- 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.
  Use the following managemente CLI command to configure the secret identity to use the system property.

```
/host=SLAVE_HOST_NAME/core-service=management/security-
realm=ManagementRealm/server-
identity=secret:add(value="${server.identity.password}")
```

You will need to reload the server. The **--host** argument is not applicable for a standalone server.

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

b. Set the password for the system property when starting the server.
You can set the **server.identity.password** system property by passing it as a command line argument or in a properties file.

i. Pass in as a plain text command line argument.
Start the server and pass in the **server.identity.password** property.

```
$ EAP_HOME/bin/domain.sh --host-config=host-slave.xml -
Dserver.identity.password=changeme
```

WARNING

The password must be entered in plain text and will be visible to anyone who issues a **ps -ef** command.

ii. Set the property in a properties file.
Create a properties file and add the key/value pair to a properties file, for example:

```
server.identity.password=changeme
```

WARNING

The password is in plain text and will be visible to anyone who has access to this properties file.

Start the server with the command line arguments.

```
$ EAP_HOME/bin/domain.sh --host-config=host-slave.xml --
properties=PATH_TO_PROPERTIES_FILE
```

6. 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.

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.

22.6.4. Configure the mod_cluster fail_on_status Parameter

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 fail_on_status Configuration

```
ProxyPass / balancer://MyBalancer stickysession=JSESSIONID|jsessionid
nofailover=on failonstatus=203,204
ProxyPassReverse / balancer://MyBalancer
ProxyPreserveHost on
```

22.6.5. Migrate Traffic Between Clusters

After creating a new cluster using JBoss EAP, 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.

- A new cluster setup: (we will call this cluster: *ClusterNEW*).
- An old cluster setup that is being made redundant (we will call this cluster: *ClusterOLD*).

Upgrade Process for Clusters - Load-Balancing Groups

1. Set up 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)
   ```

3. Set the **load-balancing-group** to **ClusterOLD**, assuming that all the cluster nodes in *ClusterOLD* are members of *ClusterOLD* load-balancing group.

```
/profile=full-ha/subsystem=modcluster/mod-cluster-
config=configuration/:write-attribute(name=load-balancing-
group,value=ClusterOLD)
```

```
<subsystem xmlns="urn:jboss:domain:modcluster:2.0">
  <mod-cluster-config load-balancing-group="ClusterOLD" advertise-
socket="modcluster" connector="ajp">
    <dynamic-load-provider>
      <load-metric type="cpu"/>
    </dynamic-load-provider>
  </mod-cluster-config>
</subsystem>
```

4. Add the nodes in *ClusterNEW* to the mod_cluster configuration individually using the process described in the Configure a mod_cluster Worker Node section. Additionally use the aforementioned procedure and set their load-balancing group to *ClusterNEW*.
At this point, you 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]
```

5. 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 *ClusterOLD* nodes. Contexts of these nodes will be disabled and once there are no active sessions present, they will be ready for removal. New clients' sessions will be created only on nodes with enabled contexts, presumably *ClusterNEW* members in this example.

Using the Management CLI
In addition to using the *mod_cluster-manager* web console, you can use the JBoss EAP management CLI to stop or disable a particular context.

Stop a Context

```
/host=master/server=server-one/subsystem=modcluster:stop-
context(context=/my-deployed-application-context, virtualhost=default-
host, waittime=0)
```

Stopping a context with **`waittime`** set to **`0`**, meaning no timeout, instructs the balancer to stop routing any request to it immediately, which forces failover to another available context.

If you set a timeout value using the **`waittime`** argument, no new sessions are created on this context, but existing sessions will continue to be directed to this node until they complete or the specified timeout has elapsed. The **`waittime`** argument defaults to **`10`** seconds.

Disable a Context

```
/host=master/server=server-one/subsystem=modcluster:disable-
context(context=/my-deployed-application-context, virtualhost=default-
host)
```

Disabling a context tells the balancer that no new sessions should be created on this context.

22.7. APACHE MOD_JK HTTP CONNECTOR

Apache mod_jk is an HTTP connector that is provided for customers who need it for compatibility purposes.

JBoss EAP 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 servers. If sticky sessions are enabled, the same client request always goes to the same JBoss EAP server, unless the server is unavailable.

mod_jk communicates over the AJP 1.3 protocol. Other protocols can be used with *mod_cluster* or *mod_proxy*. See the Overview of HTTP Connectors for more information.

NOTE

mod_cluster is a more advanced load balancer than mod_jk and is the recommended HTTP connector. mod_cluster provides all of the functionality of mod_jk, plus additional features. Unlike the JBoss EAP 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.

See the Apache mod_jk documentation for more information.

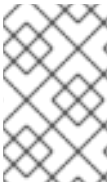
22.7.1. Configure mod_jk in Apache HTTP Server

The mod_jk module (**`mod_jk.so`**) is already included when installing JBoss Core Services Apache HTTP Server or using JBoss Web Server, however, it is not loaded by default.

NOTE

Apache HTTP Server is no longer distributed with JBoss Web Server as of version 3.1.0.

Use the following steps to load and configure mod_jk in Apache HTTP Server. Note that these steps assume that you have already navigated to the **`httpd/`** directory for Apache HTTP Server, which will vary depending on your platform. For more information, see the installation instructions for your platform in the JBoss Core Services *Apache HTTP Server Installation Guide*



NOTE

Red Hat customers can also use the Load Balancer Configuration Tool on the Red Hat Customer Portal to quickly generate optimal configuration templates for mod_jk and other connectors. Note that you must be logged in to access this tool.

1. Configure the mod_jk module.

NOTE

A sample mod_jk configuration file is provided at **conf.d/mod_jk.conf.sample**. You can use this sample instead of creating your own file by removing the **.sample** extension and modifying its contents as needed.

Create a new file called **conf.d/mod_jk.conf**. Add the following configuration to the file, making sure to modify the contents to suite your needs.

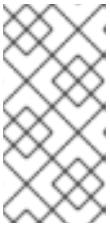
```
# Load mod_jk module
# Specify the filename of the mod_jk lib
LoadModule jk_module modules/mod_jk.so

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

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

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

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

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

# JkRequestLogFormat
JkRequestLogFormat "%w %V %T"

# Mount your applications
JkMount /application/* loadbalancer

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

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

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.

Aside from general mod_jk configuration, this file specifies to load the **mod_jk.so** module, and defines where to find the **workers.properties** file.

2. Configure the mod_jk worker nodes.



NOTE

A sample workers configuration file is provided at **conf.d/workers.properties.sample**. You can use this sample instead of creating your own file by removing the **.sample** extension and modifying its contents as needed.

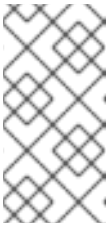
Create a new file called **conf.d/workers.properties**. Add the following configuration to the file, making sure to modify the contents to suite your needs.

```
# 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 details on the syntax of the mod_jk **workers.properties** file and other advanced configuration options, see mod_jk Worker Properties.

3. Optionally, 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.

   a. Create a **uriworkermap.properties** file.



NOTE

A sample URI worker map configuration file is provided at **`conf.d/uriworkermap.properties.sample`**. You can use this sample instead of creating your own file by removing the **`.sample`** extension and modifying its contents as needed.

Create a new file called **`conf.d/uriworkermap.properties`**. Add a line for each URL pattern to be matched, for example:

```
# Simple worker configuration file
/*=loadbalancer
```

b. Update the configuration to point to **`uriworkermap.properties`** file. Append the following to **`conf.d/mod_jk.conf`**.

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

For more details on configuring mod_jk, see the Configuring Apache HTTP Server to Load mod_jk section of the JBoss Web Server *HTTP Connectors and Load Balancing Guide*

22.7.2. Configure JBoss EAP to Communicate with mod_jk

The JBoss EAP **`undertow`** subsystem needs to specify a listener in order to accept requests from and send replies back to an external web server. Since mod_jk uses the AJP protocol, an AJP listener must be configured.

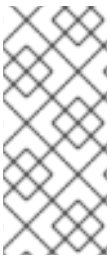
If you are using one of the default high availability configurations (*ha* or *full-ha*), an AJP listener is already configured.

For instructions, see Accepting Requests From External Web Servers .

22.8. APACHE MOD_PROXY HTTP CONNECTOR

Apache mod_proxy is an HTTP connector that supports connections over AJP, HTTP, and HTTPS protocols. mod_proxy can be configured in load-balanced or non-load-balanced configurations, and it supports the notion of sticky sessions.

The mod_proxy module requires JBoss EAP to have the HTTP, HTTPS or AJP listener configured in the **`undertow`** subsystem, depending on which protocol you plan to use.



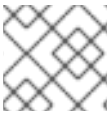
NOTE

mod_cluster is a more advanced load balancer than mod_proxy and is the recommended HTTP connector. mod_cluster provides all of the functionality of mod_proxy, plus additional features. Unlike the JBoss EAP mod_cluster HTTP connector, an Apache mod_proxy HTTP connector does not know the status of deployments on servers or server groups, and cannot adapt where it sends its work accordingly.

See the Apache mod_proxy documentation for more information.

22.8.1. Configure mod_proxy in Apache HTTP Server

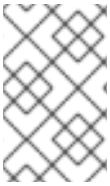
The mod_proxy modules are already included when installing JBoss Core Services Apache HTTP Server or using JBoss Web Server and are loaded by default.



NOTE

Apache HTTP Server is no longer distributed with JBoss Web Server as of version 3.1.0.

See the appropriate section below to configure a basic load-balancing or non-load-balancing proxy. These steps assume that you have already navigated to the **`httpd/`** directory for Apache HTTP Server, which will vary depending on your platform. For more information, see the installation instructions for your platform in the JBoss Core Services *Apache HTTP Server Installation Guide*. These steps also assume that the necessary HTTP listener has already been configured in the JBoss EAP **`undertow`** subsystem.



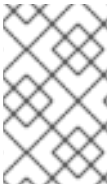
NOTE

Red Hat customers can also use the Load Balancer Configuration Tool on the Red Hat Customer Portal to quickly generate optimal configuration templates for mod_proxy and other connectors. Note that you must be logged in to access this tool.

Add a Non-load-balancing Proxy
Add the following configuration to your **`conf/httpd.conf`** file, directly beneath any other **`<VirtualHost>`** directives you may have. Replace the values with ones appropriate to your setup.

```
<VirtualHost *:80>
# Your domain name
ServerName YOUR_DOMAIN_NAME

ProxyPreserveHost On

# The IP and port of JBoss
# These represent the default values, if your httpd is on the same host
# as your JBoss 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>
```

Add a Load-balancing Proxy



NOTE

The default Apache HTTP Server configuration has the **mod_proxy_balancer.so** module disabled, as it is incompatible with mod_cluster. In order to complete this task, you will need to load this module and disable the mod_cluster modules.

To use mod_proxy as a load balancer, and send work to multiple JBoss EAP instances, add the following configuration to your **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. See the Apache mod_proxy documentation for more details.

Enable Sticky Sessions

A *sticky session* means that if a client request originally goes to a specific JBoss EAP worker, all future requests will be sent to the same worker, unless it becomes unavailable. This is almost always the recommended behavior.

To enable sticky sessions for mod_proxy, add the **stickysession** parameter to the **ProxyPass** statement.

```
ProxyPass / balancer://mycluster stickysession=JSESSIONID
```

You can specify additional parameters to the **ProxyPass** statement, such as **lbmethod** and **nofailover**. See the Apache mod_proxy documentation for more information on the available parameters.

22.8.2. Configure JBoss EAP to Communicate with mod_proxy

The JBoss EAP **undertow** subsystem needs to specify a listener in order to accept requests from and send replies back to an external web server. Depending on the protocol that you will be using, you may need to configure a listener.

An HTTP listener is configured in the JBoss EAP default configuration. If you are using one of the default high availability configurations (*ha* or *full-ha*), an AJP listener is also preconfigured.

For instructions, see Accepting Requests From External Web Servers .

22.9. MICROSOFT ISAPI CONNECTOR

The 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 instances as worker nodes with IIS as a load balancer.

NOTE

See JBoss EAP supported configurations for information on the supported configurations of Windows Server and IIS.

22.9.1. Configure Microsoft IIS to Use the ISAPI Connector

Download the ISAPI connector from the Red Hat Customer Portal:

1. Open a browser and log in to the Red Hat Customer Portal JBoss Software Downloads page.
2. Select **Web Connectors** in the **Product** drop-down menu.
3. Select the latest JBoss Core Services version from the **Version** drop-down menu.
4. Find the **Red Hat JBoss Core Services ISAPI Connector** in the list, and click the **Download** link.
5. Extract the archive and copy the contents of the **sbin** directory to a location on your server. The instructions below assume that the contents were copied to **C:\connectors**.

To 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
   - Right-click *Default Web Site*, and click *Add Virtual Directory*. The *Add Virtual Directory* window opens.
   - Specify the following values to add a virtual directory:
     - **Alias**: `jboss`
     - **Physical Path**: `C:\connectors\`
   - Click **OK** to save the values and close the **Add Virtual Directory** window.
8. Define a JBoss Native ISAPI Redirect Filter
   - In the tree view pane, expand **Sites** → **Default Web Site**.
   - Double-click **ISAPI Filters**. The **ISAPI Filters Features** view appears.
   - In the **Actions** pane, click Add. The **Add ISAPI Filter** window appears.
   - Specify the following values in the **Add ISAPI Filter** window:
     - **Filter name**: `jboss`
     - **Executable**: `C:\connectors\isapi_redirect.dll`
   - Click **OK** to save the values and close the **Add ISAPI Filters** window.
9. Enable the ISAPI-dll handler
   - Double-click the **IIS 7** item in the tree view pane. The **IIS 7 Home Features View** opens.
   - Double-click **Handler Mappings**. The **Handler Mappings Features View** appears.
   - In the **Group by** combo box, select **State**. The **Handler Mappings** are displayed in **Enabled and Disabled Groups**.
   - Find `ISAPI-dll`. If it is in the **Disabled** group, right-click it and select **Edit Feature Permissions**.
   - Enable the following permissions:
     - Read
     - Script
     - Execute
   - Click **OK** to save the values, and close the **Edit Feature Permissions** window.

Microsoft IIS is now configured to use the ISAPI connector.

22.9.2. Configure the ISAPI Connector to Send Client Requests to JBoss EAP

This task configures a group of JBoss EAP servers to accept requests from the ISAPI connector. It does not include configuration for load-balancing or high-availability failover.

This configuration is done on the IIS server, and assumes that you already have JBoss EAP configured to accept requests from an external web server . You also need full administrator access to the IIS server and to have configured IIS to use the ISAPI connector .

Create Property Files and Set Up 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.

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. This file follows the syntax of the same file used for Apache mod_jk worker properties configuration.

   The following is an example of a **workers.properties** file. The worker names, **worker01** and **worker02**, must match the **instance-id** configured in the JBoss EAP **undertow** subsystem.

   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 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 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 your IIS server by using the **net stop** and **net start** commands.

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

The IIS server is configured to send client requests to the specific JBoss EAP servers you have configured, on an application-specific basis.

22.9.3. Configure the ISAPI Connector to Balance Client Requests Across Multiple JBoss EAP Servers

This configuration balances client requests across the JBoss EAP servers you specify. This configuration is done on the IIS server, and assumes that you already have JBoss EAP configured to accept requests from an external web server . You also need full administrator access to the IIS server and to have configured IIS to use the ISAPI connector .

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.

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 the next step.

   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. This file follows the syntax of the same file used for Apache mod_jk worker properties configuration.

   The following is an example of a **workers.properties** file. The load balancer is configured near the end of the file, to comprise workers **worker01** and **worker02**. These worker names must match the **instance-id** configured in the JBoss EAP **undertow** subsystem.

   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/
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
```

The IIS server is configured to send client requests to the JBoss EAP 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.

22.10. ORACLE NSAPI CONNECTOR

The 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. The NSAPI connector is used in **nsapi_redirector.so**, which is an extension of mod_jk adjusted to Oracle iPlanet Web Server. The NSAPI connector enables you to configure JBoss EAP instances as worker nodes with Oracle iPlanet Web Server as a load balancer.

NOTE

See the JBoss EAP supported configurations for information on the supported configurations of Solaris and Oracle iPlanet Web Server.

22.10.1. Configure Oracle iPlanet Web Server to use the NSAPI Connector

Prerequisites

- JBoss EAP is installed and configured on each server that will serve as a worker.

Download the NSAPI connector from the Red Hat Customer Portal:

1. Open a browser and log in to the Red Hat Customer Portal JBoss Software Downloads page.
2. Select **Web Connectors** in the **Product** drop-down menu.

3. Select the latest JBoss Core Services version from the **Version** drop-down menu.

4. Find the **Red Hat JBoss Core Services NSAPI Connector** in the list, ensuring that you select the correct platform and architecture for your system, and click the **Download** link.

5. Extract the **nsapi_redirector.so** file, which is located in either the **lib/** or the **lib64/** directory, into either the **IPLANET_CONFIG/lib/** or the **IPLANET_CONFIG/lib64/** directory.

Set up the NSAPI Connector:

NOTE

In these instructions, **IPLANET_CONFIG** refers to the Oracle iPlanet configuration directory, which is usually **/opt/oracle/webserver7/config/**. If your Oracle iPlanet configuration directory is different, modify the instructions accordingly.

1. 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.

2. 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="/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`**.

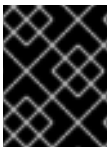
Save and exit the file.

3. 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.

   - Configure the NSAPI Connector to Send Client Requests to JBoss EAP
   - Configure the NSAPI Connector to Balance Client Requests Across Multiple JBoss EAP Servers

22.10.2. Configure the NSAPI Connector to Send Client Requests to JBoss EAP

This task configures the NSAPI connector to redirect client requests to JBoss EAP servers with no load balancing or failover. The redirection is done on a per-deployment (and hence per-URL) basis.



IMPORTANT

You must have already configured the NSAPI connector before continuing with this task.

Set Up the Basic HTTP Connector

1. Define the URL paths to redirect to the JBoss EAP 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.

   Save and exit the file.

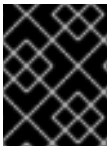
4. Restart the iPlanet Web Server
   Issue the following command to restart the iPlanet Web Server.

   ```
   IPLANET_CONFIG/../bin/stopserv
   IPLANET_CONFIG/../bin/startserv
   ```

iPlanet Web Server now sends client requests to the URLs you have configured to deployments on JBoss EAP.

22.10.3. Configure the NSAPI Connector to Balance Client Requests Across Multiple JBoss EAP Servers

This task configures the NSAPI connector to send client requests to JBoss EAP servers in a load-balancing configuration.

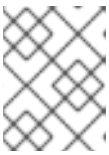


IMPORTANT

You must have already configured the NSAPI connector before continuing with this task.

Configure the Connector for Load Balancing

1. Define the URL paths to redirect to the JBoss EAP 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 that 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 **`jksnapi`** object defines the worker nodes used to serve each path that was mapped to the **`name="jksnapi"`** 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.

Save and exit the file.

4. Restart the iPlanet Web Server 7.0.

```
IPLANET_CONFIG/../bin/stopserv
IPLANET_CONFIG/../bin/startserv
```

The iPlanet Web Server redirects the URL patterns you have configured to your JBoss EAP servers in a load-balancing configuration.

APPENDIX A. REFERENCE MATERIAL

A.1. SERVER RUNTIME ARGUMENTS

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 A.1. Runtime Switches and Arguments

Argument or Switch	Operating 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**.
-c=<config>, -c <config>	Domain	Name of the server configuration file to use. The default is **domain.xml**.

Argument or Switch	Operating Mode	Description
--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	Operating 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.
-secmgr	Standalone, Domain	Runs the server with a security manager installed.
--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.

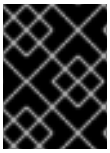
WARNING

The configuration files that ship with JBoss EAP are set up to handle the behavior of the switches, for example, `-b` and `-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.

A.2. RPM SERVICE CONFIGURATION FILES

The RPM installation of JBoss EAP includes two additional configuration files compared to a ZIP or installer installation. These files are used by the service init script to specify the JBoss EAP launch environment. The location of these service configuration files differ for Red Hat Enterprise Linux 6 and

Red Hat Enterprise Linux 7.



IMPORTANT

For Red Hat Enterprise Linux 7, RPM service configuration files are loaded using `systemd`, so variable expressions are not expanded.

Table A.2. RPM Configuration Files for Red Hat Enterprise Linux 6

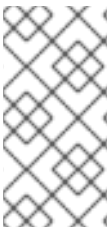
File	Description
/etc/sysconfig/eap7-standalone	Settings specific to standalone JBoss EAP servers on Red Hat Enterprise Linux 6.
/etc/sysconfig/eap7-domain	Settings specific to JBoss EAP running as a managed domain on Red Hat Enterprise Linux 6.

Table A.3. RPM Configuration Files for Red Hat Enterprise Linux 7

File	Description
/etc/opt/rh/eap7/wildfly/eap7-standalone.conf	Settings specific to standalone JBoss EAP servers on Red Hat Enterprise Linux 7.
/etc/opt/rh/eap7/wildfly/eap7-domain.conf	Settings specific to JBoss EAP running as a managed domain on Red Hat Enterprise Linux 7.

A.3. RPM SERVICE CONFIGURATION PROPERTIES

The following table shows a list of available configuration properties for the JBoss EAP RPM service along with their default values.



NOTE

If a property has the same name in both the RPM service configuration file (for example, `/etc/sysconfig/eap7-standalone`) and in the JBoss EAP startup configuration file (for example, `EAP_HOME/bin/standalone.conf`), the value that takes precedence is the one in the JBoss EAP startup configuration file. One such property is **`JAVA_HOME`**.

Table A.4. RPM Service Configuration Properties

Property	Description
JAVA_HOME	The directory where your Java Runtime Environment is installed. Default value: `/usr/lib/jvm/jre`

Property	Description
JAVAPTH	The path where the Java executable files are installed. Default value: ***$JAVA_HOME*/bin**
WILDFLY_STARTUP_WAIT	The number of seconds that the init script will wait until confirming that the server has launched successfully after receiving a start or restart command. This property only applies to Red Hat Enterprise Linux 6. Default value: **60**
WILDFLY_SHUTDOWN_WAIT	The number of seconds that the init script will wait for the server to shutdown before continuing when it receives a stop or restart command. This property only applies to Red Hat Enterprise Linux 6. Default value: **20**
WILDFLY_CONSOLE_LOG	The file that the CONSOLE log handler will be redirected to. Default value: **/var/opt/rh/eap7/log/wildfly/standalone/console.log** for a standalone server, or **/var/opt/rh/eap7/log/wildfly/domain/console.log** for a managed domain.
WILDFLY_SH	The script which is used to launch to JBoss EAP server. Default value: **/opt/rh/eap7/root/usr/share/wildfly/bin/standalone.sh** for a standalone server, or **/opt/rh/eap7/root/usr/share/wildfly/bin/domain.sh** for a managed domain.
WILDFLY_SERVER_CONFIG	The server configuration file to use. There is no default for this property. Either **standalone.xml** or **domain.xml** can be defined at start.
WILDFLY_HOST_CONFIG	For a managed domain, this property allows a user to specify the host configuration file (such as **host.xml**). It has no value set as the default.
WILDFLY_MODULEPATH	The path of the JBoss EAP module directory. Default value: **/opt/rh/eap7/root/usr/share/wildfly/modules**
WILDFLY_BIND	Sets the **jboss.bind.address** system property, which is used to configure the bind address for the public interface. This defaults to **0.0.0.0** if no value is specified.

A.4. OVERVIEW OF JBOSS EAP SUBSYSTEMS

The table below gives a brief description of the JBoss EAP subsystems.

Table A.5. JBoss EAP Subsystems

JBoss EAP Subsystem	Description
batch-jberet	Configure an environment for running batch applications and manage batch jobs.
bean-validation	Configure bean validation for validating Java object data.
datasources	Create and configure datasources and manage JDBC database drivers.
deployment-scanner	Configure deployment scanners to monitor particular locations for applications to deploy.
ee	Configure common functionality in the Java EE platform, such as defining global modules, enabling descriptor-based property replacement, and configuring default bindings.
ejb3	Configure Enterprise JavaBeans (EJBs), including session and message-driven beans. More information for the `ejb3` subsystem can be found in *Developing EJB Applications* for JBoss EAP.
iiop-openjdk	Configure Common Object Request Broker Architecture (CORBA) services for JTS transactions and other ORB services, including security. In JBoss EAP 6, this functionality was contained in the `jacorb` subsystem.
infinispan	Configure caching functionality for JBoss EAP high availability services.
io	Define workers and buffer pools to be used by other subsystems.
jaxrs	Enable the deployment and functionality of JAX-RS applications.
jca	Configure the general settings for the Java EE Connector Architecture (JCA) container and resource adapter deployments.
jdr	Enable the gathering of diagnostic data to aid in troubleshooting. JBoss EAP subscribers can provide this information to Red Hat when requesting support.
jgroups	Configure the protocol stacks and communication mechanisms for how servers in a cluster talk to each other.
jmx	Configure remote Java Management Extensions (JMX) access.

JBoss EAP Subsystem	Description
jpa	Manages the Java Persistence API (JPA) 2.1 container-managed requirements and allows you to deploy persistent unit definitions, annotations, and descriptors. More information for the **`jpa`** subsystem can be found in the JBoss EAP *Development Guide*.
jsf	Manage JavaServer Faces (JSF) implementations.
jsr77	Provide Java EE management capabilities defined by the JSR-77 specification.
logging	Configure system and application-level logging through a system of log categories and log handlers.
mail	Configure mail server attributes and custom mail transports to create a mail service that allows applications deployed to JBoss EAP to send mail using that service.
messaging-activemq	Configure JMS destinations, connection factories, and other settings for Artemis, the integrated messaging provider. In JBoss EAP 6, messaging functionality was contained in the **`messaging`** subsystem. More information for the **`messaging-activemq`** subsystem can be found in *Configuring Messaging* for JBoss EAP.
modcluster	Configure the server-side mod_cluster worker node.
naming	Bind entries into global JNDI namespaces and configure the remote JNDI interface.
picketlink-federation	Configure PicketLink SAML-based single sign-on (SSO). More information on the **`picketlink-federation`** subsystem can be found in *How To Set Up SSO with SAML v2* for JBoss EAP.
picketlink-identity-management	Configure PicketLink identity management services. This subsystem is unsupported.
pojo	Enable deployment of applications containing JBoss Microcontainer services, as supported by previous versions of JBoss EAP.
remoting	Configure settings for inbound and outbound connections for local and remote services.
request-controller	Configure settings to suspend and shut down servers gracefully.

JBoss EAP Subsystem	Description
resource-adapters	Configure and maintain resource adapters for communication between Java EE applications and an Enterprise Information System (EIS) using the Java Connector Architecture (JCA) specification.
rts	Unsupported implementation of REST-AT.
sar	Enable deployment of SAR archives containing MBean services, as supported by previous versions of JBoss EAP.
security	Configure application security settings. More information on the `security` subsystem can be found in *Security Architecture* for JBoss EAP.
security-manager	Configure Java security policies to be used by the Java Security Manager. More information on the `security-manager` subsystem can be found in *How to Configure Server Security* for JBoss EAP.
singleton	Define singleton policies to configure the behavior of singleton deployments or to create singleton MSC services. More information on the `singleton` subsystem can be found in the JBoss EAP *Development Guide*.
transactions	Configure transaction manager options, such as timeout values, transaction logging, and whether to use Java Transaction Service (JTS).
undertow	Configure JBoss EAP's web server and servlet container settings. In JBoss EAP 6, this functionality was contained in the `web` subsystem.
webservices	Configure published endpoint addresses and endpoint handler chains, as well as the host name, ports, and WSDL address for the web services provider. More information for the `webservices` subsystem can be found in *Developing Web Services Applications* for JBoss EAP.
weld	Configure Contexts and Dependency Injection (CDI) functionality for JBoss EAP.
xts	Configure settings for coordinating web services in a transaction.

A.5. ADD-USER UTILITY ARGUMENTS

The following table describes the arguments available for the `add-user.sh` or `add-user.bat` script, which is a utility for adding new users to the properties file for out-of-the-box authentication.

Table A.6. Add-User Command Arguments

Command Line Argument	Description
-a	Create a user in the application realm. If omitted, the default is to create a user in the management realm.
-dc <value>	The domain configuration directory that will contain the properties files. If it is omitted, the default directory is `EAP_HOME/domain/configuration/`.
-sc <value>	An alternative standalone server configuration directory that will contain the properties files. If omitted, the default directory is `EAP_HOME/standalone/configuration/`.
-up, --user-properties <value>	The name of the alternative 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 alternative configuration directory.
-g, --group <value>	A comma-separated list of groups to assign to this user.
-gp, --group-properties <value>	The name of the alternative 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 alternative configuration directory.
-p, --password <value>	The password of the user.
-u, --user <value>	The name of the user. Only alphanumeric characters and the following symbols are valid: `,.=@\`.
-r, --realm <value>	The name of the realm used to secure the management interfaces. If omitted, the default is `ManagementRealm`.
-s, --silent	Run the `add-user` script with no output to the console.
-e, --enable	Enable the user.
-d, --disable	Disable the user.
-cw, --confirm-warning	Automatically confirm warning in interactive mode.
-h, --help	Display usage information for the `add-user` script.

A.6. MANAGEMENT AUDIT LOGGING ATTRIBUTES

Table A.7. Logger Attributes

Attribute	Description
enabled	Whether audit logging is enabled.

Attribute	Description
log-boot	Whether operations should be logged on server boot.
log-read-only	Whether operations that do not modify the configuration or any runtime services should be logged.

Table A.8. Log Formatter Attributes

Attribute	Description
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`**.
date-format	The date format to use as understood by **`java.text.SimpleDateFormat`**. This is ignored if **`include-date`** is set to **`false`**.
date-separator	The separator between the date and the rest of the formatted log message. This is ignored if **`include-date`** is set to **`false`**.
escape-control-characters	If **`true`**, it will escape all control characters (ASCII entries with a decimal value greater than **`32`**) with the ASCII code in octal. For example, a new line becomes **`#012`**. If **`true`**, this will **`override escape-new-line=false`**.
escape-new-line	If **`true`**, it will escape all new lines with the ASCII code in octal **`#012`**.
include-date	Whether or not to include the date in the formatted log record.

Table A.9. File Handler Attributes

Attribute	Description
disabled-due-to-failure	Whether this handler has been disabled due to logging failures (read-only).
failure-count	The number of logging failures since the handler was initialized (read-only).
formatter	The JSON formatter used to format the log messages.
max-failure-count	The maximum number of logging failures before disabling this handler.
path	The path of the audit log file.

Attribute	Description
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.

Table A.10. Syslog Handler Attributes

Attribute	Description
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	Whether this handler has been disabled due to logging failures (read-only).
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 (read-only).
formatter	The JSON formatter used to format the log messages.
max-failure-count	The maximum number of logging failures before disabling this handler.
max-length	The maximum length in bytes a log message, including the header, is allowed to be. 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`.
syslog-format	The syslog format: `RFC5424` or `RFC3164`.
truncate	Whether or not a message, including the header, should truncate the message 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.

NOTE

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.

This table lists only the high-level attributes. Each attribute has configuration parameters, and some have child configuration parameters.

A.7. INTERFACE ATTRIBUTES

Table A.11. 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	Empty element indicating that sockets using this interface should be bound to a wildcard address. The IPv6 wildcard address (`::`) will be used unless the `java.net.preferIPv4Stack` system property is set to true, in which case the IPv4 wildcard address (`0.0.0.0`) will be used. 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.
inet-address	Either an IP address in IPv6 or IPv4 dotted decimal notation, or a host name 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.

Interface Element	Description
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.

A.8. SOCKET BINDING ATTRIBUTES

Table A.12. Socket Binding Attributes

Attribute	Description
client-mappings	Specifies the client mappings for this socket binding. A client connecting to this socket should use the destination address specified in the mapping that matches its desired outbound interface. This allows for advanced network topologies that use either network address translation, or have bindings on multiple network interfaces to function. Each mapping should be evaluated in declared order, with the first successful match used to determine the destination.
fixed-port	Whether the port value should remain fixed even if numeric offsets are applied to the other sockets in the socket group.
interface	Name of the interface to which the socket should be bound, or, for multicast sockets, the interface on which it should listen. This should be one of the declared interfaces. If not defined, the value of the `default-interface` attribute from the enclosing socket binding group will be used.
multicast-address	Multicast address on which the socket should receive multicast traffic. If unspecified, the socket will not be configured to receive multicast.

Attribute	Description
multicast-port	Port on which the socket should receive multicast traffic. Must be configured if 'multicast-address' is configured.
name	The name of the socket. Services needing to access the socket configuration information will find it using this name. This attribute is required.
port	Number of the port to which the socket should be bound. Note that this value can be overridden if servers apply a port-offset to increment or decrement all port values.

A.9. DEFAULT SOCKET BINDINGS

Table A.13. Default Socket Bindings

Name	Port	Multicast Port	Description	Socket Binding Groups
ajp	8009		Apache JServ Protocol. Used for HTTP clustering and load balancing.	standard-sockets, ha-sockets, full-sockets, full-ha-sockets
http	8080		The default port for deployed web applications.	standard-sockets, ha-sockets, full-sockets, full-ha-sockets
https	8443		SSL-encrypted connection between deployed web applications and clients.	standard-sockets, ha-sockets, full-sockets, full-ha-sockets
iiop	3528		CORBA services for JTS transactions and other ORB-dependent services.	full-sockets, full-ha-sockets
iiop-ssl	3529		SSL-encrypted CORBA services.	full-sockets, full-ha-sockets
jgroups-mping		45700	Multicast. Used to discover initial membership in a HA cluster.	ha-sockets, full-ha-sockets
jgroups-tcp	7600		Unicast peer discovery in HA clusters using TCP.	ha-sockets, full-ha-sockets
jgroups-tcp-fd	57600		Used for HA failure detection over TCP.	ha-sockets, full-ha-sockets

Name	Port	Multicast Port	Description	Socket Binding Groups
jgroups-udp	55200	45688	Multicast peer discovery in HA clusters using UDP.	ha-sockets, full-ha-sockets
jgroups-udp-fd	54200		Used for HA failure detection over UDP.	ha-sockets, full-ha-sockets
management-http	9990		Used for HTTP communication with the management layer.	standard-sockets, ha-sockets, full-sockets, full-ha-sockets
management-https	9993		Used for HTTPS communication with the management layer.	standard-sockets, ha-sockets, full-sockets, full-ha-sockets
modcluster		23364	Multicast port for communication between JBoss EAP and the HTTP load balancer.	ha-sockets, full-ha-sockets
txn-recovery-environment	4712		The JTA transaction recovery manager.	standard-sockets, ha-sockets, full-sockets, full-ha-sockets
txn-status-manager	4713		The JTA / JTS transaction manager.	standard-sockets, ha-sockets, full-sockets, full-ha-sockets

A.10. DEPLOYMENT SCANNER MARKER FILES

Marker files are used by the deployment scanner to mark the status of an application within the deployment directory of the JBoss EAP server instance. A marker file has the same name as the deployment, with the file suffix indicating the state of the application's deployment.

For example, a successful deployment of `test-application.war` would have a marker file named `test-application.war.deployed`.

The following table lists the available marker file types and their meanings.

Table A.14. Marker File Types

Filename Suffix	Origin	Description
.deployed	System-generated	Indicates that the content has been deployed. The content will be undeployed if this file is deleted.

Filename Suffix	Origin	Description
.dodeploy	User-generated	Indicates that the content should be deployed or redeployed.
.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 eligible for auto-deployment again.
.isdeploying	System-generated	Indicates that the deployment is in progress. This marker file will be deleted upon completion.
.isundeploying	System-generated	Triggered by deleting a `.deployed` file, this indicates that the content is being undeployed. This marker file will be deleted upon completion.
.pending	System-generated	Indicates that the deployment scanner recognizes the need to deploy content, but an issue is currently preventing auto-deployment (for example, if content is in the process of being copied). This marker serves as a global deployment road-block, meaning that the scanner will not instruct the server to deploy or undeploy *any* content while this marker file exists.
.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 being pushed. Can be used with zipped content, although the scanner detects in-progress changes to zipped content and waits until completion.
.undeployed	System-generated	Indicates that the content has been undeployed. Deletion of this marker file has no impact to content redeployment.

A.11. DEPLOYMENT SCANNER ATTRIBUTES

The deployment scanner contains the following configurable attributes.

Table A.15. Deployment Scanner Attributes

Name	Default	Description
auto-deploy-exploded	false	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.
auto-deploy-xml	true	Allows the automatic deployment of XML content without requiring a `.dodeploy` marker file.

Name	Default	Description
auto-deploy-zipped	true	Allows the automatic deployment of zipped content without requiring a `.dodeploy` marker file.
deployment-timeout	600	The time value in seconds for the deployment scanner to allow a deployment attempt before being canceled.
path	deployments	The actual file system path to be scanned. Treated as an absolute path, unless the `relative-to` attribute is specified, in which case the value is treated as relative to that path.
relative-to	jboss.server.base.dir	Reference to a file system path defined as a path in the server configuration.
runtime-failure-causes-rollback	false	Whether a runtime failure of a deployment causes a rollback of the deployment as well as all other (possibly unrelated) deployments as part of the scan operation.
scan-enabled	true	Allows the automatic scanning for applications by `scan-interval` and at startup.
scan-interval	5000	The time interval in milliseconds that the repository should be scanned for changes. A value of less than `1` causes the scan to occur only at initial startup.

A.12. MAIL SUBSYSTEM ATTRIBUTES

The following tables describe the attributes in the `mail` subsystem for mail sessions and the following mail server types:

- imap
- pop3
- smtp
- custom

Table A.16. Mail Session Attributes

Attribute	Description
debug	Whether to enable JavaMail debugging.
from	The default "from" address to use if not set when sending.

Attribute	Description
jndi-name	The JNDI name to which the mail session should be bound.

Table A.17. IMAP Mail Server Attributes

Attribute	Description
outbound-socket-binding-ref	Reference to the outbound socket binding for the mail server.
password	The password to authenticate on the server.
ssl	Whether the server requires SSL.
tls	Whether the server requires TLS.
username	The username to authenticate on the server.

Table A.18. POP3 Mail Server Attributes

Attribute	Description
outbound-socket-binding-ref	Reference to the outbound socket binding for the mail server.
password	The password to authenticate on the server.
ssl	Whether the server requires SSL.
tls	Whether the server requires TLS.
username	The username to authenticate on the server.

Table A.19. SMTP Mail Server Attributes

Attribute	Description
outbound-socket-binding-ref	Reference to the outbound socket binding for the mail server.
password	The password to authenticate on the server.
ssl	Whether the server requires SSL.
tls	Whether the server requires TLS.

Attribute	Description
username	The username to authenticate on the server.

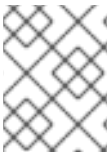
Table A.20. Custom Mail Server Attributes

Attribute	Description
outbound-socket-binding-ref	Reference to the outbound socket binding for the mail server.
password	The password to authenticate on the server.
properties	The JavaMail properties for this server.
ssl	Whether the server requires SSL.
tls	Whether the server requires TLS.
username	The username to authenticate on the server.

A.13. ROOT LOGGER ATTRIBUTES

Table A.21. Root Logger Attributes

Attribute	Description
filter	Defines a simple filter type. *Deprecated in favor of* ***`filter-spec`***.
filter-spec	An expression value that defines a filter. The following expression defines a filter that excludes log entries that do *not* match a pattern: `not(match("WFLY.*"))`
handlers	A list of log handlers that are used by the root logger.
level	The lowest level of log message that the root logger records.



NOTE

A `filter-spec` specified for the root logger is *not* inherited by other handlers. Instead a `filter-spec` must be specified per handler.

A.14. LOG CATEGORY ATTRIBUTES

Table A.22. Log Category Attributes

Attribute	Description
category	The log category from which log messages will be captured.
filter	Defines a simple filter type. *Deprecated in favor of* ***`filter-spec`***.
filter-spec	An expression value that defines a filter. The following expression defines a filter that does not match a pattern: **`not(match("WFLY.*"))`**
handlers	A list of log handlers associated with the logger.
level	The lowest level of log message that the log category records.
use-parent-handlers	If set to **`true`**, this category will use the log handlers of the root logger in addition to any other assigned handlers.

A.15. LOG HANDLER ATTRIBUTES

Table A.23. Console Log Handler Attributes

Attribute	Description
autoflush	If set to **`true`**, the log messages will be sent to the handlers assigned file immediately upon receipt.
enabled	If set to **`true`**, the handler is enabled and functioning as normal. If set to **`false`**, the handler is ignored when processing log messages.
encoding	The character encoding scheme to be used for the output.
filter	Defines a simple filter type. *Deprecated in favor of* ***`filter-spec`***.
filter-spec	An expression value that defines a filter. The following expression defines a filter that does not match a pattern: **`not(match("WFLY.*"))`**
formatter	The log formatter used by this log handler.
level	The lowest level of log message the log handler records.
name	The name of the log handler. *Deprecated since the handler's address contains the name.*
named-formatter	The name of the defined formatter to be used on the handler.
target	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.

Table A.24. File Log Handler Attributes

Attribute	Description
append	If set to `true`, 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.
autoflush	If set to `true`, the log messages will be sent to the handlers assigned file immediately upon receipt.
enabled	If set to `true`, the handler is enabled and functioning as normal. If set to `false`, the handler is ignored when processing log messages.
encoding	The character encoding scheme to be used for the output.
file	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`.
filter	Defines a simple filter type. *Deprecated in favor of* ***`filter-spec`***.
filter-spec	An expression value that defines a filter. The following expression defines a filter that does not match a pattern: `not(match("WFLY.*"))`
formatter	The log formatter used by this log handler.
level	The lowest level of log message the log handler records.
name	The name of the log handler. *Deprecated since the handler's address contains the name.*
named-formatter	The name of the defined formatter to be used on the handler.

Table A.25. Periodic Log Handler Attributes

Attribute	Description
append	If set to `true`, 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.
autoflush	If set to `true`, the log messages will be sent to the handlers assigned file immediately upon receipt.
enabled	If set to `true`, the handler is enabled and functioning as normal. If set to `false`, the handler is ignored when processing log messages.
encoding	The character encoding scheme to be used for the output.

Attribute	Description
file	Object that represents the file to which the output of this log handler is written. It has two configuration properties, `relative-to` and `path`.
filter	Defines a simple filter type. *Deprecated in favor of* ***`filter-spec`***.
filter-spec	An expression value that defines a filter. The following expression defines a filter that does not match a pattern: `not(match("WFLY.*"))`.
formatter	The log formatter used by this log handler.
level	The lowest level of log message the log handler records.
name	The name of the log handler. *Deprecated since the handler's address contains the name.*
named-formatter	The name of the defined formatter to be used on the handler.
suffix	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.

Table A.26. Size Log Handler Attributes

Attribute	Description
append	If set to `true`, 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.
autoflush	If set to `true` the log messages will be sent to the handlers assigned file immediately upon receipt.
enabled	If set to `true`, the handler is enabled and functioning as normal. If set to`false`, the handler is ignored when processing log messages.
encoding	The character encoding scheme to be used for the output.
file	Object that represents the file where the output of this log handler is written to. It has two configuration properties, `relative-to` and `path`.
filter	Defines a simple filter type. *Deprecated in favor of* ***`filter-spec`***.
filter-spec	An expression value that defines a filter. The following expression defines a filter that does not match a pattern: `not(match("WFLY.*"))`

Attribute	Description
formatter	The log formatter used by this log handler.
level	The lowest level of log message the log handler records.
max-backup-index	The maximum number of rotated logs that are kept. When this number is reached, the oldest log is reused. The default is `1`. 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`.
name	The name of the log handler. *Deprecated since the handler's address contains the name.*
named-formatter	The name of the defined formatter to be used on the handler.
rotate-on-boot	If set to `true`, a new log file will be created on server restart. The default is `false`.
rotate-size	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. For example, `50m` for 50 megabytes.
suffix	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.

Table A.27. Periodic Size Log Handler Attributes

Attribute	Description
append	If set to `true`, 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.
autoflush	If set to `true`, the log messages will be sent to the handlers assigned file immediately upon receipt.
enabled	If set to `true`, the handler is enabled and functioning as normal. If set to `false`, the handler is ignored when processing log messages.
encoding	The character encoding scheme to be used for the output.
file	Object that represents the file where the output of this log handler is written to. It has two configuration properties, `relative-to` and `path`.

Attribute	Description
filter-spec	An expression value that defines a filter. The following expression defines a filter that does not match a pattern: `not(match("WFLY.*"))`
formatter	The log formatter used by this log handler.
level	The lowest level of log message the log handler records.
max-backup-index	The maximum number of rotated logs that are kept. When this number is reached, the oldest log is reused. The default is `1`. 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`.
name	The name of the log handler. *Deprecated since the handler's address contains the name.*
named-formatter	The name of the defined formatter to be used on the handler.
rotate-on-boot	If set to `true`, a new log file will be created on server restart. The default is `false`.
rotate-size	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. For example, `50m` for 50 megabytes.
suffix	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.

Table A.28. Syslog Handler Attributes

Attribute	Description
app-name	The app name used when formatting the message in RFC5424 format. By default the app name is `java`.
enabled	If set to `true`, the handler is enabled and functioning as normal. If set to `false`, the handler is ignored when processing log messages.
facility	The facility as defined by RFC-5424 and RFC-3164.
hostname	The name of the host from which the messages are being sent. For example, the name of the host the application server is running on.

Attribute	Description
level	The lowest level of log message the log handler records.
port	The port on which the syslog server is listening.
server-address	The address of the syslog server.
syslog-format	Formats the log message according to the RFC specification.

Table A.29. Custom Log Handler Attributes

Attribute	Description
class	The logging handler class to be used.
enabled	If set to `true`, the handler is enabled and functioning as normal. If set to `false`, the handler is ignored when processing log messages.
encoding	The character encoding scheme to be used for the output.
filter	Defines a simple filter type. *Deprecated in favor of* ***`filter-spec`***.
filter-spec	An expression value that defines a filter. The following expression defines a filter that does not match a pattern: `not(match("WFLY.*"))`
formatter	The log formatter used by this log handler.
level	The lowest level of log message the log handler records.
module	The module one which the logging handler depends.
name	The name of the log handler. *Deprecated since the handler's address contains the name.*
named-formatter	The name of the defined formatter to be used on the handler.
properties	The properties used for the logging handler.

Table A.30. Async Log Handler Attributes

Attribute	Description
enabled	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	Defines a simple filter type. *Deprecated in favor of* ***`filter-spec`***.

Attribute	Description
filter-spec	An expression value that defines a filter. The following expression defines a filter that does not match a pattern: `not(match("WFLY.*"))`
level	The lowest level of log message the log handler records.
name	The name of the log handler. *Deprecated since the handler's address contains the name.*
overflow-action	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 an non-async log handler. **DISCARD** allows the logging application to continue but the log message is deleted.
queue-length	Maximum number of log messages that will be held by this handler while waiting for sub-handlers to respond.
subhandlers	The list of log handlers to which this async handler passes its log messages.

A.16. DATASOURCE CONNECTION URLS

Table A.31. Datasource Connection URLs

Datasource	Connection URL
IBM DB2	jdbc:db2://SERVER_NAME:PORT/DATABASE_NAME
MariaDB	jdbc:mariadb://SERVER_NAME:PORT/DATABASE_NAME
Microsoft SQL Server	jdbc:sqlserver://SERVER_NAME:PORT;DatabaseName=DATABASE_NAME
MySQL	jdbc:mysql://SERVER_NAME:PORT/DATABASE_NAME
Oracle	jdbc:oracle:thin:@SERVER_NAME:PORT:ORACLE_SID
PostgreSQL	jdbc:postgresql://SERVER_NAME:PORT/DATABASE_NAME
Sybase	jdbc:sybase:Tds:SERVER_NAME:PORT/DATABASE_NAME

A.17. DATASOURCE PARAMETERS

Table A.32. Datasource Parameters

Parameter	Datasource Type	Description
allocation-retry	Non-XA, XA	The number of times that allocating a connection should be tried before throwing an exception. The default is **0**, so an exception is thrown upon the first failure.
allocation-retry-wait-millis	Non-XA, XA	The amount of time, in milliseconds, to wait between retrying to allocate a connection. The default is **5000** ms.
allow-multiple-users	Non-XA, XA	Whether multiple users will access the datasource through the **getConnection(user, password)** method and if the internal pool type accounts for this behavior.
background-validation	Non-XA, XA	Whether connections should be validated on a background thread versus being validated prior to use. Background validation is typically not to be used with **validate-on-match** or there will be redundant checks. With background validation, there is an opportunity for a connection to go bad between the time of the validations can and being handed to the client, so the application must account for this possibility.
background-validation-millis	Non-XA, XA	The frequency, in milliseconds, that background validation will run.
blocking-timeout-wait-millis	Non-XA, XA	The maximum time, in milliseconds, to block while waiting for a connection before throwing an exception. Note that this blocks only while waiting for locking a connection, and will never throw an exception if creating a new connection takes an inordinately long time.
capacity-decrementer-class	Non-XA, XA	Class defining the policy for decrementing connections in the pool.
capacity-decrementer-properties	Non-XA, XA	Properties to be injected in the class defining the policy for decrementing connections in the pool.
capacity-incrementer-class	Non-XA, XA	Class defining the policy for incrementing connections in the pool.
capacity-incrementer-properties	Non-XA, XA	Properties to be injected in the class defining the policy for incrementing connections in the pool.
check-valid-connection-sql	Non-XA, XA	An SQL statement to check validity of a pool connection. This may be called when a managed connection is obtained from the pool.

Parameter	Datasource Type	Description
connectable	Non-XA, XA	Enable the use of CMR, which means that a local resource can reliably participate in an XA transaction.
connection-listener-class	Non-XA, XA	Specifies class name extending **`org.jboss.jca.adapters.jdbc.spi.listener.ConnectionListener`**. This class listens for connection activation and passivation in order to perform actions before the connection is returned to the application or to the pool. The specified class must be bundled together with the JDBC driver in one module using two resource jars (Install a JDBC Driver as a Core Module), or in separate global module (Define Global Modules).
connection-listener-property	Non-XA, XA	Properties to be injected into the class specified in the **`connection-listener-class`**. The properties injected are compliant with the JavaBeans conventions. For example, if you specify a property named **`foo`**, then the connection listener class needs to have a method **`setFoo`** that accepts **`String`** as argument.
connection-properties	Non-XA Only	Arbitrary string name/value pair connection properties to pass to the **`Driver.connect(url, props)`** method.
connection-url	Non-XA Only	The JDBC driver connection URL.
datasource-class	Non-XA Only	The fully-qualified name of the JDBC datasource class.
driver-class	Non-XA Only	The fully-qualified name of the JDBC driver class.
driver-name	Non-XA, XA	Defines the JDBC driver the datasource should use. It is a symbolic name matching the name of installed driver. If the driver is deployed as JAR, the name is the name of the deployment.
enabled	Non-XA, XA	Whether the datasource should be enabled.
enlistment-trace	Non-XA, XA	Whether enlistment traces should be recorded.
exception-sorter-class-name	Non-XA, XA	An instance of **`org.jboss.jca.adapters.jdbc.ExceptionSorter`** that provides a method to validate if an exception should broadcast an error.

Parameter	Datasource Type	Description
exception-sorter-properties	Non-XA, XA	The exception sorter properties.
flush-strategy	Non-XA, XA	Specifies how the pool should be flushed in case of an error. Valid values are: **`FailingConnectionOnly`** Only the failing connection is removed. This is the default setting. **`InvalidIdleConnections`** The failing connection and idle connections that share the same credentials and are returned as invalid by the ValidatingManagedConnectionFactory.getInvalidConnections(...) method are removed. **`IdleConnections`** The failing connection and idle connections that share the same credentials are removed. **`Gracefully`** The failing connection and idle connections that share the same credentials are removed. Active connections that share the same credentials are destroyed upon return to the pool. **`EntirePool`** The failing connection and idle and active connections that share the same credentials are removed. This setting is not recommended for production systems. **`AllInvalidIdleConnections`** The failing connection and idle connections that are returned as invalid by the ValidatingManagedConnectionFactory.getInvalidConnections(...) method are removed. **`AllIdleConnections`** The failing connection and all idle connections are removed. **`AllGracefully`** The failing connection and all idle connections are removed. Active connections are destroyed upon return to the pool. **`AllConnections`** The failing connection and all idle and active connections are removed. This setting is not recommended for production systems.
idle-timeout-minutes	Non-XA, XA	The maximum time, in minutes, a connection may be idle before being closed. If not specified, the default is **`30`** minutes. The actual maximum time also depends on the IdleRemover scan time, which is half of the smallest idle-timeout-minutes value of any pool.

Parameter	Datasource Type	Description
initial-pool-size	Non-XA, XA	The initial number of connections a pool should hold.
interleaving	XA Only	Whether to enable interleaving for XA connections.
jndi-name	Non-XA, XA	The unique JNDI name for the datasource.
jta	Non-XA Only	Enable JTA integration.
max-pool-size	Non-XA, XA	The maximum number of connections that a pool can hold.
mcp	Non-XA, XA	The `ManagedConnectionPool` implementation. For example, `org.jboss.jca.core.connectionmanager.pool.mcp.SemaphoreArrayListManagedConnectionPool`
min-pool-size	Non-XA, XA	The minimum number of connections that a pool can hold.
new-connection-sql	Non-XA, XA	An SQL statement to execute whenever a connection is added to the connection pool.
no-recovery	XA Only	Whether the connection pool should be excluded from recovery.
no-tx-separate-pool	XA Only	Whether to create a separate sub-pool for each context. This may be required for some Oracle datasources, which may not allow XA connections to be used both inside and outside of a JTA transaction. Using this option will cause your total pool size to be twice the `max-pool-size`, because two actual pools will be created.
pad-xid	XA Only	Whether to pad the Xid.
password	Non-XA, XA	The password to use when creating a new connection.
pool-fair	Non-XA, XA	Defines if pool should be fair. This setting is part of a `Semaphore` class used to manage the connection pools in JCA, which provides a performance benefit in some use cases where the order of leasing connections is not required.
pool-prefill	Non-XA, XA	Whether the pool should be prefilled.

Parameter	Datasource Type	Description
pool-use-strict-min	Non-XA, XA	Whether **`min-pool-size`** should be considered strictly.
prepared-statements-cache-size	Non-XA, XA	The number of prepared statements per connection in a Least Recently Used (LRU) cache.
query-timeout	Non-XA, XA	The timeout for queries, in seconds. The default is no timeout.
reauth-plugin-class-name	Non-XA, XA	The fully-qualified class name of the reauthentication plugin implementation to reauthenticate physical connections.
reauth-plugin-properties	Non-XA, XA	The properties for the reauthentication plugin.
recovery-password	XA Only	The password to use to connect to the resource for recovery.
recovery-plugin-class-name	XA Only	The fully-qualified class name of the recovery plugin implementation.
recovery-plugin-properties	XA Only	The properties for the recovery plugin.
recovery-security-domain	XA Only	The security domain to use to connect to the resource for recovery.
recovery-username	XA Only	The user name to use to connect to the resource for recovery.
same-rm-override	XA Only	Whether the **`javax.transaction.xa.XAResource.isSameRM(XAResource)`** class returns **`true`** or **`false`**.
security-domain	Non-XA, XA	The name of a JAAS security-manager which handles authentication. This name correlates to the application-policy/name attribute of the JAAS login configuration.
set-tx-query-timeout	Non-XA, XA	Whether to set the query timeout based on the time remaining until transaction timeout. Any configured query timeout will be used if no transaction exists.
share-prepared-statements	Non-XA, XA	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`**.

Parameter	Datasource Type	Description
spy	Non-XA, XA	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.
stale-connection-checker-class-name	Non-XA, XA	An instance of **`org.jboss.jca.adapters.jdbc.StaleConnectionChecker`** that provides an **`isStaleConnection(SQLException)`** method. If this method returns **`true`**, then the exception is wrapped in an **`org.jboss.jca.adapters.jdbc.StaleConnectionException`**.
stale-connection-checker-properties	Non-XA, XA	The stale connection checker properties.
statistics-enabled	Non-XA, XA	Whether runtime statistics are enabled. The default is **`false`**.
track-statements	Non-XA, XA	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. Valid values: • **`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 (*default*).
tracking	Non-XA, XA	Whether to track connection handles across transaction boundaries.
transaction-isolation	Non-XA, XA	The **`java.sql.Connection`** transaction isolation level. Valid values: • **`TRANSACTION_READ_UNCOMMITTED`** • **`TRANSACTION_READ_COMMITTED`** • **`TRANSACTION_REPEATABLE_READ`** • **`TRANSACTION_SERIALIZABLE`** • **`TRANSACTION_NONE`**

Parameter	Datasource Type	Description
url-delimiter	Non-XA, XA	The delimiter for URLs in connection-url for High Availability (HA) datasources.
url-property	XA Only	The property for the **URL** property in the **`xa-datasource-property`** values.
url-selector-strategy-class-name	Non-XA, XA	A class that implements **`org.jboss.jca.adapters.jdbc.URLSelectorStrategy`**.
use-ccm	Non-XA, XA	Enable the cached connection manager.
use-fast-fail	Non-XA, XA	If true, fail a connection allocation on the first attempt if the connection is invalid. If false, keep trying until the pool is exhausted.
use-java-context	Non-XA, XA	Whether to bind the datasource into global JNDI.
use-try-lock	Non-XA, XA	A timeout value for internal locks. This attempts to obtain the lock for the configured number of seconds, before timing out, rather than failing immediately if the lock is unavailable. Uses **`tryLock()`** instead of **`lock()`**.
user-name	Non-XA, XA	The user name to use when creating a new connection.
valid-connection-checker-class-name	Non-XA, XA	An implementation of **`org.jboss.jca.adaptors.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.
valid-connection-checker-properties	Non-XA, XA	The valid connection checker properties.
validate-on-match	Non-XA, XA	Whether connection validation is performed when a connection factory attempts to match a managed connection. This should be used when a client must have a connection validated prior to use. Validate-on-match is typically not to be used with **`background-validation`** or there will be redundant checks.

Parameter	Datasource Type	Description
wrap-xa-resource	XA Only	Whether to wrap the XAResource in an `org.jboss.tm.XAResourceWrapper` instance.
xa-datasource-class	XA Only	The fully-qualified name of the `javax.sql.XADataSource` implementation class.
xa-datasource-properties	XA Only	String name/value pair of XA datasource properties.
xa-resource-timeout	XA Only	If non-zero, this value is passed to the `XAResource.setTransactionTimeout` method.

A.18. DATASOURCE STATISTICS

Table A.33. Core Pool 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. This value is in milliseconds.
AverageCreationTime	The average time spent creating a connection. This value is in milliseconds.
AverageGetTime	The average time spent obtaining a connection.
AverageUsageTime	The average time spent using a connection.
BlockingFailureCount	The number of failures trying to obtain a connection.
CreatedCount	The number of connections created.
DestroyedCount	The number of connections destroyed.
IdleCount	The number of connections that are currently idle.
InUseCount	The number of connections currently in use.

Name	Description
MaxCreationTime	The maximum time it took to create a connection. This value is in milliseconds.
MaxGetTime	The maximum time for obtaining a connection.
MaxPoolTime	The maximum time for a connection in the pool.
MaxUsageTime	The maximum time using a connection.
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. This value is in milliseconds.
TotalCreationTime	The total time spent creating connections. This value is in milliseconds.
TotalGetTime	The total time spent obtaining connections.
TotalPoolTime	The total time spent by connections in the pool.
TotalUsageTime	The total time spent using connections.
WaitCount	The number of requests that had to wait to obtain a connection.
XACommitAverageTime	The average time for an XAResource commit invocation.
XACommitCount	The number of XAResource commit invocations.
XACommitMaxTime	The maximum time for an XAResource commit invocation.
XACommitTotalTime	The total time for all XAResource commit invocations.
XAEndAverageTime	The average time for an XAResource end invocation.
XAEndCount	The number of XAResource end invocations.

Name	Description
XAEndMaxTime	The maximum time for an XAResource end invocation.
XAEndTotalTime	The total time for all XAResource end invocations.
XAForgetAverageTime	The average time for an XAResource forget invocation.
XAForgetCount	The number of XAResource forget invocations.
XAForgetMaxTime	The maximum time for an XAResource forget invocation.
XAForgetTotalTime	The total time for all XAResource forget invocations.
XAPrepareAverageTime	The average time for an XAResource prepare invocation.
XAPrepareCount	The number of XAResource prepare invocations.
XAPrepareMaxTime	The maximum time for an XAResource prepare invocation.
XAPrepareTotalTime	The total time for all XAResource prepare invocations.
XARecoverAverageTime	The average time for an XAResource recover invocation.
XARecoverCount	The number of XAResource recover invocations.
XARecoverMaxTime	The maximum time for an XAResource recover invocation.
XARecoverTotalTime	The total time for all XAResource recover invocations.
XARollbackAverageTime	The average time for an XAResource rollback invocation.
XARollbackCount	The number of XAResource rollback invocations.
XARollbackMaxTime	The maximum time for an XAResource rollback invocation.
XARollbackTotalTime	The total time for all XAResource rollback invocations.
XAStartAverageTime	The average time for an XAResource start invocation.
XAStartCount	The number of XAResource start invocations.
XAStartMaxTime	The maximum time for an XAResource start invocation.
XAStartTotalTime	The total time for all XAResource start invocations.

Table A.34. 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.

A.19. TRANSACTION MANAGER CONFIGURATION OPTIONS

Table A.35. Transactions Subsystem Attributes

Attribute	Description
default-timeout	The default transaction timeout. This defaults to **`300`** seconds. You can override this programmatically, on a per-transaction basis.
enable-statistics	*Deprecated in favor of* ***`statistics-enabled`***.
enable-tsm-status	Whether to enable the transaction status manager (TSM) service, which is used for out-of-process recovery. This option is not supported, as running an out-of-process recovery manager to contact the **`ActionStatusService`** from a different process, instead of in memory, is not supported.
hornetq-store-enable-async-io	*Deprecated in favor of* ***`journal-store-enable-async-io`***.
jdbc-action-store-drop-table	Whether JDBC action store should drop tables. The default is **`false`**.
jdbc-action-store-table-prefix	Optional prefix for table used to write transaction logs in configured JDBC action store.
jdbc-communication-store-drop-table	Whether JDBC communication store should drop tables. The default is **`false`**.
jdbc-communication-store-table-prefix	Optional prefix for table used to write transaction logs in configured JDBC communication store.
jdbc-state-store-drop-table	Whether JDBC state store should drop tables. The default is **`false`**.

Attribute	Description
jdbc-state-store-table-prefix	Optional prefix for table used to write transaction logs in configured JDBC state store.
jdbc-store-datasource	JNDI name of non-XA datasource used. Datasource should be defined in the **`datasources`** subsystem.
journal-store-enable-async-io	Whether **`AsyncIO`** should be enabled for the journal store or not. Defaults to **`false`**. The server should be restarted for this setting to take effect.
jts	Whether to use Java Transaction Service (JTS) transactions. Defaults to **`false`**, which uses JTA transactions only.
node-identifier	The node identifier for the transaction manager. If this option is not set, you will see a warning upon server startup. This option is required in the following situations: • For JTS to JTS communications • When two transaction managers access shared resource managers • When two transaction managers access shared object stores 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.
object-store-path	A relative or absolute file system path where the transaction manager object store stores data. By default relative to the **`object-store-relative-to`** parameter's value. If**`object-store-relative-to`** is set to an empty string, this value is treated as an absolute path.
object-store-relative-to	References a global path configuration in the domain model. The default value is the data directory for JBoss EAP, 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`** transaction manager attribute is relative to this path. Set this attribute to an empty string to have **`object-store-path`** be treated as an absolute path.
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.

Attribute	Description
process-id-socket-max-ports	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. 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 transaction manager will try before failing. The default value is **10**.
process-id-uuid	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**. See **process-id-socket-max-ports** for more information. To enable **process-id-socket-binding**, set **process-id-uuid** to **false**.
recovery-listener	Whether or not the transaction recovery process should listen on a network socket. Defaults to **false**.
socket-binding	Specifies the name of the socket binding used by the transaction periodic recovery listener when **recovery-listener** is set to **true**.
statistics-enabled	Whether statistics should be enabled. The default is **false**.
status-socket-binding	Specifies the socket binding to use for the transaction status manager. This configuration option is not supported.
use-hornetq-store	*Deprecated in favor of **use-journal-store**.*
use-jdbc-store	Use the JDBC store for writing transaction logs. Set to **true** to enable and to **false** to use the default log store type.
use-journal-store	Use Apache ActiveMQ Artemis 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.

Table A.36. Log Store Attributes

Attribute	Description
expose-all-logs	Whether to expose all logs. The default is **false**, meaning that only a subset of transaction logs is exposed.

Attribute	Description
type	Specifies the implementation type of the logging store. The default is **default**.

Table A.37. Commit Markable Resource Attributes

Attribute	Description
batch-size	The batch size for this CMR resource. The default is **100**.
immediate-cleanup	Whether to perform immediate cleanup for this CMR resource. The default is **true**.
jndi-name	The JNDI name of this CMR resource.
name	The table name for storing XIDs. The default is **xids**.

A.20. IIOP SUBSYSTEM ATTRIBUTES

Table A.38. IIOP Subsystem Attributes

Attribute	Description
add-component-via-interceptor	Indicates whether SSL components should be added by an IOR interceptor.
auth-method	The authentication method. Valid values are **none** and **username_password**.
caller-propagation	Indicates whether the caller identity should be propagated in the SAS context. Valid values are **none** and **supported**.
client-requires	Value that indicates the client SSL required parameters. Valid values are **None**, **ServerAuth**, **ClientAuth**, and **MutualAuth**.
client-supports	Value that indicates the client SSL supported parameters. Valid values are **None**, **ServerAuth**, **ClientAuth**, and **MutualAuth**.
confidentiality	Indicates whether the transport must require confidentiality protection or not. Valid values are **none**, **supported**, and **required**.
detect-misordering	Indicates whether the transport must require misordering detection or not. Valid values are **none**, **supported**, and **required**.

Attribute	Description
detect-replay	Indicates whether the transport must require replay detection or not. Valid values are **none**, **supported**, and **required**.
export-corbaloc	Indicates whether the root context should be exported as **corbaloc::address:port/NameService**.
giop-version	The GIOP version to be used.
high-water-mark	TCP connection cache parameter. Each time the number of connections exceeds this value, the ORB tries to reclaim connections. The number of reclaimed connections is specified by the **number-to-reclaim** property. If this property is not set, then the OpenJDK ORB default is used.
integrity	Indicates whether the transport must require integrity protection or not. Valid values are **none**, **supported**, and **required**.
number-to-reclaim	TCP connection cache parameter. Each time the number of connections exceeds the **high-water-mark** property, then the ORB tries to reclaim connections. The number of reclaimed connections is specified by this property. If it is not set, then the OpenJDK ORB default is used.
persistent-server-id	Persistent ID of the server. Persistent object references are valid across many activations of the server and they identify it using this property. As a result of that, many activations of the same server should have this property set to the same value, and different server instances running on the same host should have different server IDs.
properties	A list of generic key/value properties.
realm	The authentication service realm name.
required	Indicates whether authentication is required.
root-context	The naming service root context.
security	Indicates whether the security interceptors are to be installed. Valid values are **client**, **identity**, and **none**.
security-domain	The name of the security domain that holds the key and trust stores that will be used to establish SSL connections.
server-requires	Value that indicates the server SSL required parameters. Valid values are **None**, **ServerAuth**, **ClientAuth**, and **MutualAuth**.

Attribute	Description
server-supports	Value that indicates the server SSL supported parameters. Valid values are **None**, **ServerAuth**, **ClientAuth**, and **MutualAuth**.
socket-binding	The name of the socket binding configuration that specifies the ORB port.
ssl-socket-binding	The name of the socket binding configuration that specifies the ORB SSL port.
support-ssl	Indicates whether SSL is supported.
transactions	Indicates whether the transactions interceptors are to be installed or not. Valid values are **full**, **spec**, and **none**. A value of **full** enables JTS while a value of **spec** enables a non-JTS spec-compliant mode that rejects incoming transaction contexts.
trust-in-client	Indicates if the transport must require trust in client to be established. Valid values are **none**, **supported**, and **required**.
trust-in-target	Indicates if the transport must require trust in target to be established. Valid values are **none** and **supported**.

A.21. RESOURCE ADAPTER ATTRIBUTES

The following tables describe the resource adapter attributes.

Table A.39. Main Attributes

Attribute	Description
archive	The resource adapter archive.
beanvalidationgroups	The bean validation groups that should be used.
bootstrap-context	The unique name of the bootstrap context that should be used.
config-properties	Custom defined config properties.
module	The module from which the resource adapter will be loaded.
statistics-enabled	Whether runtime statistics are enabled or not.

Attribute	Description
transaction-support	The transaction support level of the resource adapter.
wm-security	Toggle on/off `wm.security` for this resource adapter. In case of false, all `wm-security-*` parameters are ignored, even the defaults.
wm-security-default-groups	A default groups list that should be added to the used `Subject` instance.
wm-security-default-principal	A default principal name that should be added to the used `Subject` instance.
wm-security-domain	The name of the security domain that should be used.
wm-security-mapping-groups	List of groups mappings.
wm-security-mapping-required	Defines if a mapping is required for security credentials.
wm-security-mapping-users	List of user mappings.

Table A.40. admin-objects Attributes

Attribute	Description
class-name	The fully qualified class name of an administration object.
enabled	Specifies if the administration object should be enabled.
jndi-name	The JNDI name for the administration object.
use-java-context	Setting this to false will bind the object into global JNDI.

Table A.41. connection-definitions Attributes

Attribute	Description
allocation-retry	Indicates the number of times that allocating a connection should be tried before throwing an exception.
allocation-retry-wait-millis	The amount of time, in milliseconds, to wait between retrying to allocate a connection.
background-validation	Specifies that connections should be validated on a background thread versus being validated prior to use. Changing this value requires a server restart.

Attribute	Description
background-validation-millis	The amount of time, in milliseconds, that background validation will run. Changing this value requires a server restart.
blocking-timeout-wait-millis	The maximum time, in milliseconds, to block while waiting for a connection before throwing an exception. Note that this blocks only while waiting for locking a connection, and will never throw an exception if creating a new connection takes an inordinately long time.
capacity-decrementer-class	Class defining the policy for decrementing connections in the pool.
capacity-decrementer-properties	Properties to inject in class defining the policy for decrementing connections in the pool.
capacity-incrementer-class	Class defining the policy for incrementing connections in the pool.
capacity-incrementer-properties	Properties to inject in class defining the policy for incrementing connections in the pool.
class-name	The fully qualified class name of a managed connection factory or admin object.
connectable	Enable the use of CMR. This feature means that a local resource can reliably participate in an XA transaction.
enabled	Specifies if the resource adapter should be enabled.
enlistment	Specifies if lazy enlistment should be used if supported by the resource adapter.
enlistment-trace	Specifies if JBoss EAP/IronJacamar should record enlistment traces.

Attribute	Description
flush-strategy	Specifies how the pool should be flushed in case of an error. Valid values are: **`FailingConnectionOnly`** Only the failing connection is removed. This is the default setting. **`InvalidIdleConnections`** The failing connection and idle connections that share the same credentials and are returned as invalid by the ValidatingManagedConnectionFactory.getInvalidConnections(...) method are removed. **`IdleConnections`** The failing connection and idle connections that share the same credentials are removed. **`Gracefully`** The failing connection and idle connections that share the same credentials are removed. Active connections that share the same credentials are destroyed upon return to the pool. **`EntirePool`** The failing connection and idle and active connections that share the same credentials are removed. This setting is not recommended for production systems. **`AllInvalidIdleConnections`** The failing connection and idle connections that are returned as invalid by the ValidatingManagedConnectionFactory.getInvalidConnections(...) method are removed. **`AllIdleConnections`** The failing connection and all idle connections are removed. **`AllGracefully`** The failing connection and all idle connections are removed. Active connections are destroyed upon return to the pool. **`AllConnections`** The failing connection and all idle and active connections are removed. This setting is not recommended for production systems.
idle-timeout-minutes	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 half of the smallest **`idle-timeout-minutes`** value of any pool. Changing this value requires a server restart.
initial-pool-size	The initial number of connections a pool should hold.
interleaving	Specifies whether to enable interleaving for XA connections.
jndi-name	The JNDI name for the connection factory.
max-pool-size	The maximum number of connections for a pool. No more connections will be created in each sub-pool.

Attribute	Description
mcp	The `ManagedConnectionPool` implementation. For example: `org.jboss.jca.core.connectionmanager.pool.mcp.SemaphoreArrayListManagedConnectionPool`.
min-pool-size	The minimum number of connections for a pool.
no-recovery	Specifies if the connection pool should be excluded from recovery.
no-tx-separate-pool	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	Specifies whether the Xid should be padded.
pool-fair	Specifies if pool use should be fair.
pool-prefill	Specifies if the pool should be prefilled. Changing this value requires a server restart.
pool-use-strict-min	Specifies if the `min-pool-size` should be considered strict.
recovery-password	The password used for recovery.
recovery-plugin-class-name	The fully qualified class name of the recovery plugin implementation.
recovery-plugin-properties	The properties for the recovery plugin.
recovery-security-domain	The security domain used for recovery.
recovery-username	The user name used for recovery.
same-rm-override	Unconditionally set whether `javax.transaction.xa.XAResource.isSameRM(XAResource)` returns true or false.
security-application	Indicates that application-supplied parameters, such as from `getConnection(user, pw)`, are used to distinguish connections in the pool.
security-domain	The security domain which defines the `javax.security.auth.Subject` that is used to distinguish connections in the pool.
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.

Attribute	Description
sharable	Enable the use of sharable connections, which allows lazy association to be enabled if supported.
tracking	Specifies if IronJacamar should track connection handles across transaction boundaries.
use-ccm	Enable the use of a cached connection manager.
use-fast-fail	Whether to fail a connection allocation on the first try if it is invalid (`true`), or keep trying until the pool is exhausted of all potential connections (`false`).
use-java-context	Setting this to `false` will bind the object into global JNDI.
validate-on-match	Specifies if connection validation should be done when a connection factory attempts to match a managed connection. This is typically exclusive to the use of background validation.
wrap-xa-resource	Specifies whether `XAResource` instances should be wrapped in an `org.jboss.tm.XAResourceWrapper` instance.
xa-resource-timeout	The value is passed to `XAResource.setTransactionTimeout()`, in seconds. The default is `0`.

The resource adapter schema can be found at `EAP_HOME/docs/schema/wildfly-resource-adapters_4_0.xsd`.

A.22. RESOURCE ADAPTER STATISTICS

Table A.42. Resource Adapter 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.

Name	Description
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.

A.23. UNDERTOW SUBSYSTEM ATTRIBUTES

Table A.43. undertow Attributes

Attribute	Default	Description
default-security-domain	other	The default security domain used by web deployments.
default-server	default-server	The default server to use for deployments.
default-servlet-container	default	The default servlet container to use for deployments.
default-virtual-host	default-host	The default virtual host to use for deployments.
instance-id	${jboss.node.name}	The cluster instance ID.
statistics-enabled	false	Whether statistics are enabled.

Buffer Cache Attributes

Table A.44. buffer-cache Attributes

Attribute	Default	Description
buffer-size	1024	The size of the buffers. Smaller buffers allow space to be utilized more effectively.
buffers-per-region	1024	The numbers of buffers per region.
max-regions	10	The maximum number of regions. This controls the maximum amount of memory that can be used for caching.

Servlet Container Attributes

The servlet container component has the following structure:

- servlet-container
  - mime-mapping
  - welcome-file
  - crawler-session-management (part of settings)
  - jsp (part of settings)
  - persistent-sessions (part of settings)
  - session-cookie (part of settings)
  - websockets (part of settings)

servlet-container Attributes

Table A.45. servlet-container Attributes

Attribute	Default	Description
allow-non-standard-wrappers	false	Whether request and response wrappers that do not extend the standard wrapper classes can be used.
default-buffer-cache	default	The buffer cache to use for caching static resources.
default-encoding		Default encoding to use for all deployed applications.
default-session-timeout	30	The default session timeout in minutes for all applications deployed in the container.
directory-listing		If directory listing should be enabled for default servlets.

Attribute	Default	Description
disable-caching-for-secured-pages	true	Whether to set headers to disable caching for secured paged. Disabling this can cause security problems, as sensitive pages may be cached by an intermediary.
eager-filter-initialization	false	Whether to call filter init() on deployment start rather than when first requested.
ignore-flush	false	Ignore flushes on the servlet output stream. In most cases these just hurt performance for no good reason.
max-sessions		The maximum number of sessions that can be active at one time.
proactive-authentication	false	Whether proactive authentication should be used. If this is `true`, a user will always be authenticated if credentials are present.
session-id-length	30	The length of the generated session ID. Longer session ID's are more secure.
stack-trace-on-error	local-only	If an error page with the stack trace should be generated on error. Values are all, none and local-only.
use-listener-encoding	false	Use encoding defined on listener.

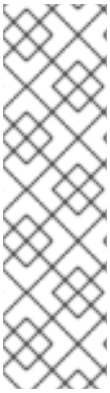
mime-mapping Attributes

Table A.46. mime-mapping Attributes

Attribute	Default	Description
value		The mime type for this mapping.

welcome-file Attributes
Defines a welcome file and has no options.

crawler-session-management Attributes
Configures special session handling for crawler bots.



NOTE

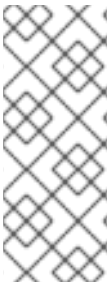
When using the management CLI to manage the **`crawler-session-management`** element, it is available under **`settings`** in the **`servlet-container`** element. For example:

```
/subsystem=undertow/servlet-container=default/setting=crawler-session-management:add
/subsystem=undertow/servlet-container=default/setting=crawler-session-management:read-resource
```

Table A.47. crawler-session-management Attributes

Attribute	Default	Description
session-timeout		The session timeout in seconds for sessions that are owned by crawlers.
user-agents		Regular expression that is used to match the user agent of a crawler.

jsp Attributes



NOTE

When using the management CLI to manage the **`jsp`** element, it is available under **`settings`** in the **`servlet-container`** element. For example:

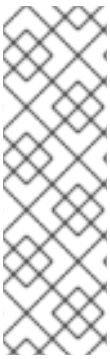
```
/subsystem=undertow/servlet-container=default/setting=jsp:read-resource
```

Table A.48. jsp Attributes

Attribute	Default	Description
check-interval	0	Check interval for JSP updates using a background thread.
development	false	Enable development mode which enables reloading JSP on-the-fly.
disabled	false	Enable the JSP container.
display-source-fragment	true	When a runtime error occurs, attempts to display corresponding JSP source fragment.
dump-smap	false	Write SMAP data to a file.

Attribute	Default	Description
error-on-use-bean-invalid-class-attribute	false	Enable errors when using a bad class in useBean.
generate-strings-as-char-arrays	false	Generate String constants as char arrays.
java-encoding	UTF8	Specify the encoding used for Java sources.
keep-generated	true	Keep the generated servlets.
mapped-file	true	Map to the JSP source.
modification-test-interval	4	Minimum amount of time between two tests for updates, in seconds.
optimize-scriptlets	false	If JSP scriptlets should be optimized to remove string concatenation.
recompile-on-fail	false	Retry failed JSP compilations on each request.
scratch-dir		Specify a different work directory.
smap	true	Enable SMAP.
source-vm	1.8	Source VM level for compilation.
tag-pooling	true	Enable tag pooling.
target-vm	1.8	Target VM level for compilation.
trim-spaces	false	Trim some spaces from the generated servlet.
x-powered-by	true	Enable advertising the JSP engine in x-powered-by.

persistent-sessions Attributes



NOTE

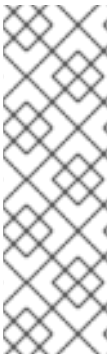
When using the management CLI to manage the **`persistent-sessions`** element, it is available under **`settings`** in the **`servlet-container`** element. For example:

```
/subsystem=undertow/servlet-
container=default/setting=persistent-sessions:add
/subsystem=undertow/servlet-
container=default/setting=persistent-sessions:read-resource
```

Table A.49. persistent-sessions Attributes

Attribute	Default	Description
path		The path to the persistent session data directory. If this is null, sessions will be stored in memory.
relative-to		The directory the path is relative to.

session-cookie Attributes



NOTE

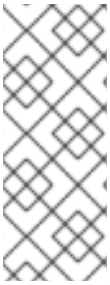
When using the management CLI to manage the **`session-cookie`** element, it is available under **`settings`** in the **`servlet-container`** element. For example:

```
/subsystem=undertow/servlet-container=default/setting=session-cookie:add
/subsystem=undertow/servlet-container=default/setting=session-cookie:read-resource
```

Table A.50. session-cookie Attributes

Attribute	Default	Description
comment		Cookie comment.
domain		Cookie domain.
http-only		Whether the cookie is http-only.
max-age		Maximum age of the cookie.
name		Name of the cookie.
secure		Whether the cookie is secure.

websockets Attributes



NOTE

When using the management CLI to manage the **`websockets`** element, it is available under **`settings`** in the **`servlet-container`** element. For example:

```
/subsystem=undertow/servlet-container=default/setting=websockets:read-resource
```

Table A.51. websockets Attributes

Attribute	Default	Description
buffer-pool	default	The buffer pool to use for websocket deployments.
dispatch-to-worker	true	Whether callbacks should be dispatched to a worker thread. If this is `false`, then they will be run in the IO thread, which is faster however care must be taken not to perform blocking operations.
worker	default	The worker to use for websocket deployments.

Filter Attributes

custom-filter Filters

Table A.52. custom-filter Attributes

Attribute	Default	Description
class-name		Class name of HttpHandler.
module		Module name where class can be loaded from.
parameters		Filter parameters.

error-page Filters

The error pages

Table A.53. error-page Attributes

Attribute	Default	Description
code		Error page code.
path		Error page path.

expression-filter Filters

A filter parsed from the Undertow expression language.

Table A.54. expression-filter Attributes

Attribute	Default	Description
expression		The expression that defines the filter.
module		Module to use to load the filter definitions.

gzip Filters

Defines the gzip filter and has no attributes.

mod-cluster Filters
The mod-cluster filter component has the following structure:

- mod-cluster
  - balancer
    - load-balancing-group
    - node
      - context

Table A.55. mod-cluster Attributes

Attribute	Default	Description
advertise-frequency	10000	The frequency in milliseconds that mod_cluster advertises itself on the network.
advertise-path	/	The path that mod_cluster is registered under.
advertise-protocol	http	The protocol that is in use.
advertise-socket-binding		The multicast group that is used to advertise.
broken-node-timeout	60000	The amount of time that must elapse before a broken node is removed from the table.
cached-connections-per-thread	5	The number of connections that will be kept alive indefinitely.
connection-idle-timeout	60	The amount of time a connection can be idle before it will be closed. Connections will not time out once the pool size is down to the configured minimum, which is configured by **`cached-connections-per-thread`**.
connections-per-thread	10	The number of connections that will be maintained to back-end servers, per IO thread.
enable-http2	false	Whether the load balancer should attempt to upgrade back-end connections to HTTP/2. If HTTP/2 is not supported, HTTP or HTTPS will be used as normal.
health-check-interval	10000	The frequency of health check pings to back-end nodes.

Attribute	Default	Description
management-access-predicate		A predicate that is applied to incoming requests to determine if they can perform mod cluster management commands. Provides additional security on top of what is provided by limiting management to requests that originate from the `management-socket-binding`.
management-socket-binding		The socket binding of the mod_cluster management port. When using mod_cluster two HTTP listeners should be defined, a public one to handle requests, and one bound to the internal network to handle mod cluster commands. This socket binding should correspond to the internal listener, and should not be publicly accessible.
max-request-time	-1	The maximum amount of time that a request to a back-end node can take before it is killed.
request-queue-size	10	The number of requests that can be queued if the connection pool is full before requests are rejected with a 503.
security-key		The security key that is used for the mod_cluster group. All members must use the same security key.
security-realm		The security realm that provides the SSL configuration.
use-alias	false	Whether an alias check is performed.
worker	default	The XNIO worker that is used to send the advertise notifications.

Table A.56. balancer Attributes

Attribute	Default	Description
max-attempts		The number of attempts to send the request to a back-end server.
sticky-session		If sticky sessions are enabled.
sticky-session-cookie		The session cookie name.

Attribute	Default	Description
sticky-session-force		If this is `true`, then an error will be returned if the request cannot be routed to the sticky node, otherwise it will be routed to another node.
sticky-session-path		The path of the sticky session cookie.
sticky-session-remove		Remove the session cookie if the request cannot be routed to the correct host.
wait-worker		The number of seconds to wait for an available worker.

load-balancing-group Attributes

Defines a load balancing group and has no options.

Table A.57. node Attributes

Attribute	Default	Description
aliases		The nodes aliases.
cache-connections		The number of connections to keep alive indefinitely.
elected		The elected count.
flush-packets		If received data should be immediately flushed.
load		The current load of this node.
load-balancing-group		The load balancing group this node belongs to.
max-connections		The maximum number of connections per IO thread.
open-connections		The current number of open connections.
ping		The nodes ping.
queue-new-requests		If a request is received and there is no worker immediately available should it be queued.
read		The number of bytes read from the node.
request-queue-size		The size of the request queue.
status		The current status of this node.

Attribute	Default	Description
timeout		The request timeout.
ttl		The time connections will stay alive with no requests before being closed, if the number of connections is larger than `cache-connections`.
uri		The URI that the load balancer uses to connect to the node.
written		The number of bytes transferred to the node.

Table A.58. context Attributes

Attribute	Default	Description
requests		The number of requests against this context.
status		The status of this context.

request-limit Filters

Table A.59. request-limit Attributes

Attribute	Default	Description
max-concurrent-requests		Maximum number of concurrent requests.
queue-size		Number of requests to queue before they start being rejected.

response-header Filters
Response header filter allows you to add custom headers.

Table A.60. response-header Attributes

Attribute	Default	Description
header-name		The header name.
header-value		The header value.

rewrite Filters

Table A.61. rewrite Attributes

Attribute	Default	Description
redirect	false	Whether a redirect will be done instead of a rewrite.
target		The expression that defines the target. If you are redirecting to a constant target put single quotes around the value.

Handler Attributes
file Attributes

Table A.62. file Attributes

Attribute	Default	Description
cache-buffer-size	1024	Size of the buffers.
cache-buffers	1024	Number of buffers.
case-sensitive	true	Whether to use case-sensitive file handling. Note that setting this to `false` for case insensitivity will only work if the underlying file system is case insensitive.
directory-listing	false	Whether to enable directory listing.
follow-symlink	false	Whether to enable following symbolic links.
path		Path on the file system from where file handler will serve resources.
safe-symlink-paths		Paths that are safe to be targets of symbolic links.

Using WebDAV for Static Resources
Previous versions of JBoss EAP allowed for using WebDAV with the **web** subsystem, by way of the **`WebdavServlet`**, to host static resources and enable additional HTTP methods for accessing and manipulating those files. In JBoss EAP 7, the **`undertow`** subsystem does provide a mechanism for serving static files using a file handler, but the **`undertow`** subsystem does not support WebDAV. If you want to use WebDAV with JBoss EAP 7, you can write a custom WebDAV servlet.

reverse-proxy attributes
The reverse-proxy handler component has the following structure:

- reverse-proxy
  - host

Table A.63. reverse-proxy Attributes

Attribute	Default	Description
cached-connections-per-thread	5	The number of connections that will be kept alive indefinitely.
connection-idle-timeout	60	The amount of time a connection can be idle before it will be closed. Connections will not time out once the pool size is down to the configured minimum (as configured by cached-connections-per-thread).
connections-per-thread	10	The number of connections that will be maintained to back-end servers, per IO thread.
max-request-time	-1	The maximum time that a proxy request can be active for, before being killed. Defaults to unlimited.
problem-server-retry	30	Time in seconds to wait before attempting to reconnect to a server that is down.
request-queue-size	10	The number of requests that can be queued if the connection pool is full before requests are rejected with a 503.
session-cookie-names	JSESSIONID	Comma-separated list of session cookie names. Generally this will just be JSESSIONID.

Table A.64. host Attributes

Attribute	Default	Description
instance-id		The instance ID, or JVM route, that will be used to enable sticky sessions.
outbound-socket-binding		Outbound socket binding for this host.
path	/	Optional path if host is using non root resource.
scheme	http	The kind of scheme that is used.
security-realm		The security realm that provides the SSL configuration for the connection to the host.

Server Attributes

The server component has the following structure:

- server
  - http-listener

- https-listener
- ajp-listener
- host
  - access-log (part of settings)
  - single-sign-on (part of settings)
  - filter-ref
  - location
    - filter-ref

server Attributes

Table A.65. server Attributes

Attribute	Default	Description
default-host	default-host	The server's default virtual host.
servlet-container	default	The server's default servlet container.

http-listener Attributes

Table A.66. http-listener Attributes

Attribute	Default	Description
allow-encoded-slash	false	If a request comes in with encoded characters, for example **%2F**, whether these will be decoded.
allow-equals-in-cookie-value	false	Whether to allow non-escaped equals characters in unquoted cookie values. Unquoted cookie values may not contain equals characters. If present the value ends before the equals sign. The remainder of the cookie value will be dropped.
always-set-keep-alive	true	Whether a Connection: keep-alive header will be added to responses, even when it is not strictly required by the specification.
buffer-pipelined-data	false	Whether to buffer pipelined requests.
buffer-pool	default	The listener's buffer pool.

Attribute	Default	Description
certificate-forwarding	false	Whether certificate forwarding should be enabled. If this is enabled then the listener will take the certificate from the **SSL_CLIENT_CERT** attribute. This should only be enabled if behind a proxy, and the proxy is configured to always set these headers.
decode-url	true	Whether the parser will decode the URL and query parameters using the selected character encoding (UTF-8 by default). If this is false they will not be decoded. This will allow a later handler to decode them into whatever charset is desired.
disallowed-methods	["TRACE"]	A comma-separated list of HTTP methods that are not allowed.
enable-http2	false	Whether to enable HTTP/2 support for this listener.
enabled	true	Whether the listener is enabled.
http2-enable-push	true	Whether server push is enabled for this connection.
http2-header-table-size		The size of the header table used for HPACK compression, in bytes. This amount of memory will be allocated per connection for compression. Larger values use more memory but may give better compression.
http2-initial-window-size		The flow control window size that controls how quickly the client can send data to the server.
http2-max-concurrent-streams		The maximum number of HTTP/2 streams that can be active at any time on a single connection.
http2-max-frame-size		The maximum HTTP/2 frame size.
http2-max-header-list-size		The maximum size of request headers the server is prepared to accept.
max-buffered-request-size	16384	Maximum size of a buffered request, in bytesRequests are not usually buffered, the most common case is when performing SSL renegotiation for a POST request, and the post data must be fully buffered in order to perform the renegotiation.
max-connections		The maximum number of concurrent connections.
max-cookies	200	The maximum number of cookies that will be parsed. This is used to protect against hash vulnerabilities.

Attribute	Default	Description
max-header-size	1048576	The maximum size in bytes of a HTTP request header.
max-headers	200	The maximum number of headers that will be parsed. This is used to protect against hash vulnerabilities.
max-parameters	1000	The maximum number of parameters that will be parsed. This is used to protect against hash vulnerabilities. This applies to both query parameters, and to POST data, but is not cumulative. For example, you can potentially have max parameters * 2 total parameters).
max-post-size	10485760	The maximum size of a post that will be accepted.
no-request-timeout	60000	The length of time in milliseconds that the connection can be idle before it is closed by the container.
proxy-address-forwarding	false	Whether to enable x-forwarded-host and similar headers and set a remote IP address and host name.
read-timeout		Configure a read timeout for a socket, in milliseconds. If the given amount of time elapses without a successful read taking place, the socket's next read will throw a {@link ReadTimeoutException}.
receive-buffer		The receive buffer size.
record-request-start-time	false	Whether to record the request start time, to allow for request time to be logged. This has a small but measurable performance impact.
redirect-socket		If this listener is supporting non-SSL requests, and a request is received for which a matching requires SSL transport, whether to automatically redirect the request to the socket binding port specified here.
request-parse-timeout		The maximum amount of time in milliseconds that can be spent parsing the request.
resolve-peer-address	false	Enables host DNS lookup.
send-buffer		The send buffer size.

Attribute	Default	Description
socket-binding		The listener's socket binding
tcp-backlog		Configure a server with the specified backlog.
tcp-keep-alive		Configure a channel to send TCP keep-alive messages in an implementation-dependent manner.
url-charset	UTF-8	URL charset.
worker	default	The listener's XNIO worker.
write-timeout		Configure a write timeout for a socket, in milliseconds. If the given amount of time elapses without a successful write taking place, the socket's next write will throw a {@link WriteTimeoutException}.

The following attributes are read-only and only available when statistics are enabled for the `undertow` subsystem:

Table A.67. http-listener Metric Attributes

Attribute	Default	Description
bytes-received		The number of bytes that have been received by this listener.
bytes-sent		The number of bytes that have been sent out on this listener.
error-count		The number of 500 responses that have been sent by this listener.
max-processing-time		The maximum processing time taken by a request on this listener.
processing-time		The total processing time of all requests handed by this listener.
request-count		The number of requests this listener has served.

https-listener Attributes

Table A.68. https-listener Attributes

Attribute	Default	Description
allow-encoded-slash	false	If a request comes in with encoded characters, for example **%2F**, whether these will be decoded.
allow-equals-in-cookie-value	false	Whether to allow non-escaped equals characters in unquoted cookie values. Unquoted cookie values may not contain equals characters. If present the value ends before the equals sign. The remainder of the cookie value will be dropped.
always-set-keep-alive	true	Whether a Connection: keep-alive header will be added to responses, even when it is not strictly required by the specification.
buffer-pipelined-data	false	Whether to buffer pipelined requests.
buffer-pool	default	The listener's buffer pool.
decode-url	true	Whether the parser will decode the URL and query parameters using the selected character encoding (UTF-8 by default). If this is false they will not be decoded. This will allow a later handler to decode them into whatever charset is desired.
disallowed-methods	["TRACE"]	A comma-separated list of HTTP methods that are not allowed.
enable-http2	false	Enables HTTP/2 support for this listener.
enable-spdy	false	Enables SPDY support for this listener.
enabled	true	If the listener is enabled.
enabled-cipher-suites		Configures Enabled SSL ciphers.
enabled-protocols		Configures SSL protocols.
http2-enable-push	true	If server push is enabled for this connection.
http2-header-table-size		The size of the header table used for HPACK compression, in bytes. This amount of memory will be allocated per connection for compression. Larger values use more memory but may give better compression.
http2-initial-window-size		The flow control window size that controls how quickly the client can send data to the server.

Attribute	Default	Description
http2-max-concurrent-streams		The maximum number of HTTP/2 streams that can be active at any time on a single connection.
http2-max-frame-size		The maximum HTTP/2 frame size.
http2-max-header-list-size		The maximum size of request headers the server is prepared to accept.
max-buffered-request-size	16384	Maximum size of a buffered request, in bytesRequests are not usually buffered, the most common case is when performing SSL renegotiation for a POST request, and the post data must be fully buffered in order to perform the renegotiation.
max-connections		The maximum number of concurrent connections.
max-cookies	100	The maximum number of cookies that will be parsed. This is used to protect against hash vulnerabilities.
max-header-size	1048576	The maximum size in bytes of a HTTP request header.
max-headers	200	The maximum number of headers that will be parsed. This is used to protect against hash vulnerabilities..
max-parameters	1000	The maximum number of parameters that will be parsed. This is used to protect against hash vulnerabilities. This applies to both query parameters, and to POST data, but is not cumulative. For example, you can potentially have max parameters * 2 total parameters.
max-post-size	10485760	The maximum size of a post that will be accepted.
no-request-timeout	60000	The length of time in milliseconds that the connection can be idle before it is closed by the container.
read-timeout		Configure a read timeout for a socket, in milliseconds. If the given amount of time elapses without a successful read taking place, the socket's next read will throw a {@link ReadTimeoutException}.
receive-buffer		The receive buffer size.

Attribute	Default	Description
record-request-start-time	false	Whether to record the request start time, to allow for request time to be logged. This has a small but measurable performance impact.
request-parse-timeout		The maximum amount of time in milliseconds that can be spent parsing the request.
resolve-peer-address	false	Enables host DNS lookup.
security-realm		The listener's security realm.
send-buffer		The send buffer size.
socket-binding		The listener's socket binding.
ssl-session-cache-size		The maximum number of active SSL sessions.
ssl-session-timeout		The timeout for SSL sessions, in seconds.
tcp-backlog		Configure a server with the specified backlog.
tcp-keep-alive		Configure a channel to send TCP keep-alive messages in an implementation-dependent manner.
url-charset	UTF-8	URL charset.
verify-client	NOT_REQUESTED	The desired SSL client authentication mode for SSL channels.
worker	default	The listener's XNIO worker.
write-timeout		Configure a write timeout for a socket, in milliseconds. If the given amount of time elapses without a successful write taking place, the socket's next write will throw a {@link WriteTimeoutException}.

The following attributes are read-only and only available when statistics are enabled for the `undertow` subsystem:

Table A.69. https-listener Metric Attributes

Attribute	Default	Description
bytes-received		The number of bytes that have been received by this listener.

Attribute	Default	Description
bytes-sent		The number of bytes that have been sent out on this listener.
error-count		The number of 500 responses that have been sent by this listener.
max-processing-time		The maximum processing time taken by a request on this listener.
processing-time		The total processing time of all requests handed by this listener.
request-count		The number of requests this listener has served.

ajp-listener Attributes

Table A.70. ajp-listener Attributes

Attribute	Default	Description
allow-encoded-slash	false	If a request comes in with encoded characters, for example **%2F**, whether these will be decoded.
allow-equals-in-cookie-value	false	Whether to allow non-escaped equals characters in unquoted cookie values. Unquoted cookie values may not contain equals characters. If present the value ends before the equals sign. The remainder of the cookie value will be dropped.
always-set-keep-alive	true	Whether a Connection: keep-alive header will be added to responses, even when it is not strictly required by the specification.
buffer-pipelined-data	false	Whether to buffer pipelined requests.
buffer-pool	default	The AJP listener's buffer pool.
decode-url	true	If this is true then the parser will decode the URL and query parameters using the selected character encoding (UTF-8 by default). If this is false they will not be decoded. This will allow a later handler to decode them into whatever charset is desired.
disallowed-methods	["TRACE"]	A comma-separated list of HTTP methods that are not allowed.
enabled	true	If the listener is enabled.

Attribute	Default	Description
max-ajp-packet-size		The maximum supported size of AJP packets. If this is modified it has be increased on the load balancer and the back-end server.
max-buffered-request-size	16384	Maximum size of a buffered request, in bytesRequests are not usually buffered, the most common case is when performing SSL renegotiation for a POST request, and the post data must be fully buffered in order to perform the renegotiation.
max-connections		The maximum number of concurrent connections.
max-cookies	200	The maximum number of cookies that will be parsed. This is used to protect against hash vulnerabilities.
max-header-size	1048576	The maximum size in bytes of a HTTP request header.
max-headers	200	The maximum number of headers that will be parsed. This is used to protect against hash vulnerabilities.
max-parameters	100	The maximum number of parameters that will be parsed. This is used to protect against hash vulnerabilities. This applies to both query parameters, and to POST data, but is not cumulative. For example, you can potentially have max parameters * 2 total parameters.
max-post-size	10485760	The maximum size of a post that will be accepted
no-request-timeout	60000	The length of time in milliseconds that the connection can be idle before it is closed by the container.
read-timeout		Configure a read timeout for a socket, in milliseconds. If the given amount of time elapses without a successful read taking place, the socket's next read will throw a {@link ReadTimeoutException}.
receive-buffer		The receive buffer size.
record-request-start-time	false	Whether to record the request start time, to allow for request time to be logged. This has a small but measurable performance impact.

Attribute	Default	Description
redirect-socket		If this listener is supporting non-SSL requests, and a request is received for which a matching requires SSL transport, whether to automatically redirect the request to the socket binding port specified here.
request-parse-timeout		The maximum amount of time in milliseconds that can be spent parsing the request.
resolve-peer-address	false	Enables host DNS lookup.
scheme		The listener scheme, can be HTTP or HTTPS. By default the scheme will be taken from the incoming AJP request.
send-buffer		The send buffer size.
socket-binding		The AJP listener's socket binding.
tcp-backlog		Configure a server with the specified backlog.
tcp-keep-alive		Configure a channel to send TCP keep-alive messages in an implementation-dependent manner.
url-charset	UTF-8	URL charset.
worker	default	The listener's XNIO worker.
write-timeout		Configure a write timeout for a socket, in milliseconds. If the given amount of time elapses without a successful write taking place, the socket's next write will throw a {@link WriteTimeoutException}.

The following attributes are read-only and only available when statistics are enabled for the `undertow` subsystem:

Table A.71. ajp-listener Metric Attributes

Attribute	Default	Description
bytes-received		The number of bytes that have been received by this listener.
bytes-sent		The number of bytes that have been sent out on this listener.

Attribute	Default	Description
error-count		The number of 500 responses that have been sent by this listener.
max-processing-time		The maximum processing time taken by a request on this listener.
processing-time		The total processing time of all requests handed by this listener.
request-count		The number of requests this listener has served.

host Attributes

Table A.72. host Attributes

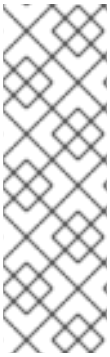
Attribute	Default	Description
alias		Comma-separated list of aliases for the host.
default-response-code	404	If set, this will be response code sent back in case requested context does not exist on server.
default-web-module	ROOT.war	Default web module.
disable-console-redirect	false	if set to true, /console redirect wont be enabled for this host.

filter-ref Attributes

Table A.73. filter-ref Attributes

Attribute	Default	Description
predicate		Predicates provide a simple way of making a true/false decision based on an exchange. Many handlers have a requirement that they be applied conditionally, and predicates provide a general way to specify a condition.
priority	1	Defines filter order. It should be set to 1 or more. A higher number instructs the server to be included earlier in the handler chain than others under the same context.

access-log Attributes



NOTE

When using the management CLI to manage the **access-log** element, it is available under **settings** in the **host** element. For example:

```
/subsystem=undertow/server=default-server/host=default-
host/setting=access-log:add
/subsystem=undertow/server=default-server/host=default-
host/setting=access-log:read-resource
```

Table A.74. access-log Attributes

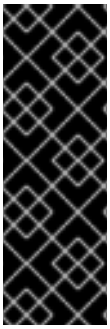
Attribute	Default	Description
directory	${jboss.server.log.dir}	The directory in which to save logs.
extended	false	Whether the log uses the extended log file format.
pattern	common	The access log pattern.
predicate		Predicate that determines if the request should be logged.
prefix	access_log.	Prefix for the log file name.
relative-to		The directory the path is relative to.
rotate	true	Whether to rotate the access log every day.
suffix	log	Suffix for the log file name.
use-server-log	false	Whether the log should be written to the server log, rather than a separate file.
worker	default	Name of the worker to use for logging.

single-sign-on Attributes

NOTE

When using the management CLI to manage the **single-sign-on** element, it is available under **settings** in the **host** element. For example:

```
/subsystem=undertow/server=default-server/host=default-
host/setting=single-sign-on:add
/subsystem=undertow/server=default-server/host=default-
host/setting=single-sign-on:read-resource
```



IMPORTANT

While distributed single sign-on is no different from an application perspective from previous versions of JBoss EAP, in JBoss EAP 7 the caching and distribution of authentication information is handled differently. For JBoss EAP 7, when running the *HA* profile, by default each host will have its own Infinispan cache which will store the relevant session and SSO cookie information. This cache is based on the default cache of the web cache container. JBoss EAP will also handle propagating information between all hosts' individual caches.

Table A.75. single-sign-on Attributes

Attribute	Default	Description
cookie-name	JSESSIONIDSSO	Name of the cookie.
domain		The cookie domain that will be used.
http-only	false	Set cookie httpOnly attribute.
path	/	Cookie path.
secure	false	Set cookie secure attribute.

location Attributes

Table A.76. location Attributes

Attribute	Default	Description
handler		Default handler for this location.

A.24. DEFAULT BEHAVIOR OF HTTP METHODS

Compared to the `web` subsystem in previous JBoss EAP releases, the `undertow` subsystem in JBoss EAP 7.0 has different default behaviors of HTTP methods. The following table outlines the default behaviors in JBoss EAP 7.0.

Table A.77. HTTP Method Default Behavior

HTTP Method	JSP	Servlet	Static HTML
GET	OK	Depends on its implementation	OK
POST	OK	Depends on its implementation	NOT_ALLOWED

HTTP Method	JSP	Servlet	Static HTML
HEAD	OK	Depends on its implementation	OK
PUT	NOT_ALLOWED	Depends on its implementation	NOT_ALLOWED
TRACE	NOT_ALLOWED	NOT_ALLOWED	NOT_ALLOWED
DELETE	NOT_ALLOWED	Depends on its implementation	NOT_ALLOWED
OPTIONS	NOT_ALLOWED	Depends on its implementation	OK

A.25. IO SUBSYSTEM ATTRIBUTES

Table A.78. worker Attributes

Attribute	Default	Description
io-threads		Number of IO threads to use.
stack-size	0	Size of the stack.
task-keepalive	60	Keepalive time for a task. *This attribute should not be used as it is currently ignored.*
task-max-threads		Maximum number of threads for a task.

Table A.79. buffer-pool Attributes

Attribute	Default	Description
buffer-size		Size of the buffer.
buffers-per-slice		How many buffers per slice.
direct-buffers		Does the buffer pool use direct buffers.

A.26. REMOTING SUBSYSTEM ATTRIBUTES

Table A.80. remoting Attributes

Attribute	Default	Description
worker-read-threads	1	The number of read threads to create for the remoting worker.
worker-task-core-threads	4	The number of core threads for the remoting worker task thread pool.
worker-task-keepalive	60	The number of milliseconds to keep non-core remoting worker task threads alive.
worker-task-limit	16384	The maximum number of remoting worker tasks to allow before rejecting.
worker-task-max-threads	16	The maximum number of threads for the remoting worker task thread pool.
worker-write-threads	1	The number of write threads to create for the remoting worker.

IMPORTANT

The above attributes of the *remoting* element are deprecated. These attributes should now be configured using the `io` subsystem.

Table A.81. endpoint Attributes

Attribute	Default	Description
auth-realm		The authentication realm to use if no authentication CallbackHandler is specified.
authentication-retries	3	Specify the number of times a client is allowed to retry authentication before closing the connection.
authorize-id		The SASL authorization ID. Used as authentication user name to use if no authentication CallbackHandler is specified and the selected SASL mechanism demands a user name.
buffer-region-size		The size of allocated buffer regions.
heartbeat-interval	2147483647	The interval to use for connection heartbeat, in milliseconds. If the connection is idle in the outbound direction for this amount of time, a ping message will be sent, which will trigger a corresponding reply message.

Attribute	Default	Description
max-inbound-channels	40	The maximum number of concurrent inbound messages on a channel.
max-inbound-message-size	9223372036854775807	The maximum inbound message size to be allowed. Messages exceeding this size will cause an exception to be thrown on the reading side as well as the writing side.
max-inbound-messages	80	The maximum number of inbound channels to support for a connection.
max-outbound-channels	40	The maximum number of concurrent outbound messages on a channel.
max-outbound-message-size	9223372036854775807	The maximum outbound message size to send. No messages larger than this well be transmitted; attempting to do so will cause an exception on the writing side.
max-outbound-messages	65535	The maximum number of outbound channels to support for a connection.
receive-buffer-size	8192	The size of the largest buffer that this endpoint will accept over a connection.
receive-window-size	131072	The maximum window size of the receive direction for connection channels, in bytes.
sasl-protocol	*remoting*	Where a SaslServer or SaslClient are created by default the protocol specified is *remoting*, this can be used to override this.
send-buffer-size	8192	The size of the largest buffer that this endpoint will transmit over a connection.
server-name		The server side of the connection passes it's name to the client in the initial greeting, by default the name is automatically discovered from the local address of the connection or it can be overridden using this.
transmit-window-size	131072	The maximum window size of the transmit direction for connection channels, in bytes.
worker	*default*	Worker to use

NOTE

When using the management CLI to update the *endpoint* element, it is available under *configuration* in the *remoting* element e.g.: `/subsystem=remoting/configuration=endpoint/`.

Connector Attributes

The connector component has the following structure:

- connector
  - property
  - security
    - sasl
      - property
      - sasl-policy
        - policy

Table A.82. connector Attributes

Attribute	Default	Description
authentication-provider		The *authentication-provider* element contains the name of the authentication provider to use for incoming connections.
sasl-protocol	*remote*	The protocol to pass into the SASL mechanisms used for authentication.
security-realm		The associated security realm to use for authentication for this connector.
server-name		The server name to send in the initial message exchange and for SASL based authentication.
socket-binding		The name (or names) of the socket binding(s) to attach to.

Table A.83. property Attributes

Attribute	Default	Description
value		The property value.

Security Attributes

The *security* component allows you to configure the security for the connector, but contains no direct configuration attributes. It can be configured using its nested components, such as sasl.

Table A.84. sasl Attributes

Attribute	Default	Description
include-mechanisms		The optional nested *include-mechanisms* element contains a whitelist of allowed SASL mechanism names. No mechanisms will be allowed which are not present in this list.
qop		The optional nested *qop* element contains a list of quality-of-protection values, in decreasing order of preference.
reuse-session	false	The optional nested *reuse-session* boolean element specifies whether or not the server should attempt to reuse previously authenticated session information. The mechanism may or may not support such reuse, and other factors may also prevent it.
server-auth	false	The optional nested *server-auth* boolean element specifies whether the server should authenticate to the client. Not all mechanisms may support this setting.
strength		The optional nested "strength" element contains a list of cipher strength values, in decreasing order of preference.

sasl-policy Attributes

The *sasl-policy* component allows you to specify an optional policy to use to narrow down the available set of mechanisms, but contains no direct configuration attributes. It can be configured using its nested components, such as policy.

Table A.85. policy Attributes

Attribute	Default	Description
forward-secrecy	true	The optional nested *forward-secrecy* element contains a boolean value which specifies whether mechanisms that implement forward secrecy between sessions are required. Forward secrecy means that breaking into one session will not automatically provide information for breaking into future sessions.
no-active	true	The optional nested *no-active* element contains a boolean value which specifies whether mechanisms susceptible to active (non-dictionary) attacks are not permitted. *false* to permit, *true* to deny.

Attribute	Default	Description
no-anonymous	true	The optional nested *no-anonymous* element contains a boolean value which specifies whether mechanisms that accept anonymous login are permitted. *false* to permit, *true* to deny.
no-dictionary	true	The optional nested *no-dictionary* element contains a boolean value which specifies whether mechanisms susceptible to passive dictionary attacks are permitted. *false* to permit, *true* to deny.
no-plain-text	true	The optional nested *no-plain-text* element contains a boolean value which specifies whether mechanisms susceptible to simple plain passive attacks (e.g., *PLAIN*) are not permitted. *false* to permit, *true* to deny.
pass-credentials	true	The optional nested *pass-credentials* element contains a boolean value which specifies whether mechanisms that pass client credentials are required.

HTTP Connector Attributes

The http-connector component has the following structure:

- http-connector
  - property (same as connector)
  - security (same as connector)
    - sasl (same as connector)
      - property (same as connector)
      - sasl-policy (same as connector)
        - policy (same as connector)

Table A.86. http-connector Attributes

Attribute	Default	Description
authentication-provider		The *authentication-provider* element contains the name of the authentication provider to use for incoming connections.
connector-ref		The name (or names) of a connector in the `undertow` subsystem to connect to.

Attribute	Default	Description
sasl-protocol	*remote*	The protocol to pass into the SASL mechanisms used for authentication.
security-realm		The associated security realm to use for authentication for this connector.
server-name		The server name to send in the initial message exchange and for SASL based authentication.

Outbound Connection Attributes

The **`outbound-connection`** component has the following structure:

- outbound-connection
  - property

Table A.87. outbound-connection Attributes

Attribute	Default	Description
uri		The connection URI for the outbound connection.

Table A.88. property Attributes

Attribute	Default	Description
value		The property value.

NOTE

The above **`property`** attributes are related to the XNIO Options that will be used during the connection creation.

Remote Outbound Connection

The **`remote-outbound-connection`** component has the following structure:

- remote-outbound-connection
  - property (same as outbound-connection)

Table A.89. remote-outbound-connection Attributes

Attribute	Default	Description
outbound-socket-binding-ref		Name of the **`outbound-socket-binding`** which will be used to determine the destination address and port for the connection.

Attribute	Default	Description
protocol	*http-remoting*	The protocol to use for the remote connection. Defaults to **`http-remoting`**.
security-realm		Reference to the security realm to use to obtain the password and SSL configuration.
username		The user name to use when authenticating against the remote server.

Local Outbound Connection Attributes

The **`local-outbound-connection`** component has the following structure:

- local-outbound-connection
  - property (same as outbound-connection)

Table A.90. local-outbound-connection Attributes

Attribute	Default	Description
outbound-socket-binding-ref		Name of the **`outbound-socket-binding`** which will be used to determine the destination address and port for the connection.

A.27. APACHE HTTP SERVER MOD_CLUSTER DIRECTIVES

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 directives can be set to configure mod_cluster.

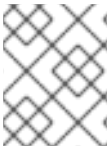
NOTE

There is no need to use *ProxyPass* directives because mod_cluster automatically configures the URLs that must be forwarded to Apache HTTP Server.

Table A.91. mod_cluster Directives

Directive	Description	Values

Directive	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)*
UseAlias	Check that the alias corresponds to the server name.	• **0**: Ignore aliases *(default)* • **1**: Check aliases
LBstatusRecalTime	Time interval in seconds for load-balancing 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
ProxyPassMatch/ProxyPass	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: `ProxyPassMatch ^(/.*\.gif)$ !` This example allows the Apache HTTP Server to serve the .gif files directly.	



NOTE

Due to performance optimizations for sessions in JBoss EAP 7, configuring hot-standby nodes is not supported.

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 A.92. mod_manager Directives

Directive	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.

Directive	Description	Values
MaxMCMPMaxMessSize	The maximum size of MCMP messages from other Max directives	Calculated from other Max directives. Min: **1024**
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.
	
	`<Location /mod_cluster-manager>`
	`  SetHandler mod_cluster-manager`
	`  Require ip 127.0.0.1`
	`</Location>`
		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.

A.28. MODCLUSTER SUBSYSTEM ATTRIBUTES

The `modcluster` subsystem has the following structure:

- mod-cluster-config
  - dynamic-load-provider
    - custom-load-metric
    - load-metric
  - ssl

Table A.93. mod-cluster-config Configuration Options

Attribute	Default	Description
advertise	true	Whether or not advertising is enabled.
advertise-security-key		String containing the security key for the Advertise logic.
advertise-socket		Name of Socket binding to use for the Advertise socket.
auto-enable-contexts	true	If set to `false`, contexts are registered with the reverse proxy as disabled. You can enable the context using the `enable-context` operation or by using the mod_cluster_manager console.
balancer		The name of the balancer on the reverse proxy to register with. If not set, the value is configured on the Apache HTTP Server side with the `ManagerBalancerName` directive, which defaults to `mycluster`.
connector		The name of Undertow listener that mod_cluster reverse proxy will connect to.

Attribute	Default	Description
excluded-contexts		A list of contexts to exclude from registration with the reverse proxies. If no host is indicated, the host is assumed to be **`localhost`**. **`ROOT`** indicates the root context of the web application.
flush-packets	false	Whether or not to enable packet flushing to the web server.
flush-wait	-1	Time to wait before flushing packets in httpd. Max value is **`2,147,483,647`**.
load-balancing-group		If set, requests are sent to the specified load balancing group on the load balancer.
max-attempts	1	The number of times the reverse proxy will attempt to send a given request to a worker before giving up.
node-timeout	-1	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.
ping	10	Time, in seconds, in which to wait for a pong answer to a ping.
proxies		List of proxies for mod_cluster to register with defined by **`outbound-socket-binding`** in **`socket-binding-group`**.
proxy-list		List of proxies. The format is ***`HOST_NAME:PORT`***, separated with commas. *Deprecated in favor of* ***`proxies`***.
proxy-url	/	Base URL for MCMP requests.

Attribute	Default	Description
session-draining-strategy	DEFAULT	Session draining strategy used during undeployment of a web application. Valid values are **DEFAULT**, **ALWAYS**, or **NEVER**. **DEFAULT** Drain sessions before web application undeploy only if the web application is non-distributable. **ALWAYS** Always drain sessions before web application undeploy, even for distributable web applications. **NEVER** Do not drain sessions before web application undeploy.
simple-load-provider		A simple load provider to use 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.
smax	-1	Soft maximum idle connection count in httpd.
socket-timeout	20	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.
status-interval	10	Number of seconds a STATUS message is sent from the application server to the reverse proxy. Allowed values are between **1** and **2,147,483,647**.
sticky-session	true	Whether subsequent requests for a given session should be routed to the same node, if possible.
sticky-session-force	false	Whether the reverse proxy should return an error in the event that the balancer is unable to route a request to the node to which it is stuck. This setting is ignored if sticky sessions are disabled.
sticky-session-remove	false	Remove session information on failover.
stop-context-timeout	10	The maximum time, in seconds, to wait for a context to process pending requests, for a distributable context, or to destroy active sessions, for a non-distributable context.
ttl	-1	Time to live, in seconds, for idle connections above smax. Allowed values are between **-1** and **2,147,483,647**.

Attribute	Default	Description
worker-timeout	-1	Timeout to wait in httpd for an available worker to process the requests. Allowed values are between **-1** and **2,147,483,647**.

Table A.94. dynamic-load-provider Configuration Options

Attribute	Default	Description
decay	2	Decay.
history	9	History.

Table A.95. custom-load-metric Attribute Options

Attribute	Default	Description
capacity	1.0	Capacity of the metric.
class		Class name of the custom metric.
property		Properties for the metric.
weight	1	Weight of the metric.

Table A.96. load-metric Attribute Options

Attribute	Default	Description
capacity	1.0	Capacity of the metric.
property		Properties for the metric.
type		Type of the metric.
weight	1	Weight of the metric.

Table A.97. ssl Attribute Options

Attribute	Default	Description
ca-certificate-file		Certificate authority.
ca-revocation-url		Certificate authority revocation list.

Attribute	Default	Description
certificate-key-file	${user.home}/.keystore	Key file for the certificate.
cipher-suite		The allowed cipher suite.
key-alias		The key alias.
password	changeit	Password.
protocol	TLS	The SSL protocols that are enabled.

A.29. MOD_JK WORKER PROPERTIES

The **workers.properties** file defines the behavior of the workers to which mod_jk passes client requests. The **workers.properties** file defines where the different application servers are located and the way the workload should be balanced across them.

The general structure of a property is **worker.WORKER_NAME.DIRECTIVE**. The **WORKER_NAME** is a unique name that must match the **instance-id** configured in the JBoss EAP **undertow** subsystem. The **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.

Table A.98. Global properties

Property	Description
worker.list	A comma separated list of worker names that will be used by *mod_jk*.

Table A.99. Mandatory Directives

Property	Description
type	The type of worker. The default type is **ajp13**. Other possible values are **ajp14**, **lb**, **status**. For more information on these directives, see the Apache Tomcat Connectors Reference at https://tomcat.apache.org/connectors-doc/reference/workers.html.

Table A.100. Load Balancing Directives

Property	Description

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.
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.

Table A.101. Connection Directives

Property	Description
host	The host name or IP address of the back-end server. The back-end server must support the *ajp* protocol stack. The default value is*localhost*.
port	The port number of the back-end server instance listening for defined protocol requests. The default value is *8009*, which is the default listening port for AJP13 workers. The default value for AJP14 workers is *8011*.
ping_mode	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. • 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.

Property	Description
lbfactor	Specifies the load-balancing factor for an individual back-end server instance. This is useful to give a more powerful server more of the workload. To give a worker 3 times the default load, set this to *3*: **`worker.my_worker.lbfactor=3`**

The example below demonstrates load balancing with sticky sessions between two worker nodes (**node1** and **node2**) listening on port **8009**.

Example workers.properties File

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

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

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

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

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

Further configuration details for Apache mod_jk are out of the scope of this document and can be found in the Apache documentation.

A.30. SECURITY MANAGER SUBSYSTEM ATTRIBUTES

The **`security-manager`** subsystem itself does not have configurable attributes, but it has one child resource with configurable attributes: **`deployment-permissions=default`**.

Table A.102. default Configuration Options

Attribute	Default	Description
maximum-permissions		The maximum set of permissions that can be granted to a deployment or jars.
minimum-permissions		The minimum set of permissions to be granted to a deployment or jars.

Revised on 2018-02-08 10:15:47 EST