redhat®
TRAINING

CHAPTER 3

SCRIPT CONFIGURATION AND IMPLEMENTATION OF APPLICATIONS

General description	
Meta	Configure JBoss EAP using the Command Line Interface tool and deploy Java EE applications.
Goals	• Connect to a JBoss EAP instance with the CLI and execute general commands. • Deploy a Java EE application to a server instance that runs as a standalone server.
sections	• Configuring JBoss EAP with the Command Line Interface (and guided exercise) • Deploying applications to a stand-alone server (and guided exercise) •
Laboratory work	Configuring scripts and deploying applications

Configuring JBoss EAP with the Command Line Interface

Goals

After completing this section, a system administrator should be able to do the following:

• Connect to a JBoss EAP instance in standalone mode with the CLI and run general commands.

CLI Tool Overview

In JBoss EAP 7, most configurations are in XML files. There are three different options for configuring EAP servers:

1. Administration Console: A web application that modifies the underlying XML configuration files.

2. CLI: The Command Line Interface (CLI) for remotely managing and configuring EAP instances from a command line, and even writing scripts for repetitive tasks.

3. XML: Editing the XML configuration files directly is possible, but not considered a good practice.

Some of the benefits of using the CLI include the following:

• Manage servers without needing a GUI.

• Commands can be written to a separate file and executed as a batch, allowing you to write scripts for common tasks.

• The management console has limitations, but the CLI configures almost all aspects of its XML configuration files.

CLI Admin Only Mode

JBoss EAP 7 introduced a new feature to the CLI. It is now possible to embed an EAP server instance within the CLI process. Running an instance of embedded EAP 7 servers within the CLI process means that you have access to the same CLI remote administration commands without having to connect to a remote server.

The main goal of using this approach is to enable local EAP management without requiring a socket-based connection. This can be very useful when, for security reasons, the management port (9990) is not available.

Another example of why this is useful is being able to take the configuration offline and apply it in production only when the upgrade can be done.

To use this function, it is necessary to start the CLI without supplying the connection parameter:

```
# $JBOSS_HOME/bin/jboss-cli.sh
```

Then it is possible to start the embedded server:

```
[disconnected /] embed-server --server-config=standalone-ha.xml
```

In the example above, each change made by the CLI will be persisted in the standalone-ha.xml file, which is available in the JBOSS_HOME/standalone/ configuration folder. If the --server-config parameter is not specified by default, standalone.xml will contain the changes.

It is possible to start with an empty configuration using the --empty-config parameter.

```
[disconnected /] embed-server --server-config=production-ha.xml --empty-config
```

The production-ha.xml file is created in the JBOSS_HOME/standalone/ configuration folder. This file only contains the XML tags related to the configurations made in the CLI. If the file already exists, the command fails to prevent accidental deletion of a configuration file.

Important

The file specified in the --server-config parameter must exist in the JBOSS_HOME/standalone/configuration folder, if *--empty-config* **is not specified.**

Introduction of DMR syntax

EAP 6 introduced a new syntax with a reduced number of representation types for the underlying Java objects that are used by the administration console and CLI (and any other administration tools that might be used).

This syntax is called *Dynamic Model Representation (DMR),* and EAP7 still uses it.

The DMR syntax is a way to look at EAP values. An understanding of DMR syntax is required to work with the CLI. However, the DMR syntax is intuitive and uses fewer words than XML.

To compare the EAP DMR and XML syntax, the following excerpt is part of the standalone.xml file, logging subsystem configuration:

```
<subsystem xmlns="urn:jboss:domain:logging:3.0">
    <console-handler name="CONSOLE">
        <level name="INFO"/>
        <formatter>
            <named-formatter name="COLOR-PATTERN"/> </
        formatter> </
    console-handler>

    ...remainder of the logging subsystem </
subsystem>
```

Same values in DMR syntax:

```
"logging" => {

    "console-handler" => {"CONSOLE" => {
        "autoflush" => true,
        "enabled" => true,
        "encoding" => undefined, "filter"
        => undefined, "filter-spec" =>
        undefined, "formatter" =>
        "%d{HH:mm:ss,SSS} %-5p [%c] (%t) %s%e%n", "level" => "INFO", "name"
        => "CONSOLE", "named-
        formatter" => "COLOR-
        PATTERN", "target" => "System.out" }},



    ...remainder of the logging subsystem
}
```

One of the advantages of the DMR syntax is the ability to see all attributes, including attributes that have a default value. For example, in the XML syntax, it is not clear that the autoflush attribute has the value true, while this is clear in the DMR syntax.

Note that the DMR syntax has fewer levels than XML. In DMR most values appear at the same level, while in XML values can be attributes or nested within elements. For example, the formatter pattern in XML is several levels deep, whereas in DMR syntax it is simply a property of the CONSOLE handler.

As we progress through this course, the configurations and examples we cover will gradually shift from XML to DMR syntax. The CLI is an important tool designed for both the casual user and the experienced EAP user, and you should be able to read the DMR syntax and be comfortable using it.

command execution

The CLI is started by running the jboss-cli.sh script in the bin folder of the EAP installation. This script does not require any arguments; if no arguments are supplied, it will start in disconnected mode:

```
# $JBOSS_HOME/bin/jboss-cli.sh
[disconnected /]
```

The connect command is responsible for connecting to the server. If no argument is given, it will try to connect to the default host (localhost) using the default port (9990).

```
[disconnected /] connect localhost:9990
```

To automatically connect using the jboss-cli.sh script, use the --connect argument:

```
# $JBOSS_HOME/bin/jboss-cli.sh --connect
```

In the example above, it will try to connect to the default localhost using the default port 9990.

use

The OS user running the CLI must be the same as the one that started the EAP. Authentication is based on creating a file to send a token only once. The EAP instance creates the file; if the CLI can read the file, it sends the token to EAP and the CLI authenticates.

If EAP and CLI were started by different OS users, the CLI will not be able to read the file and local authentication will fail. This returns to authentication, and the CLI prompts for a username and password.

It is possible to use a shortcut to connect to a server:

```
# $JBOSS_HOME/bin/jboss-cli.sh -c
```

To connect to a different host machine, use the --controller argument:

```
# $JBOSS_HOME/bin/jboss-cli.sh --connect --controller=ip:port
```

Important

By default, the CLI will not prompt for credentials if it is running on the same operating system that started the server. Credentials can be passed with the --user and --password arguments.

Operations

Operations are a low-level way of managing the EAP server. If such management cannot be performed with an operation, it means that it cannot be performed.

An operation in the CLI has the following format:

```
[node] : operation_name [ parameters ] [ headers ]
```

node represents the address of the target resource or the node where the operation should be invoked. As a key/value pair, it consists of a node type and the name of the node. The node / subsystem=datasources is a data source.

After specifying the desired node, the name of the operation must be defined with an optional list of parameters. The colon is required before the operation name, as it serves as a separator between the node and the operation. The colon is required even if the node is empty. In this case, the operation will be executed at the current node level.

The following operations are very common:

- :read-resource: reads the values of a model resource attribute and basic information or complete on any secondary resources.

- :read-operation-names – Gets the names of all operations for a resource determined.
- :read-operation-description: Gets the description of a given operation.
- :reload – reloads the server by shutting down all its services and starting them again. The JVM itself has not been restarted.
- :read-attribute: Gets the value of an attribute for the selected resource.
- :write-attribute – sets the value of an attribute for the selected resource.
- :remove – Removes the node.

Commands

Commands contain a simple syntax and most of them translate into operation requests.

The following commands are the most basic supported commands for the CLI:

- cn or cd: change the path of the current node to the argument.
- connect – Connect to the server or domain controller.
- data-source: used to manage subsystem=datasources/data resources source.
- deploy: deploy an application.
- help: display the help page.
- history: print or disable, enable or clear history expansion.
- ls – List the contents of the node path.
- pwn or pwd: print the current working node.
- exit or quit – Quit the command line interface.
- undeploy – Undeploy an application.
- version – Print the version and environment information.

Tab Completion

Tab completion shows all possible commands available at any time, in a current command. For example, enter / y, then press the Tab key to see all the values you can enter after /.

```
[standalone@localhost:9990 /] / core-service
deployment-overlay                         interface        socket-binding-group
 system-property
deployment             extension           path             subsystem
```

Start typing interface after / and press Tab, noting that the CLI not only fills in the interface sublevel, but also adds an equals sign because that is the only possible value after / interface.

Press the Tab key again and all the interfaces will appear:

```
[standalone@localhost:9990 /] /interface= management
public
```

Execution of a script file in the CLI EAP 7
supports the use of a text file in the CLI as a script. Using this approach, it is possible to create scripts for repetitive tasks. For example, it is possible to create a script file that configures a data source and tests a pool connection:

```
/subsystem=datasources/data-source=appDs:add\ (jndi-
 name=java:jboss/datasources/appDS,driver-name=h2,user-name=jb248, \ password=jb248,connection

url="jdbc:h2:mem:app;DB_CLOSE_DELAY=-1;DB_CLOSE_ON_EXIT=FALSE") /
subsystem=datasources/data-source=appDs:test-connection-in-pool
```

If the above content is available in the /home/student/create datasource.cli file, it can be run with:

```
# $JBOSS_HOME/jboss-cli.sh --connect --controller=localhost:9990 \ --file=/home/
 student/create-datasource.cli
```

The CLI has the batch command that supports multiple commands that are executed as a single atomic unit. If at least one of the commands or operations fails, all other successfully executed commands and operations in the batch are rolled back.

In the example above, it's a good idea to create the data source using the batch command. If the connection test for a set fails, the data source creation will be rolled back. To run a batch, use the run-batch command:

```
batch /
subsystem=datasources/data-source=appDs:add\ (jndi-
 name=java:jboss/datasources/appDS,driver-name=h2,user-name=jb248,\ password=jb248,connection

url="jdbc:h2:mem:app;DB_CLOSE_DELAY=-1;DB_CLOSE_ON_EXIT=FALSE") /
subsystem=datasources/data-source=appDs:test-connection-in-pool run-batch
```

Running a Command from an External Script The
CLI provides a feature that allows integration with external scripts. It is possible to send a command or an operation to the CLI using the --command attribute:

```
# $JBOSS_HOME/jboss-cli.sh -c --controller=localhost:9990 \ --command="/
 subsystem=datasources:read-resource"
```

The result of the operation will be returned using the DMR syntax in the script that invoked the CLI. It is possible to specify a set of commands using the commands attribute. This attribute specifies a comma-separated list (the list must not contain white space) of commands and operations to be executed in the CLI session.

```
# ./jboss-cli.sh -c --controller=localhost:9990 --command="cd /subsystem=datasources,ls"
```

Demo: Running CLI Commands

1. Open a terminal window from the workstation virtual machine (Applications > Favorites > Terminal) and run the following command to verify that EAP has been installed and is not currently running, and to create a new base directory for EAP:

```
[student@workstation ~]$ demo executing-cli setup
```

The base directory is copied from the /opt/jboss-eap-7.0/standalone directory. This will be used to start a new standalone server for you to connect to using the CLI.

2. Start the new standalone server using the /home/student/JB248/labs/ executing-cli directory as the EAP base directory. Before proceeding, wait for the server to finish starting up.

```
[student@workstation ~]$ cd /opt/jboss-eap-7.0/bin
[student@workstation bin]$ ./standalone.sh \
-Djboss.server.base.dir=/home/student/JB248/labs/executing-cli
```

3. Open a new terminal window and start the CLI by running the jboss-cli.sh script in the EAP bin directory. Use the --controller property to specify the host and port of the EAP instance to which you want to connect.

```
[student@workstation ~]$ cd /opt/jboss-eap-7.0/bin
[student@workstation bin]$ ./jboss-cli.sh --connect \ --
controller=localhost:9990
```

4. A common and useful task performed with the CLI is to read resource properties to discover or verify their configuration.

Input to the CLI is arranged in a hierarchical structure beginning at the standalone server level. For example, to view the values at the top level, enter the following command:

```
[standalone@localhost:9990 /] /:read-resource
```

5. The slash "/" is used to separate levels. The following command, for example, reads the resources of a public interface:

```
[standalone@localhost:9990 /] /interface=public:read-resource
```

A level is similar to a folder in the Linux operating system. In the example above, it is possible to make the analogy that the public interface resource is a file inside the / folder. It means that / is a top level and the public interface is a child level.

6. You can continue to access lower levels of the hierarchy using the forward slash (slash). For example:

```
[standalone@localhost:9990 /] /subsystem=ejb3/thread-pool=default:read-resource
```

use

A number of bash CLI shortcuts are also available within the JBoss CLI. For example, Ctrl+U deletes from the cursor position to the start of the line, Ctrl+K deletes from the cursor position to the end of the line, Up Arrow and Down Arrow move forward or backward through the CLI history. Feel free to explore any other combinations that can be used.

7. It is possible to use an asterisk as a wild card. For example, to see all undertow subsystem configurations:

```
[standalone@localhost:9990 /] /subsystem=undertow/configuration=*:read-resource
```

The undertow subsystem is a web server responsible for rendering Java EE-based web pages.

8. Add the recursive flag to the end of :read-resource to view a resource and all its sublevels. For example, compare the result of the following operations:

```
[standalone@localhost:9990 /] /subsystem=undertow:read-resource
[standalone@localhost:9990 /] /subsystem=undertow:read-resource(recursive=true)
```

Notice that with the recursive flag set to true, all child elements of the undertow resource are displayed, along with their child elements, and so on.

9. It is possible to see all the resources on the server, by simply booting to the top level and turning on the recursive capability:

```
[standalone@localhost:9990 /] /:read-resource(recursive=true)
```

The output shows the values for the entire server configuration, which is about 2000 lines long. Redirect the output to a file for better display:

```
[standalone@localhost:9990 /] :read-resource(recursive=true) \ > /tmp/output.txt
```

Note that no output appears in the CLI, but a new file named /tmp/output.txt has to be made available with all your standalone server settings in DMR syntax.

10. A colon is used to perform an operation, such as read-resource. Enter a colon, then press the Tab key, and all available operations for whatever level the operation is running on will appear. For example, at the root level, enter a colon and press the Tab key to see the operations at that level:

```
[standalone@localhost:9990 /] :
```

The following result is expected:

```
add-namespace                         map-put                          read-operation-names
  take-snapshot
add-schema-location                   map-remove                       read-resource
  undefine-attribute
clean-obsolete-content                product-info                     read-resource-description
  upload-deployment-bytes delete-
snapshot  upload-                     query                            reload
  deployment-stream full-replace-
deployment  upload-deployment-        read-attribute                   remove-namespace
  url list-add
                                      read-attribute-group             remove-schema-location
  validate-address list-
clear                                 read-attribute-group-names replace-deployment
  validate-operation list-
get                                   read-children-names              resolve-expression
  whoami
list-remove                           read-children-resources          resolve-internet-address
  write-attribute
list-snapshots  map-                  read-children-types  read-       resume
clear  map-                           config-as-xml  read-             shutdown
get                                   operation-description suspend
```

11. **When an operation is executed, the CLI displays the result in DMR syntax. Try running the product-info operation:**

```
[standalone@localhost:9990 /] :product-info
```

The following result is expected:

```
{
        "outcome" => "success", "result"
        => [{"summary" => {
                "host-name" => "workstation.lab.example.com", "instance-
                identifier" => "b22faa6f-b32b-4e22-8ee6-c6e138e67447", "product-name" => "JBoss
                EAP", "product-version" => "7.0.0.GA",
                "product-community-identifier" => "Product",
                "product-home" => "/opt/jboss-eap-7.0", "standalone-or-
                domain-identifier" => "STANDALONE_SERVER",
                "host-operating-system" => "Red Hat Enterprise Linux Server 7.2 (Maipo)", "host-
                cpu" => { "host-cpu-arch" => "amd64", "host-core-count" => 2



                },
                "jvm" =>
                        { "name" => "OpenJDK 64-Bit Server VM", "java-
                        version" => "1.8", "jvm-version"
                        => "1.8.0_77", "jvm-vendor" => "Oracle
                        Corporation", "java-home" => "/usr/lib/jvm/java-1.8.0-

openjdk-1.8.0.77-0.b03.el7_2.x86_64/jre"
                }
        }}]
}
```

12. Browsing resources in the CLI is similar to browsing a folder system from a Linux terminal. For example, if you need to enter several commands at the subsystem level, you can cd to that level. For example, if you need to run multiple operations on the datasource subsystem, cd /subsystem=datasources at that level. Enter the following commands:

```
[standalone@localhost:9990 /] cd /subsystem=datasources
[standalone@localhost:9990 subsystem=datasources] ./ data-source
jdbc-driver xa-data-source
```

Note the use of ./ to display sublevels relative to the current level.

13. Use the ls command to view a more detailed list of resources at the level where you is currently:

```
[standalone@localhost:9990 subsystem=datasources] ls
```

14. Use the pwd command to print the current worker node:

```
[standalone@localhost:9990 subsystem=datasources] pwd
```

The following result is expected:

```
/subsystem=datasources
```

15. Until now, the CLI has only been used to read information. However, you can use the write-attribute operation to modify the attributes of a resource with the CLI. To demonstrate this operation, change the min-pool-size attribute of the ExampleDS data source resource. We'll discuss data source configuration later, but for now keep in mind that this attribute defines the minimum number of connections to keep open for a data source.

```
[standalone@localhost:9990 /] cd /subsystem=datasources/data-source=ExampleDS
[standalone@localhost:9990 data-source=ExampleDS] :write-attribute\ (name=min-pool-
size,value=5) {"outcome" =>
"success"}
```

Important

The attributes of all resources cannot be written. Use the :read-resource-description command to check if the access-type of the attribute you want to access is read-write. Other valid types include read-only and metric.

16. Use the :read-attribute command and verify that the change has occurred:

```
[standalone@localhost:9990 data-source=ExampleDS] :read-attribute\ (name=min-pool-
size)
```

The following result is expected:

```
{
        "outcome" => "success", "result"
        => 5
}
```

17. View the standalone.xml file for your instance (in the /home/student/JB248/labs/ executing-cli/configuration/standalone.xml folder). Notice in the ExampleDS definition in the datasource subsystem that the <min-pool-size> element is 5, and now you see that the CLI and the underlying XML files are in sync.

18. The CLI add operation is used to add new resources to a configuration. To add a new configuration, start by typing the name of the new resource at the level where the resource should appear. For example, the path element is at the root level. The following declaration adds a new path named files:

```
[standalone@localhost:9990 data-source=ExampleDS] /path=files:add\ (path=/files,relative-
to=user.home)
```

Attributes are defined in an add operation as a comma-separated list of name=value pairs.

A path is a logical name available to an instance that refers to a location on the file system.

19. The CLI remove operation is used to remove resources. To remove a configuration, navigate to the resource and run the remove operation.

```
[standalone@localhost:9990 data-source=ExampleDS] cd /path=files
[standalone@localhost:9990 path=files] :remove
```

20. JBoss EAP 7 has a new management mode for the CLI. Now it is possible Start an embedded instance of the CLI. This is useful when you need to prepare some configuration without changing the production environment. Exit the CLI and stop the running instance.

21. Run the CLI without the -c parameter:

```
[student@workstation bin]$ sudo -u jboss /opt/jboss-eap-7.0/bin/jboss-cli.sh
```

Important

Because the files in /opt/jboss-eap-7.0 are owned by the jboss user, use the sudo -u jboss command to start the CLI to ensure that you have the proper permissions to write configurations.

22. The embed-server command is responsible for starting an embedded server for offline configuration. By default, the changes will be applied in the standalone.xml configuration file available in the standalone folder of the JBoss installation path. Start the embedded server:

```
[disconnected /] embed-server
```

use

It is possible to define the desired file with the --server-config parameter. The file must already exist in the standalone folder.

use

It is also possible to start an embedded process to configure managed domain mode with the embed-host-controller command.

23. Create a new system property called course, whose value is JB248, and exit the CLI:

```
[standalone@embedded /] /system-property=course:add(value=JB248)
[standalone@embedded /] exit
```

24. Check in the /opt/jboss-eap-7.0/standalone/configuration/standalone.xml file that the new system property is available.

```
...
  </extensions>
     <system-properties>
        <property name="course" value="JB248"/>
     </system-properties>
  <management>
...
```

25. Offline mode is started with an empty configuration file using the --empty-config parameter:

```
[student@workstation bin]$ sudo ./jboss-cli.sh [disconnected /]
embed-server --server-config=my-config.xml --empty-config
```

26. Create a new system property named country, whose value is US, and exit from the CLI:

```
[standalone@embedded /] /system-property=country:add(value=US)
[standalone@embedded /] exit
```

27. Verify that the new configuration file has been created with only the new system property defined:

```
[student@workstation bin]$ cat \ /opt/jboss-
eap-7.0/standalone/configuration/my-config.xml
```

The following result is expected:

```
<server xmlns="urn:jboss:domain:4.1">

    <system-properties>
        <property name="country" value="US"/> </system-
    properties>

</server>
```

This concludes the demo.